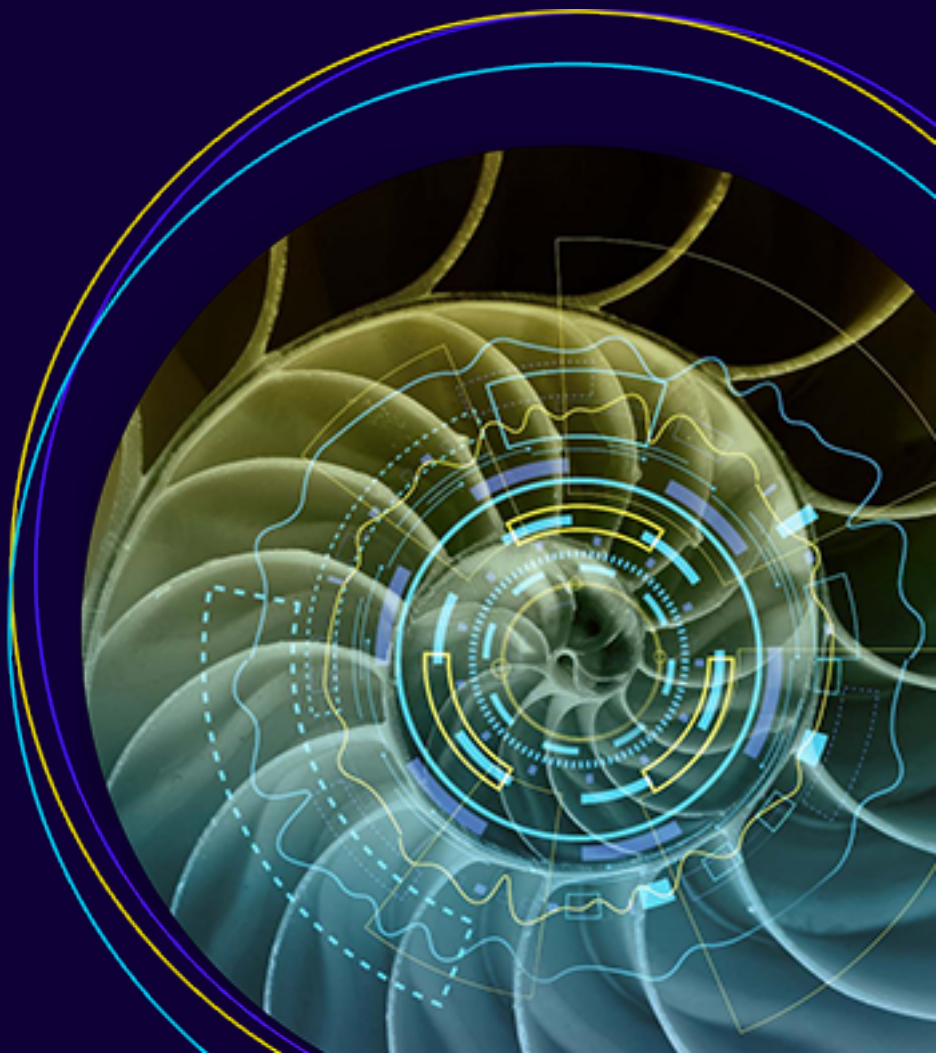
KRYON™

User Guide

Kryon Studio v19.5.1



This document contains proprietary and confidential information of Kryon Systems, and can be distributed only with the prior written consent of Kryon Systems Ltd.

© 2008-2020 Kryon Systems Ltd.
All rights reserved.

Document revision: 09-Feb-2020

Contents

CHAPTER 1: Introduction to Kryon Studio

Glossary 7

CHAPTER 2: The Catalog

Libraries 10

Applications 12

Categories 13

Wizard Properties 14

General Properties 17

Name 18

Description 18

Status 18

Keywords 19

Hyperlink 20

Run modes 23

When ends 25

Applications 25

Related wizards 26

External links 28

Embedded Wizards 30

Notes 31

Change History 32

Version History 33

Advanced Commands Count 34

Restoring the Catalog Home View 35

WIZARD DEVELOPMENT

CHAPTER 3: Creating & Recording Wizards

Creating a Wizard 38

Preparing to Record 40

Recording a Wizard 42

CHAPTER 4: The Wizard Editor

A Tour of the Wizard Editor 47

Wizard Views 49
The Navigation Pane 53
The Flow Pane 55
The Properties Pane 61
CHAPTER 5: Window Detection & Window Options
Window Detection 64
Window Properties for Desktop Applications 66
Window Properties for Web Applications 68
Editing Window Properties 69
Adding a Window Detection Rule 71
Managing Window Detection Rules 73
Adding a Set of Window Detection Data 76
Adding a Visual Object 78
Window Options 80
CHAPTER 6: Object Detection
Detected Object Types 84
Detected Object Display 86
Object Detection Criteria 87
Reusing a Detected Object 90
Reusing a detected object within the same step 90
Reusing a detected object across different steps 90
CHAPTER 7: Editing Wizard Flow
Moving a Step 94
Copying or Duplicating a Step 95
Appending Steps 96
Deleting a Step 97
Adding or Editing an Embedded Wizard 98
Organizing Steps by Color 101
Undoing & Redoing Changes 102
CHAPTER 8: Editing Wizard Steps
Step Start 104
Step End 110

CHAPTER 9: Core Actions

Core Action Types 112
Changing, Restoring, or Deleting a Core Action 115
Core Action Properties 118
Core Action Position 121
Core Action Position: Detected Object 123
Core Action Position: Fixed Position 128
Core Action Position: Relative Position 129
Core Action Position: Offset 130
Core Action Position: Highlight Box 133
Detection Match Configuration 135
Core Action Fallbacks 141

CHAPTER 10: Read from Screen

Using Read from Screen 143
Read from Screen: Position 144
Read from Screen: Value Type 148
Read from Screen: Variable 156

CHAPTER 11: Blocks

Blocks: Overview 158
Adding & Deleting Blocks 160
Block Properties 161
More About Object Blocks 164
Block Duration 166

CHAPTER 12: Bubbles

Bubbles: Overview 168
Bubbles & Mouse/Keyboard Control 169
Working with Bubbles 171
Bubble Types 176
Bubbles: General Properties 180
Bubbles: Show/Hide Properties 182
Bubble Buttons 186

CHAPTER 13: Fallbacks

What is a Fallback? 192
Fallback Events 193
Fallback Commands 194
Applying Fallbacks 195
Fallback Usage & Examples 196

CHAPTER 14: Embedded Wizards

Embedded Wizard Features 201
Embedded Wizard Fallbacks 203

CHAPTER 15: Advanced Commands

Advanced Commands: Introduction 205
Advanced Commands Editor 206
Invoking Advanced Commands 207
Variables 208

CHAPTER 16: Credentials Vault

Accessing the Credentials Vault 211
Types of Credentials 212
Credential Permissions 214
Creating Credentials 215
Using Credentials 218
Attended User Access to Credentials 220

CHAPTER 17: Usage Reports

Report types 222
Generating reports 223

APPENDIX A: Exporting/Importing Catalog Content

Moving Catalog Items Among Environments 226
Exporting Catalog Items (from DEV) 228
Preparing for Import 231
Importing Catalog Items (to TEST or PROD) 235

APPENDIX B: Kryon Plugin Development

Plugin framework 240
Plugin methods 243

Plugin usage 243
Developing a Kryon plugin 245

CHAPTER 1: Introduction to Kryon Studio

Kryon Studio is the tool with which you can create and manage automation content for Kryon Robots. Automation wizards are recorded by the RPA developer in Studio and then accessed by end users who run it on their own computers (attended automation) or by robots who run it on virtual machines (unattended automation).

Glossary

The following terms are essential to understanding Kryon RPA functionality and are used throughout this guide.

Term	Description
Business Intelligence (BI)	The handling of organizational data. BI systems are capable of retrieving data from various database, such as the Kryon Database.
RPA developer	A Kryon Studio user who creates, edits and manages Kryon automations
Robotic Process Automation (RPA)	A form of process automation technology based on the notion of software robots. A robot is a software application that replicates the actions of a human being interacting with the user interface of a computer system. For example, the execution of data entry into an SAP system – or indeed a full end-to-end business process – would be a typical activity for a software robot. The robot operates on the user interface (UI) in the same way that a human would. In addition Kryon's technology can run background activities using web services, SQL queries, DLL and integration technologies (such as SAP, HTML, .NET, Excel, PDFs, emails, etc.)
Index	A number representing the position of an item in a series or array of items. In Kryon Studio, indexes are used in some advanced commands.
Kryon RPA Server	The Kryon RPA Platform is a client-server solution. Kryon robots, running on client desktops/VMs, are connected to a central server (repository) to obtain and execute automation wizards. The Kryon RPA Server is a

Term	Description
	central repository that stores all automations, collects end-user usage statistics, and manages licenses and permissions.
Catalog	A hierarchical tree of libraries, categories and automations for predefined applications
Category	A library subfolder that contains automations
Automations	Wizards and sensors listed in the catalog
Library	The top-level folder in the catalog, containing the categories that, in turn, contain automations
Sensor	A powerful guard that intervenes only when a predetermined criteria is met, as if there were a trainer watching over the user's shoulder Sensitive to the context on the screen and to user behavior, sensors are used to push notifications and relevant information, as well as to validate user input and block user errors.
Wizard	An intelligent pre-developed script – created using Kryon Studio, stored on the Kryon RPA Server, and run by Kryon robots • In the context of Kryon's unattended automation solution, unattended robots automatically run wizards on target applications • In the context of Kryon's attended automation solution, human end-users run wizards on target applications using attended robots
Robot	A lightweight client that runs wizards on target applications • Unattended robot: installed on a virtual machine and runs wizards with no human intervention • Attended robot (known as **Kryon Assistant**): installed on an end-user desktop and runs wizards on the user's applications

CHAPTER 2: The Catalog

The catalog is a hierarchical tree made up of libraries, categories and wizards. These catalog levels, their properties, and catalog management procedures are described in the following sections:

Libraries 10
Applications 12
Categories 13
Wizard Properties 14
General Properties 17
- Name 18
- Description 18
- Status 18
- Keywords 19
- Hyperlink 20
- Run modes 23
- When ends 25
- Applications 25
- Related wizards 26
- External links 28

Embedded Wizards 30
Notes 31
Change History 32
Version History 33
Advanced Commands Count 34
Restoring the Catalog Home View 35

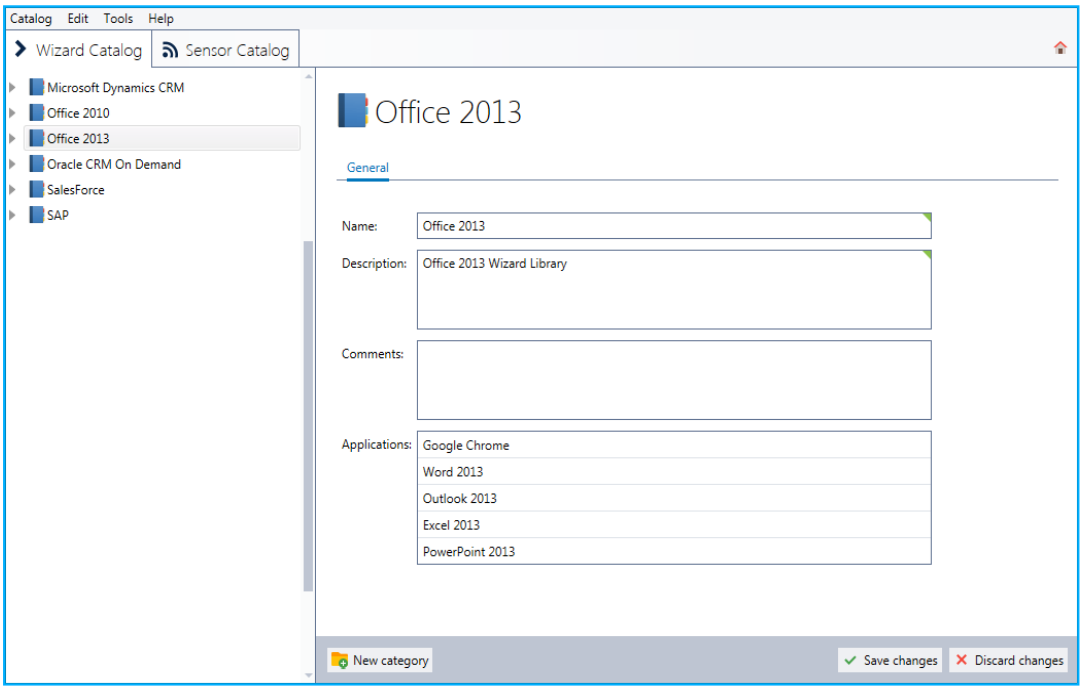
Libraries

Libraries are the top level in the catalog's hierarchy. Each library contains categories and subcategories, which contain automations (wizards and sensors).

NOTES

By default, the wizard and sensor catalogs share an identical library structure.

Libraries are not created in Kryon Studio. They are created in Kryon Admin, where they are associated with their supported applications.



All about libraries:

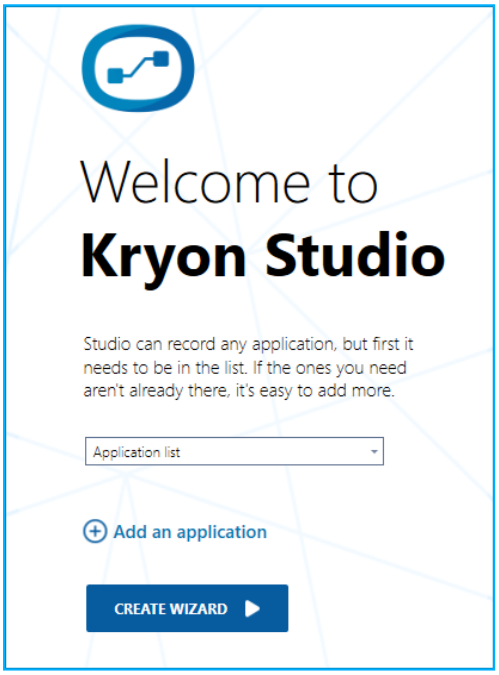
- Automations cannot be created directly under a library, but must be created in a category within that library
- Automations can be moved among categories within a library, but they cannot be moved to other libraries
- You can assign properties to each library (such as name and description) to provide information about the automations that it contains
- Library status:
  - In **active** libraries, all automations in **Published** status appear in the Kryon Robot search results and can be accessed by Kryon Robot users
  - In **inactive** libraries, automations can be edited by Kryon Studio users, but they do not appear in Kryon Robot search results (regardless of status), and they cannot be accessed by Kryon Robot users
- Each library is associated with predefined applications (desktop and/or web) for which automations can be recorded. The supported applications associated with a library are listed in the library's **APPLICATIONS** list.

Applications

A wizard can be recorded and run only on applications and websites that have been predefined for the library in which the wizard is located.

You can check the catalog (in a library's Applications list) to see which applications and websites are already defined. However, in the full version of the Kryon RPA Platform, applications are defined and associated with libraries in Kryon Admin (not in Studio), so contact your organization's Kryon RPA administrator ***before you begin recording*** if you don't see the applications you need.

If you are using the trial version of Kryon Studio, you can define supported applications from Studio's home screen.



NOTE

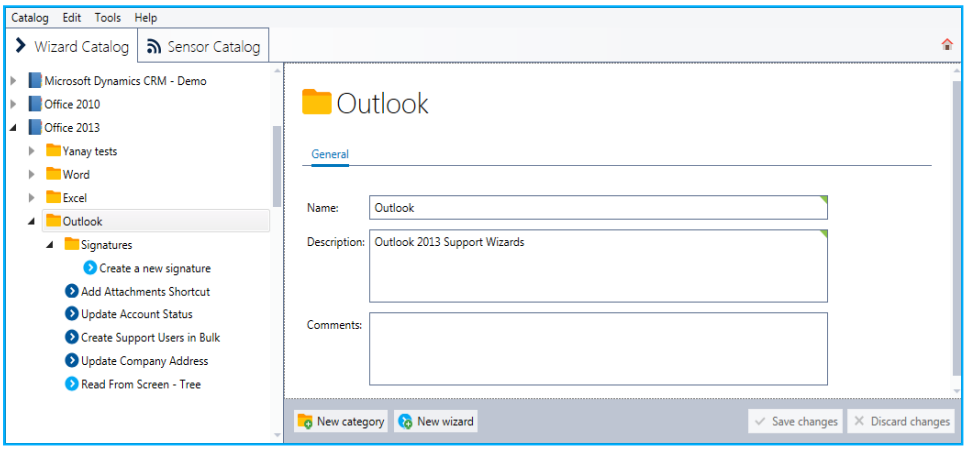
While the Kryon RPA Platform supports all applications, some capabilities may not be available for certain applications.

Categories

Categories and subcategories are subfolders created within a library in order to organize automations within that library. With categories and subcategories, you can build your catalog structure using what ever attributes are most useful for your company (e.g., by application, by department, etc.)

NOTE

Categories in the wizard and sensor catalogs are managed separately.



Creating a category

Follow these steps to create a new category or subcategory:

1. In the **Catalog** pane, select the library or category in which you want to create a new category (or subcategory)
2. Do one of the following:
   - From the menu bar, select **Catalog > New category**
   - Right-click the library or category and select **New Category**
3. In the **Properties** pane, in the **General** tab, do the following:
   - To change the category name, type a name in the **Name** field
   - (Optional) In the **Description** field, type a description that will be helpful in identifying the automations that the category contains
   - (Optional) In the **Comments** field, type any relevant comments you have about the category
4. Save your changes by clicking **Apply**

Wizard Properties

Each wizard in the catalog contains wizard properties and information about past changes and links to other wizards and sensors. This information is grouped into tabs in the **Properties** pane of the main Kryon Studio window. Each tab displays the following information when selected:

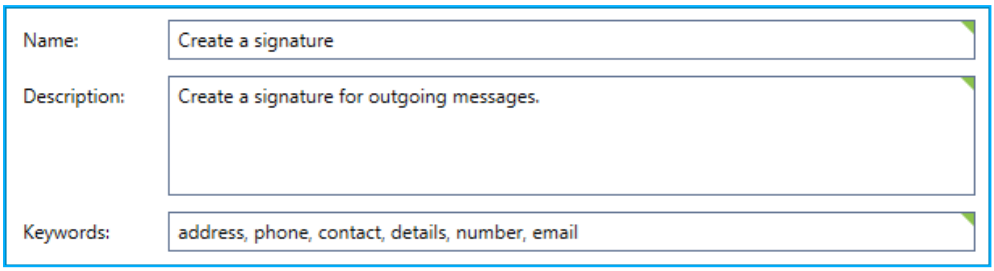
- General: The wizard's primary properties
- Embedded wizards: Information about wizards and sensors that are linked to the selected wizard
- Test cases
- Notes: A free-form text field in which you can type comments. You can use this tab to note the wizard status, testing details, required review, and more
- Change history: All changes to the wizard content and properties that were saved since the wizard was created
- Version history: Lists the previous versions of a wizard
- Advanced Commands Count: Provides a list of the Advanced Commands found in wizards and sensors

Items in the Studio catalog contain editable all-white and green-cornered meta-data fields. For published wizards, green-cornered fields are indexed in the Kryon Robot search engine and visible to end users in Kryon Studio catalog. Editable white fields are never visible to end users.

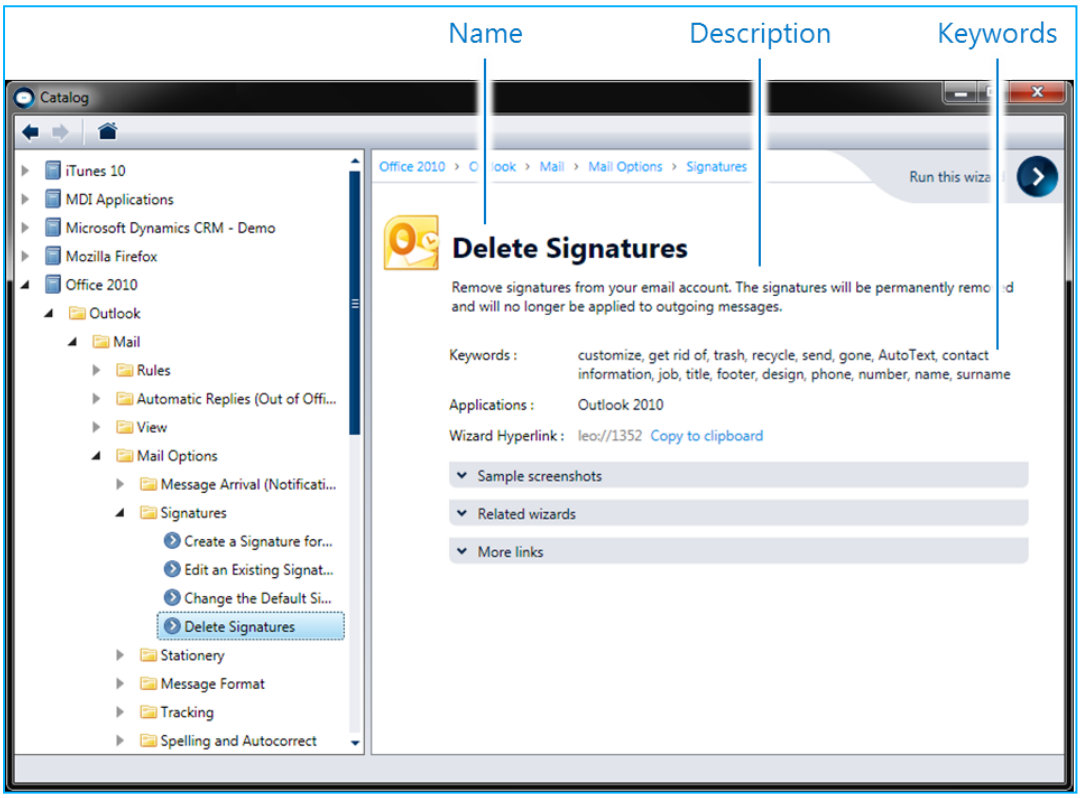
NOTE

This applies to wizards only. All editable sensor fields are invisible to end users.

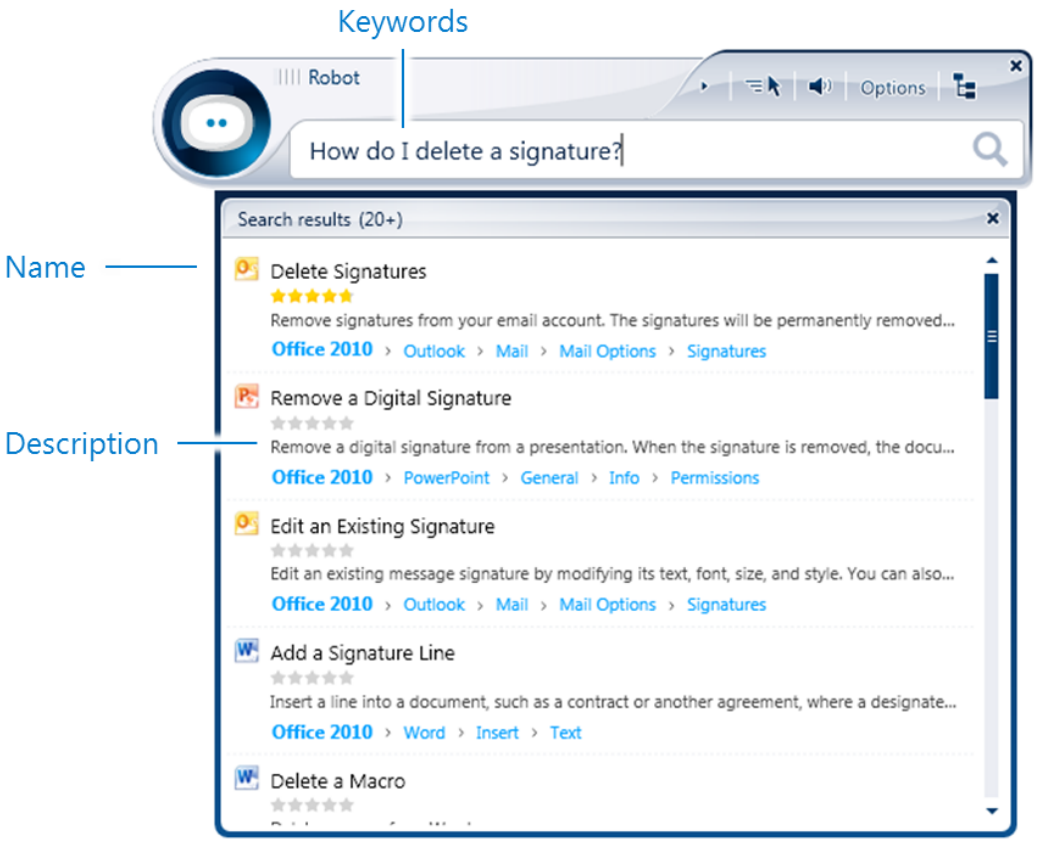
This is how the editable **Name**, **Description**, and **Keywords** fields appear in Kryon Studio:



This is how the **Name**, **Description**, and **Keywords** fields appear to users in the Kryon Robot catalog:



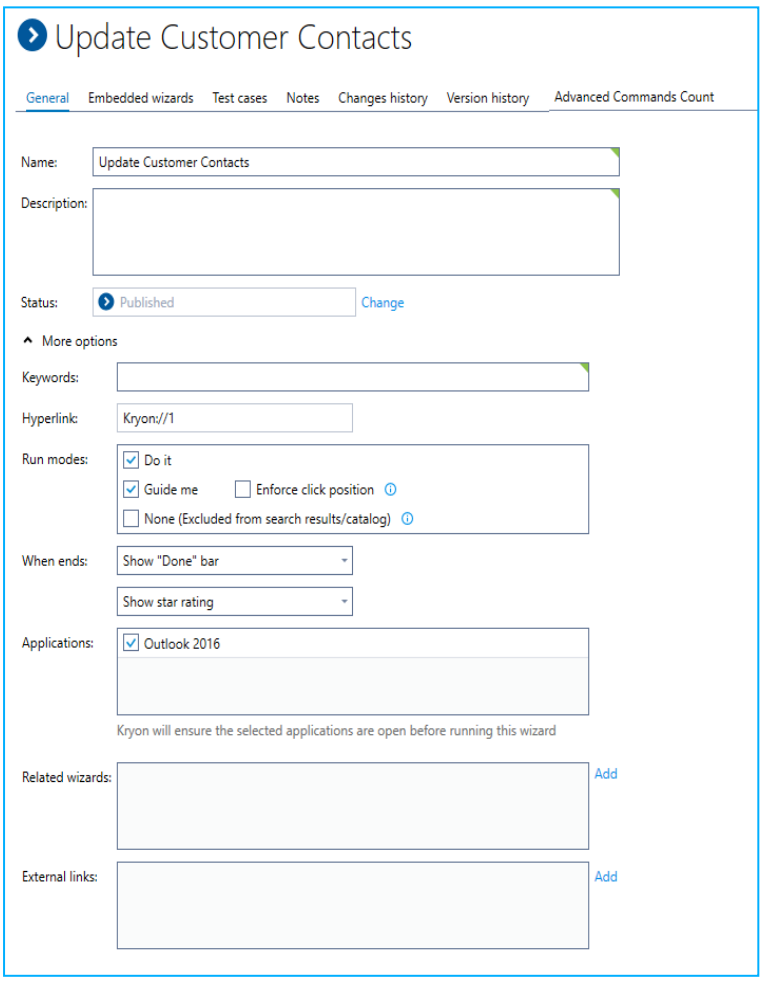
This is how the **Name**, **Description**, and **Keywords** fields appear to users in the Kryon Robot search results:



General Properties

The **GENERAL** tab contains the following properties:

- Name
- Description
- Status
- Keywords
- Hyperlink
- Run modes
- When ends
- Applications
- Related wizards
- External links



Name

A title containing the words searched by the Kryon Robot search engine. The name is visible:

- To end users in the Kryon Robot catalog and search results Attended/Hybrid Only
- In the Kryon Console catalog Unattended/Hybrid Only

Description

A description of the wizard. Words in the description are not used by the Kryon Robot search engine. The description is visible to end users in the Kryon Robot catalog and search results.

Status

The status of a wizard is a label that indicates the stage in the process of publishing the wizard. The status indicates if the wizard has been tested, approved, and is ready to be published.

Kryon Studio offers five status labels you can use to establish a publishing process to suit the work process in your organization. The available status labels are:

- **DRAFT**: The wizard is incomplete, and needs to be either recorded or further edited and tested. It does not appear in the Kryon Robot search results and cannot be accessed by Kryon Robot users. This is the default status of a new wizard.
- **PENDING APPROVAL**: The wizard is recorded and edited, and is pending approval before it can be published. It does not appear in the Kryon Robot search results and cannot be accessed by Kryon Robot users.
- **PUBLISHED**: The wizard is completed and approved. The wizard appears in the Kryon Robot search results and can be accessed by Kryon Robot users.

NOTE

Only wizards in **Published** status are visible to end users in Kryon Robot.

- **INACTIVE**: The wizard should not be published. It does not appear in the Kryon Robot search results and cannot be accessed by Kryon Robot users.
- **FAULTY**: The wizard does not work properly and needs to be fixed. It does not appear in the Kryon Robot search results and cannot be accessed by Kryon Robot users.

NOTE

When **Faulty** is selected, the **Faulty** description field is enabled.

Keywords

Comma-separated search words, used by the Kryon Robot search engine to bring up relevant results when users type a search query in the Kryon Robot search bar.

KEYWORD TIPS

- Enter keywords that are connected by association, meaning, and context to the wizard's purpose
- Include common spelling mistakes in the wizard keywords
- For keywords made up of two hyphenated words (e.g. “blue-green”), type each word as a separate keyword (e.g. “blue”, “green”)
- Do not type the following automatically generated keywords:
  - Wizard name
  - Application/library name
  - Past/present/future tense
  - Singular/plural
  - Conjugations

Hyperlink

A unique wizard link is automatically generated for each new wizard. When the link is clicked, it launches a wizard directly – without requiring the Kryon Robot search bar.

Wizard hyperlinks can be embedded wherever regular hyperlinks are allowed, such as email messages, web browsers, and the Windows **Run** command line.

CAUTION

For the hyperlink to work, the wizard must be published and Kryon Robot must be installed on the user's computer.

The wizard hyperlink is made up of a unique prefix, the wizard ID, and additional parameters that can be added manually. See Hyperlink components.

Hyperlink format

The wizard hyperlink format is as follows:

```
Kryon://<wizardID>/<runMode>/<stepNumber>/?<varname1>
=<value1>&<varname2>=<value2>
```

where –

- The wizard ID (mandatory), run mode (optional), and step number (optional) are preceded and followed by a forward slash (/)
- The first variable name (optional) is preceded by a question mark (?)
- The variable name and its value are delimited by an equals sign (=)
- Variables are delimited from other variables by a comma (,)

EXAMPLES

```
Kryon://123
Kryon://123/doit
Kryon://123/doit/3
Kryon://123/doit/3/?firstname=jane&lastname=doe
Kryon://123/doit/?firstname= jane&lastname=doe
Kryon://123/3/?firstname= jane&lastname=doe
Kryon://123/doit/3/?firstname=jane&lastname=doe
```

Hyperlink components

The following table describes each component of a wizard hyperlink:

Component	Description	Example	Comments
Prefix	The unique Kryon hyperlink prefix	`Kryon://`	Mandatory, generated automatically
Wizard ID	A number automatically generated when the wizard is created	`Kryon://123`	Mandatory, generated automatically
Run Mode	Determines whether the wizard will be run directly in **Do It** or **Guide Me** mode	`Kryon://123/doit` `Kryon://123/guideme`	Optional, added manually • The relevant wizard's **RUN MODE** must be enabled accordingly • Read more about controlling wizard run modes
Step Number	Starts the wizard from a specific step	`Kryon://123/doit/3`	Optional, added manually

Component	Description	Example	Comments
Variables & values	Variables that exist in the wizard but whose values are initialized manually by the person who writes or embeds the link	`Kryon://123/doit/3/?firstname=jane&lastname=doe`	Optional, added manually • If a variable name is added, its value must be provided • firstname and lastname are variable names that exist in the wizard • `jane` and `doe` are the values that will be inserted into the respective variables

Run modes

Indicates the modes in which the wizard can be run.

NOTE

Consider the context!

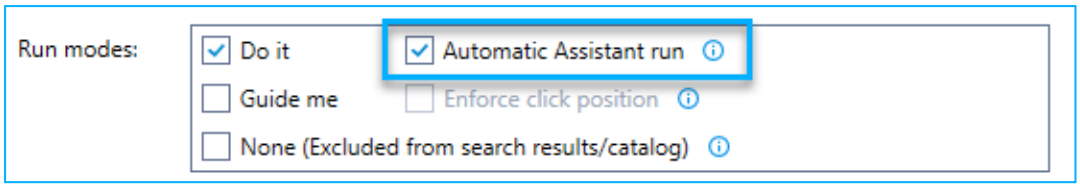
Unattended/Hybrid Only In an unattended automation context, be sure that **Do It** is selected as a run mode. This is the only mode supported by unattended robots.

Attended/Hybrid Only In an attended automation context, choose from any of the available run modes (according to the way the wizard is used within your organization).

The **RUN MODE** options are:

- **Do It:** The robot performs the actions for the user by moving the user's mouse and navigating the target application to complete the task
- **Guide Me**: The robot guides the user through the application by pointing to the exact location where the user needs to click or enter text for each step in the task. This mode includes the **ENFORCE CLICK POSITION** checkbox, which is useful for preventing unwanted clicks and mistakes. When this checkbox is selected, the following occurs when the wizard is run in **Guide Me** mode:
  - The entire screen around the highlighted click position is blocked and grayed out
  - If the user clicks outside the highlighted click position, Kryon Robot provides the user with the option of unblocking the screen
- **None**: All run modes are disabled and the wizard is excluded from the Kryon Robot search results
  - This option is used for embedded wizards
  - The embedded wizard will inherit available run modes from the wizard in which it is embedded (the "containing wizard")
- **Automatic Assistant Run**: Attended/Hybrid Only When you configure attended scenario run types (Do it or Guide Me) and set only one mode, select the **Automatic Assistant Run** check box to launch the wizard automatically without requiring additional

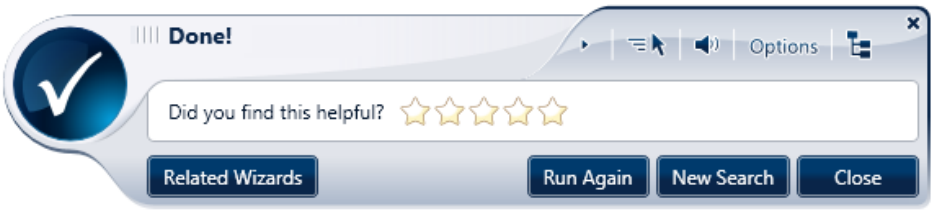
steps


Run modes:
Do it
Automatic Assistant run
Guide me
Enforce click position
None (Excluded from search results/catalog)

When ends Attended/Hybrid Only

Determines the appearance of certain features when Kryon Robot is done running a wizard and the **Done** bar appears. The feature options are:

- **DONE BAR DISPLAY:**
  - **Show "Done" bar:** Displays the **Done** bar at the end of the wizard run
  - **Hide "Done" bar:** Disables the **Done** bar at the end of the wizard run (provides a more unobtrusive user experience)
  - **Hide "Done" bar if launched by sensor:** Hides the **Done** bar if the wizard was launched by sensor.
- **STAR RATING DISPLAY:**
  - **Show star rating**
  - **Hide star rating**



Applications

Supported applications on which the wizard can run. When a wizard is recorded, Kryon Studio identifies the applications used and adds them to the wizard's **Applications** list.

Kryon Robot ensures that all selected applications are open before the wizard is run. Application checkboxes in the list are selected by default.

TIP

Control the applications that must be open when the wizard starts

Clear the application checkbox if the application will not be available when the wizard starts running.

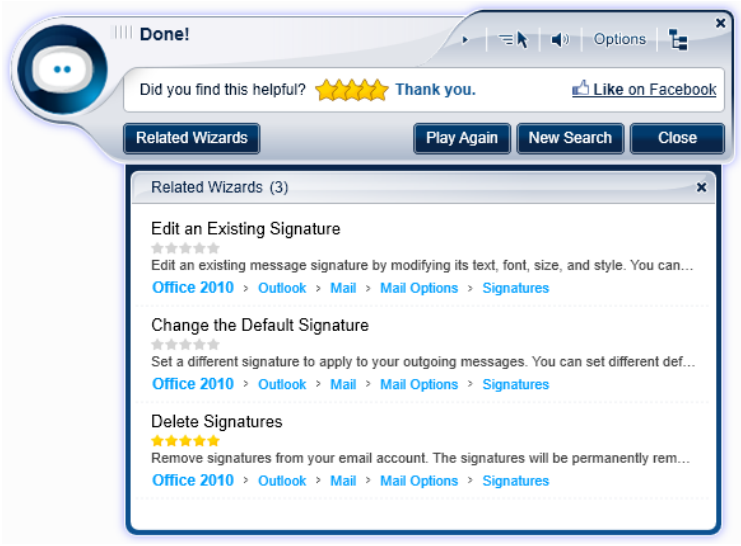
For example, when exporting Outlook contacts into a CSV file in Google Docs, the CSV file is only created in the middle of the process and cannot be launched when the process starts.

Related wizards Attended/Hybrid Only

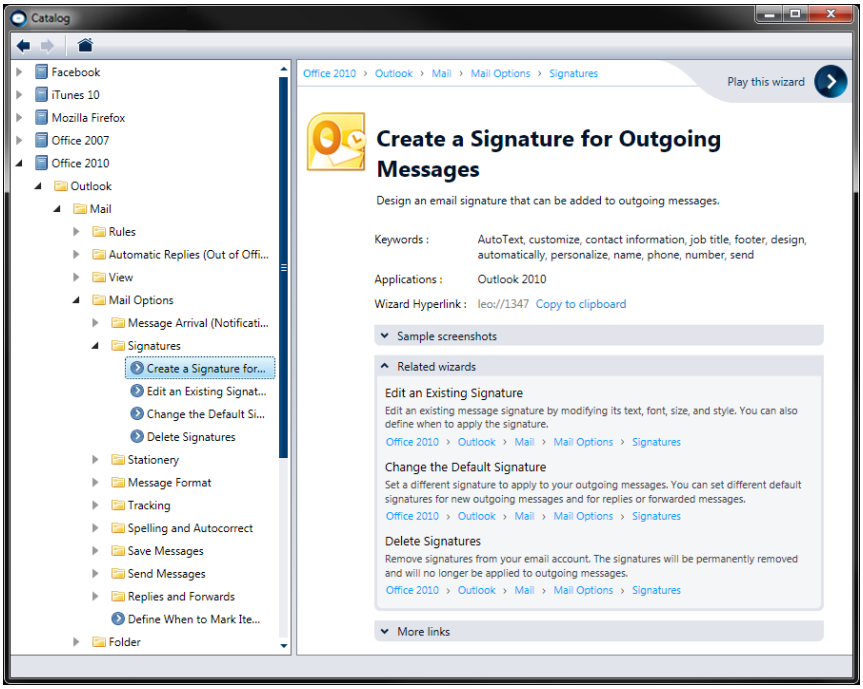
Wizards suggested to the user at the end of a wizard that are related to the task supported by the completed wizard.

- For example, for a wizard called *Create a Signature*, you might designate *Edit a Signature* and *Delete Signatures* as related wizards.

This is how related wizards appear in the Kryon Robot search bar after the user runs a wizard:



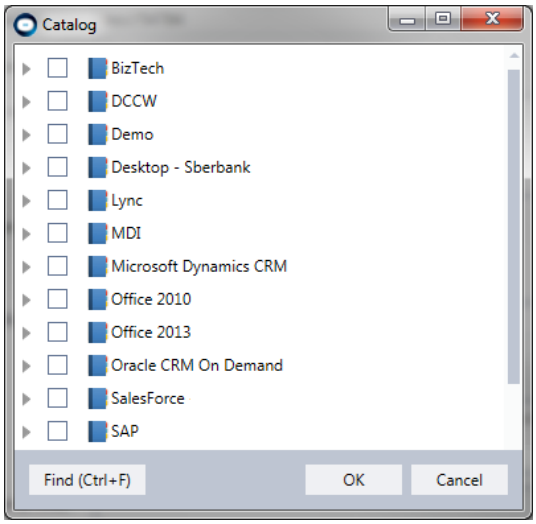
This is how related wizards appear in the Kryon Robot catalog:



Adding related wizards

Follow these steps to add related wizards:

1. In the **Catalog** pane, select the wizard to which you want to add related wizards
2. In the **General** tab, in the **Related wizards** field, click **Add**
3. The following dialog box appears:



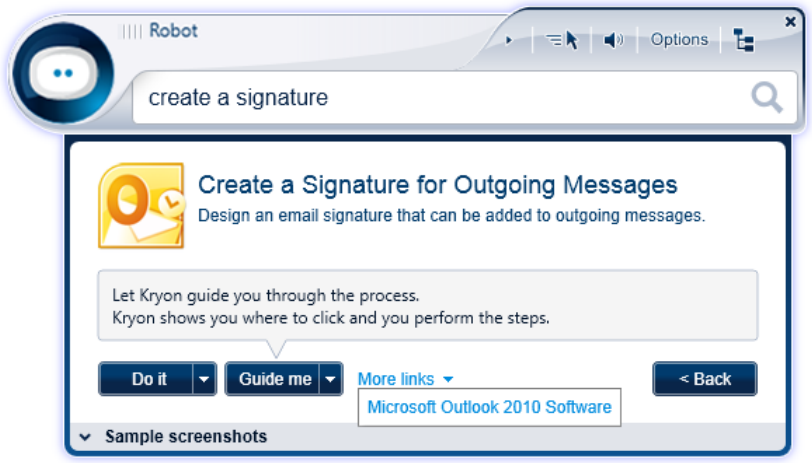
4. Tick the checkbox(es) for one or more related wizards to add
   - To search for a specific wizard, click **Find** or press **CTRL+F** and then type your search query
5. Once you have selected all the wizards to add, click **OK**
6. The wizards you selected are now related to the current wizard, and the current wizard is now related to each of the wizards you selected
7. Save your changes by clicking **Save Changes**. To cancel your changes, click **Discard Changes**.

External links Attended/Hybrid Only

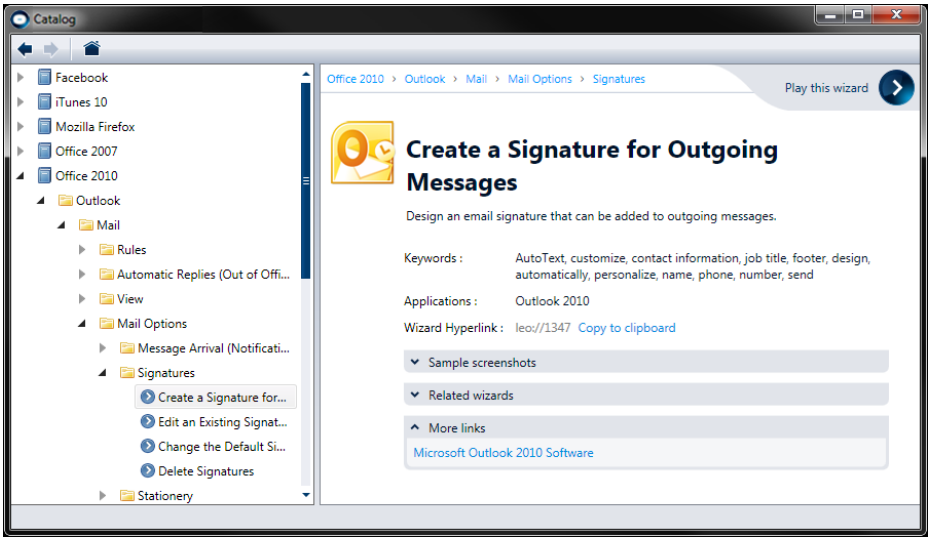
Hyperlinks that appear in Kryon Robot when the user selects the wizard. External links can direct the user to any linkable objects such as web pages or network directories.

The **EXTERNAL LINKS** field does not display links in the wizard flow itself. To learn about inserting links into the wizard flow, see Inserting a hyperlink into a bubble.

This is how external links appear in the **MORE LINKS** dropdown list in the Kryon Robot search bar, after the user selects a wizard.



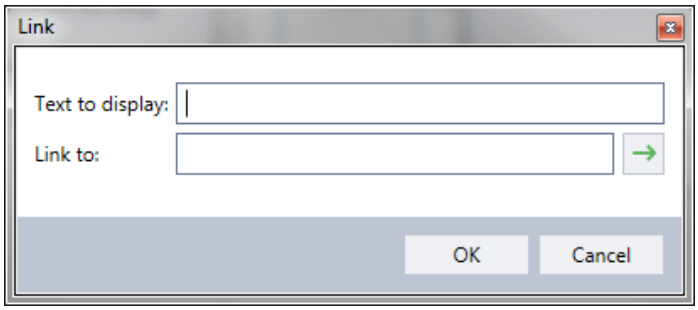
This is how external links appear in the **MORE LINKS** dropdown list in the Kryon Robot catalog, after the user selects a wizard.



Adding external links

Follow these steps to add external links to a wizard:

1. In the **Catalog** pane, select the wizard to which you want to add external links
2. In the **General** tab, in the **External links** field, click **Add**
3. The following dialog box appears:



4. In the **Text to display** field, type the text that you want to display for the link
5. In the **Link to** field, type or paste the link URL\
   - To test the link (i.e., go to the URL you entered), click the button
6. Click **OK**
7. The link is added to the wizard's external links
8. Save your changes by clicking **Save Changes**. To cancel your changes, click **Discard Changes**.

EXAMPLE

If you wanted to add an external link to the Microsoft Outlook support site, you would complete the fields as follows:

Text to display: `Microsoft Outlook Support`

Link to: `https://support.office.com/en-us/outlook`

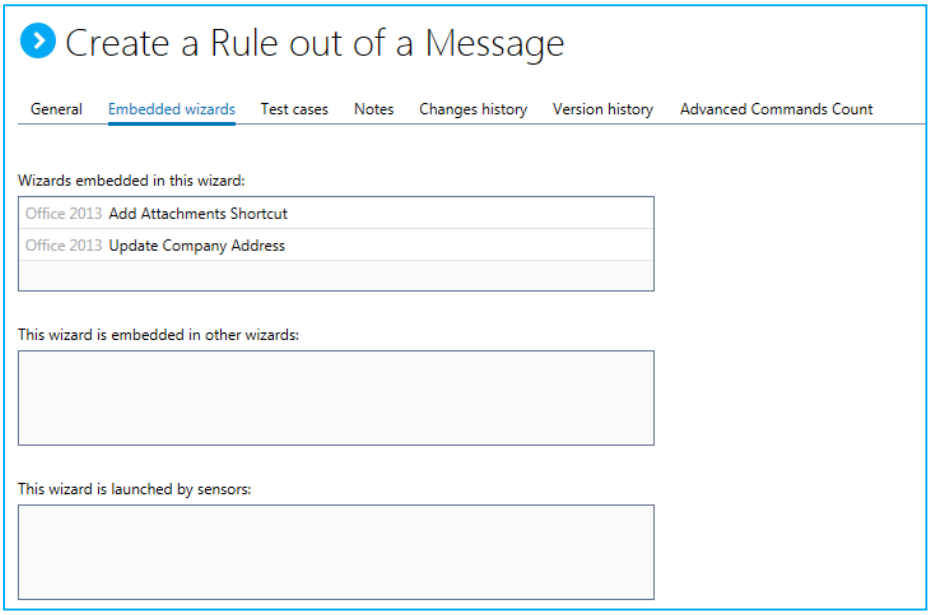
Embedded Wizards

Wizards don't have to be stand-alone entities. They can be used as building blocks within other wizards **(embedded wizards)**, and they can be launched by sensors.

The **EMBEDDED WIZARDS** tab contains the following information:

- Wizards embedded within the current wizard (if any);
- Wizards in which the current wizard is embedded (if any); and
- Sensors that launch the current wizard (if any)

These non-editable fields are invisible to end users in Kryon Robot.

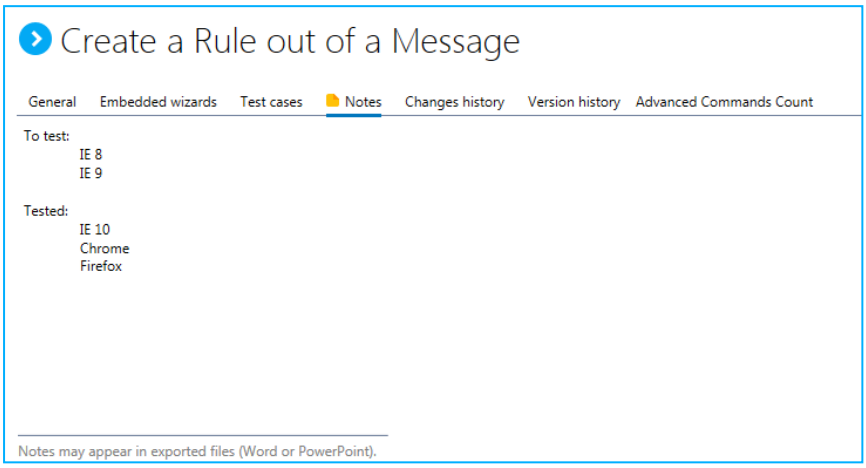


Learn more about embedding a wizard within another wizard.

Notes

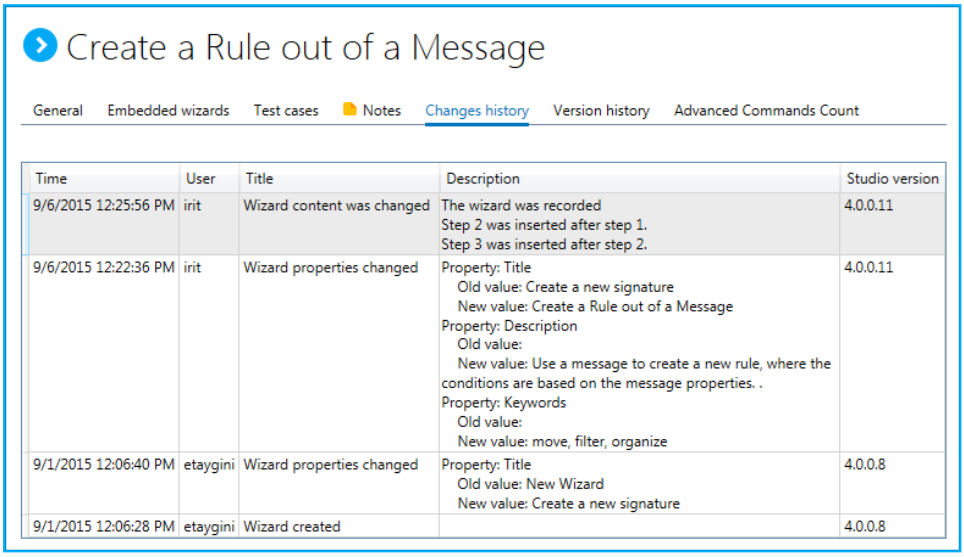
The **NOTES** tab can be used to type free text and add comments for the selected wizard.

The editable **NOTES** field is invisible to end users in Kryon Robot.



Change History

The **CHANGE HISTORY** tab contains a table listing all changes made to the selected wizard's content and properties since it was created:



Time	User	Title	Description	Studio version
9/6/2015 12:25:56 PM	irit	Wizard content was changed	The wizard was recorded Step 2 was inserted after step 1. Step 3 was inserted after step 2.	4.0.0.11
9/6/2015 12:22:36 PM	irit	Wizard properties changed	Property: Title Old value: Create a new signature New value: Create a Rule out of a Message Property: Description Old value: New value: Use a message to create a new rule, where the conditions are based on the message properties. . Property: Keywords Old value: New value: move, filter, organize	4.0.0.11
9/1/2015 12:06:40 PM	etaygini	Wizard properties changed	Property: Title Old value: New Wizard New value: Create a new signature	4.0.0.8
9/1/2015 12:06:28 PM	etaygini	Wizard created		4.0.0.8

The table lists any of the following types of changes:

- **Wizard creation:** Creation of a new wizard by the user
- **Wizard content:** Any changes to the wizard content such as steps and bubbles, or if the wizard was imported from a local drive to the database
- **Wizard properties:** Any changes to wizard properties such as the name, description, and keywords
- **Wizard migration:** When a new version of Kryon Studio is released, all wizards are migrated to the new version. This migration is logged in a wizard's **CHANGE HISTORY** as a change performed by the Kryon Migration Tool.
- **Wizard category:** Change performed when the wizard is moved from one category to another in the catalog

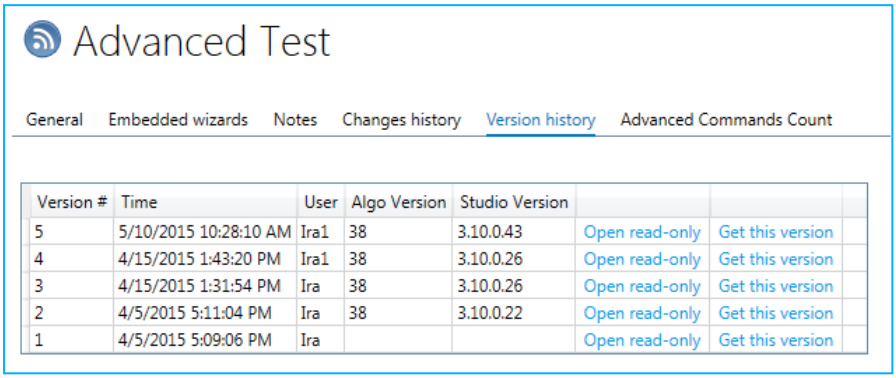
For each change, the following parameters are listed:

- **Time:** The date and time that the change was performed
- **User:** The name of the user who performed the change
- **Title:** The type of change
- **Description:** The content of the change
- **Studio version:** The version of Kryon Studio in which the change was performed

This non-editable table is invisible to end users in Kryon Robot.

Version History

The **VERSION HISTORY** tab contains a table that lists the previous versions of a wizard. By default, the platform saves and displays the last 10 versions. The maximum number of versions to save can be configured in Kryon Admin:



Version #	Time	User	Algo Version	Studio Version		
5	5/10/2015 10:28:10 AM	Ira1	38	3.10.0.43	Open read-only	Get this version
4	4/15/2015 1:43:20 PM	Ira1	38	3.10.0.26	Open read-only	Get this version
3	4/15/2015 1:31:54 PM	Ira	38	3.10.0.26	Open read-only	Get this version
2	4/5/2015 5:11:04 PM	Ira	38	3.10.0.22	Open read-only	Get this version
1	4/5/2015 5:09:06 PM	Ira			Open read-only	Get this version

For each saved version, the following parameters are listed:

- **Version #:** The wizard version number
- **Time:** The date and time the version was saved to the database
- **User:** The Kryon username of the user that saved the version to the database
- **Algo Version:** The Kryon algorithm version number
- **Studio Version:** The Kryon Studio version number

VERSION HISTORY enables you to:

- Open a read-only copy of a previous version by clicking the **OPEN READ-ONLY** link:

Version #	Time	User	Algo Version	Studio Version		
5	5/10/2015 10:28:10 AM	Ira1	38	3.10.0.43	Open read-only	Get this version

- Open a previous version for editing by clicking the **GET THIS VERSION** link:

Version #	Time	User	Algo Version	Studio Version		
5	5/10/2015 10:28:10 AM	Ira1	38	3.10.0.43	Open read-only	Get this version

Opening a previous version does not automatically overwrite the current version. Overwrite only occurs if you **save** the previous version. When you do so, the following occurs:

- The version you are saving (formerly the previous version) overwrites the current version and becomes the current version; and
- The version being overwritten (formerly the current version) is saved in the version history as a previous version

Advanced Commands Count

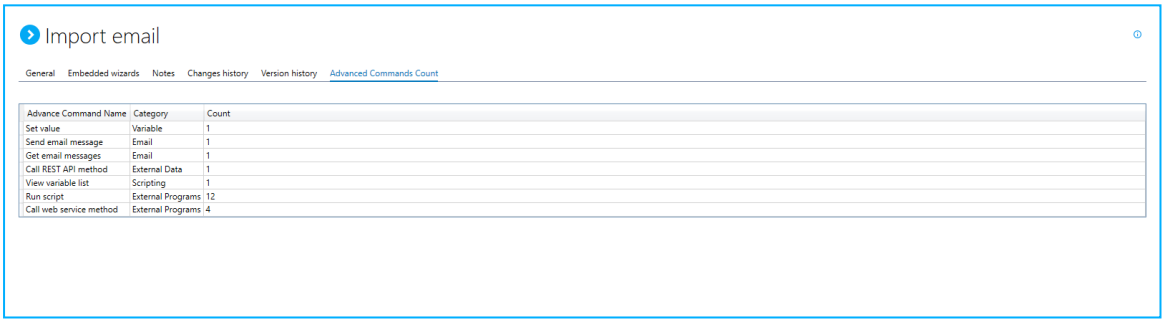
The **ADVANCED COMMANDS COUNT** tab provides a list of the Advanced Commands found in wizards and sensors.

Advanced Commands in embedded wizards are not included in the list.

TIP

No data? That can mean that either:

1. There aren't any Advanced Commands in the wizard. Feel free to add some
2. The wizard needs refreshing. If so, click **Edit Wizard** and then save it to see the data

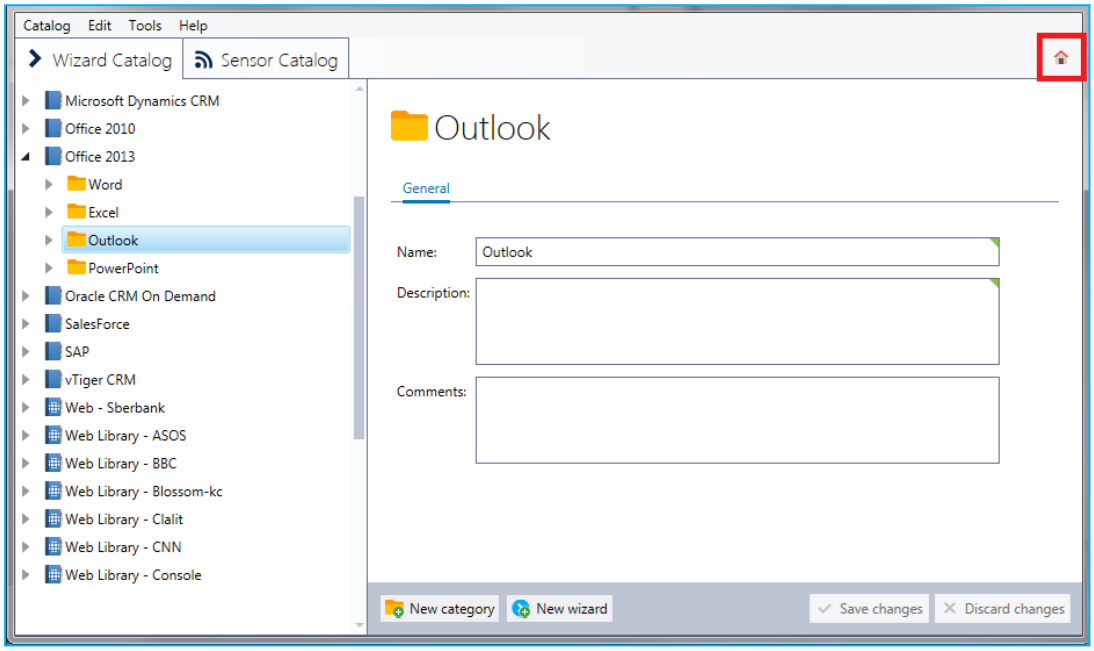


Restoring the Catalog Home View

When you launch Kryon Studio, the default **Home** catalog view appears. When you select a catalog item from the catalog, the selected catalog item's properties are displayed in the **Properties** pane.

To restore the Catalog **Home** view:

1. Click the **Home** button in the upper-right corner of the main Studio window:



2. The Catalog **Home** view is restored.

WIZARD DEVELOPMENT

CHAPTER 3: Creating & Recording Wizards

Wizards are made up of the steps and functionality required to perform a process on one or more applications.

Kryon's patented technology is based on image and optical character recognition, enabling it to "see" the screen just like the user does. Since a robot relies on screen captures to see the user's screen, wizards are first recorded on the target application(s), and then edited for functionality so that the robot can correctly execute the process.

When you record a wizard, Kryon Recorder captures all the mouse clicks and key strokes performed on the application. It then processes each recorded mouse click or key stroke as a single step in the Wizard Editor, which you can later edit and configure.

In this chapter:

Creating a Wizard 38
Preparing to Record 40
Recording a Wizard 42

Creating a Wizard

Creating a new wizard consists of the following steps:

- Adding a new wizard to the catalog: Wizards are created in categories in Kryon Studio and can then be assigned properties and recorded
- Editing general wizard properties: Properties such as the wizard name and description are assigned in the Properties Pane
- Opening the wizard: Wizards are recorded and edited in the Wizard Editor

Adding a new wizard to the catalog

Follow these steps to add a new wizard to the catalog:

1. In the Catalog, select the category in which to create the wizard
2. Do one of the following:
   - From the menu bar, select **Catalog > New wizard**
   - Right-click the category and select **New Wizard**
3. A new wizard is added to the catalog

Editing general wizard properties

Follow these steps to edit the wizard's general properties:

1. In the Catalog, select the wizard whose properties you want to edit
2. In the General tab, type the name you'd like to give the wizard
   - Users who access wizards from Kryon Robot can only see one line in the **Name** field. Make sure your title fits in the designated space.
3. Type the wizard description and keywords in the relevant fields
4. Click **Save Changes** to save the wizard's general properties. To cancel your changes, click **Discard**.

Opening the wizard

Wizards are opened, recorded and edited in the Wizard Editor. To access the wizard editor:

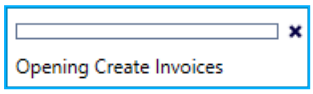
1. In the Catalog, select the wizard that you want to record or edit
2. Do one of the following:
   - At the bottom of the Properties Pane, click the **Get Started** or **Edit Wizard** button
     - The button will read **Get Started** if the wizard is brand new (with no steps)
     - The button will read **Edit Wizard** if the wizard already includes some steps

- Press **CTRL+E**
- From the menu bar, select **Edit > Open editor**

3. Right-click the wizard and select **Open editor**

The Wizard Editor window opens, and you are ready to record or edit your wizard.

- To cancel the wizard opening, click the ✖ on the progress bar in the bottom-left corner of the screen.



NOTE

If the wizard is currently open on another user's machine, the wizard is checked out and locked for editing.

TIP

You can view and/or work on several wizards at the same time by opening multiple **Wizard Editor** windows from the main Kryon Studio window.

Preparing to Record

Complete these two important preliminary steps before recording to ensure the process goes smoothly:

Define applications & websites

Set up your environment for recording

Define applications & websites

Confirm that the applications and websites on which you will record are defined ***for the library in which the wizard is located.*** To learn more about defining applications and websites, see Applications.

Set up your environment for recording

Since the wizard will eventually run on a robot – not on your computer – set up your environment to match the robot's as closely as possible, including:

- Windows version
- Windows display settings (for example, screen resolution and font size)
- Application versions
- Application color schemes or skins
- Application user settings (user permissions and settings that could affect application appearance/behavior)
- Browser type and version (if you're recording web applications)

NOTE

For a list of supported operating systems and browser versions, see the ***Kryon RPA Platform System Architecture & Requirements*** document.

TIP

Match as closely as possible, but don't worry if it's not perfect!

No worries if you can’t get all of these environment settings perfectly aligned. During the editing process, you can make changes to the wizard to account for any differences. But getting the environment to match as closely as possible now will save you editing and testing time later.

If your wizards will run on a number of different machines and/or with different users (as is common in an attended automation context), try to match the environment that matches the majority of users.

Recording a Wizard

Wizards are recorded by performing the desired process on your own application(s), while **Kryon Recorder** is running in the background, capturing every click and keystroke.

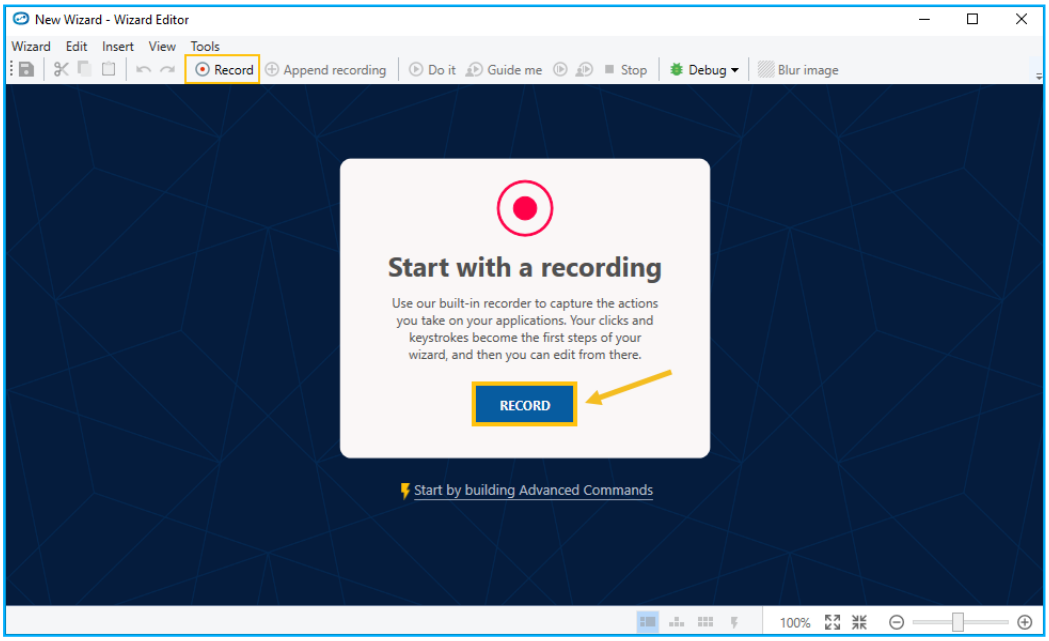
What you need

- A computer matching as closely as possible the robot's (or most common end-user) environment
- The application(s) on which you will record, launched on your computer and set to the first step of the process
- A new wizard added to the catalog

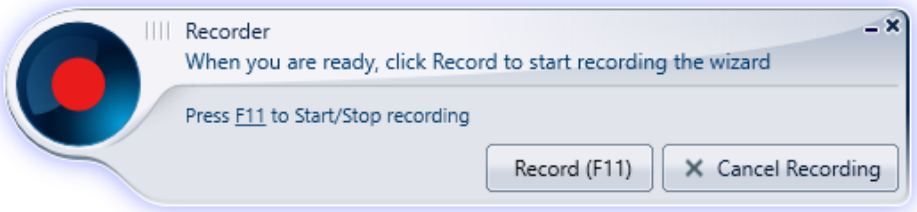
Start recording

To record a wizard:

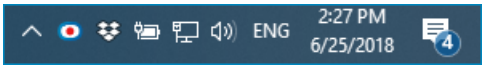
1. Open the wizard in **Wizard Editor**
2. Do one of the following:
   - On the toolbar, click **Record**
   - Click the **Record** button at the center of the Display Pane
     - **NOTE:** This option is only available when you record a new wizard
   - On the menu bar, select **Wizard > Record**



The **Wizard Editor** window is minimized, and the **Recorder** bar appears:



3. Go to the application window in which you want to start recording
4. In the Recorder bar, click the **Record** Record (F11) button
   - A 3-second countdown begins
   - When the countdown ends, the **Recorder** bar disappears and the **Recorder** icon appears on your taskbar, indicating that the recording is in progress



> **TIP**
>
> While the **Recorder** bar is still visible, you can click **Cancel Recording** to cancel the recording and return to the **Wizard Editor**.

5. Perform the process you want to record in your application(s), while **Recorder** is running in the background

> **NOTE**
>
> **Be sure to click!**
>
> Kryon supports a variety of actions (called "core actions") to allow your robots to accurately recreate the interactions of human users. These include Click, Keyboard, Hover, Detect object, and Read from screen. But when recording, **Recorder** captures only mouse clicks and keystrokes. So if you want to execute a core action on a specific object other than Click or Keyboard, when you record the wizard you should click the object instead. When you do so, **Recorder** captures the step and knows which object you want it to interact with. Then, after the wizard is recorded, you can change the core action as necessary to the one you need.

Scrolling & dragging

Any steps containing scrolling or drag-and-drop actions are automatically disabled in the generated wizard.

To learn more about using scrolling in wizard steps, see Scroll to position.

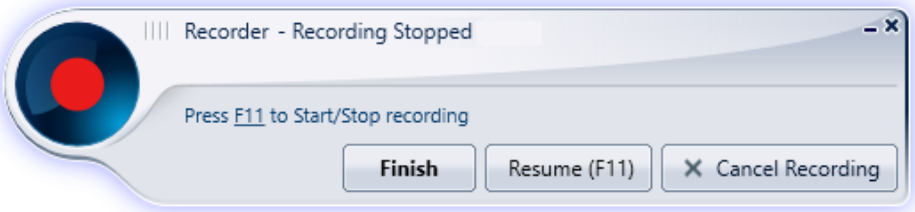
Drag-and-drop actions are not supported as recorded mouse action but can be executed by using the **Drag & Drop** advanced command, as described in the ***Advanced Commands Reference Guide (Drag & Drop)*** available from the Advanced Command View toolbar.

Pause recording

Pause a recording by doing one of the following:

- From the taskbar, click the **Recorder** icon
- Press `F11`

The recording is stopped and the **Recorder** bar appears.



When you are ready to continue recording, click the **Resume** button Resume (F11).

TIP

To change the hotkey used to **Start/Stop recording** (by default, `F11`), click the `F11` link in the **Recorder** bar and select a different hotkey from the dropdown list that appears.

Cancel a recording

To cancel the recording:

1. From the taskbar, click the **Recorder** icon

   The **Recorder** bar appears

2. Click the **Cancel Recording** button

to cancel the recording and return to the **Wizard Editor**

The recording is canceled and the **Wizard Editor** window appears in its original state.

End a recording

To end the recording:

1. From the taskbar, click the **Recorder** icon

   The **Recorder** bar appears

2. Click the **Finish** button

to end the recording

The recording is ended and the wizard (with the steps you recorded) appears in the Wizard Editor, ready for editing.

CHAPTER 4: The Wizard Editor

The Wizard Editor is used for viewing, creating and maintaining wizards in Kryon Studio. It enables you to record and edit wizards and to update previously recorded and/or edited wizards.

In this chapter:

A Tour of the Wizard Editor 47

Wizard Views 49

The Navigation Pane 53

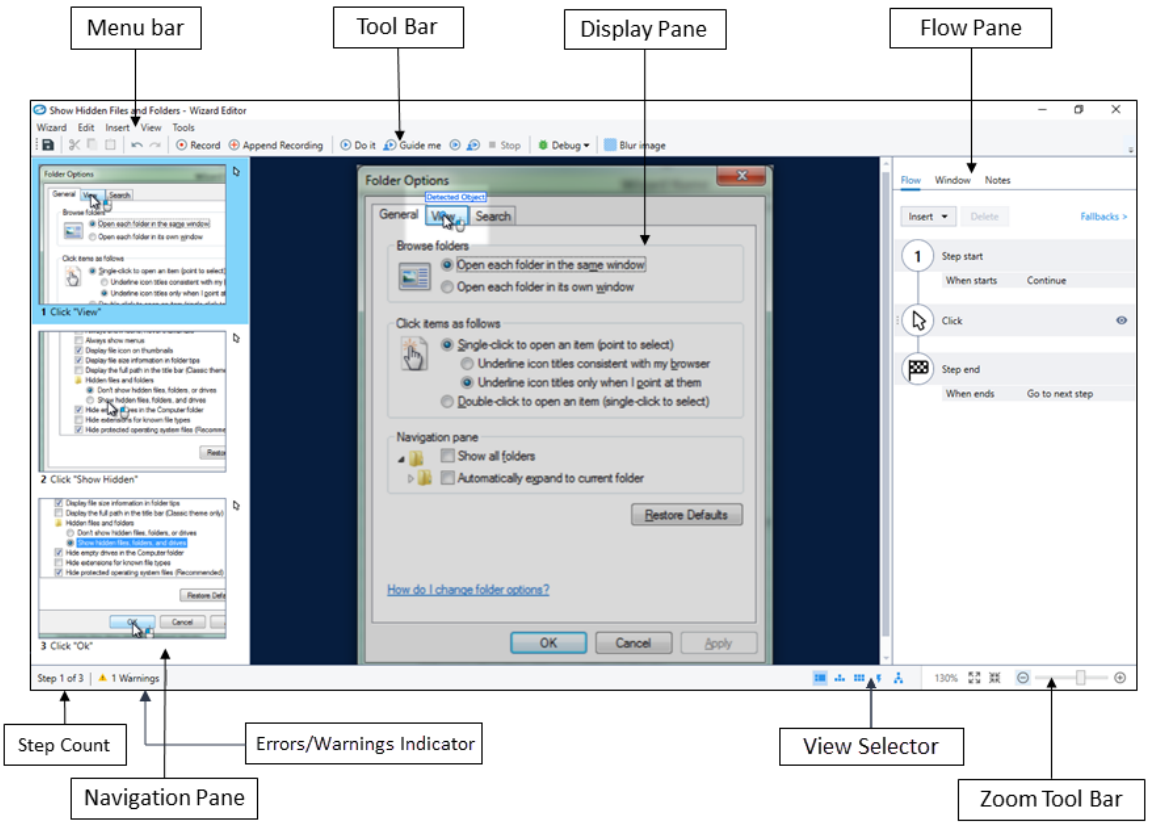
The Flow Pane 55

The Properties Pane 61

A Tour of the Wizard Editor

For information about how to access the Wizard Editor, see Creating a Wizard.

In Normal View, the Wizard Editor includes the following features and work areas:



- **Menu bar:** collection of menus that provide access to functions that affect either the entire wizard or the selected step. Some of these functions are also available from other locations in the Wizard Editor.
- **Toolbar:** collection of functions that affect either the entire wizard or the selected step
- **Display Pane:** selected step's main display, including layers of functionality
- **Flow Pane:** collection of tabs that provide access to the functions and options for the selected step
- **Step count:** selected step's number and total number of steps
- **Navigation Pane:** sequence of steps in which each step is represented by a thumbnail of the recorded image and action, with indicators for the layers of functionality added to it
- **Errors/warnings indicator**: label indicating whether there are errors or warnings in the wizard
- **View selector:** allows you to change among available Wizard Views (Normal, Diagram,

Storyboard, Advanced Commands, Workflow)

- **Zoom toolbar:** buttons that allow you to zoom in or out of the selected step display

Wizard Editor Options

You can modify the default Wizard Editor options in order to set recording settings, wizard language, default bubble setting, and more. To change these options:

1. On the menu bar, click **Tools > Options**
2. In the **Options** window , select the **Playback** or **Editor** tab, and set the relevant options
3. Click **OK** to save your options

Wizard Views

Kryon Editor offers several options for viewing and editing the flow of a wizard:

- Normal view
- Diagram view
- Storyboard view
- Advanced commands view
- Workflow view

Switch among these views using the View selector:

Normal view

Normal view is the default view in which wizards are opened in the Wizard Editor. In this view:

- Wizard steps appear as a linear string of thumbnails in the Navigation Pane
- The selected step's image appears in the Display Pane
- Functions and options for the selected step appear in the Flow Pane

Normal view options

Zooming & panning

The Zoom toolbar enables you to magnify or shrink the step view in the Display Pane:

TIP

You can also zoom in or out by pressing **CTRL** while rolling the mouse wheel up or down.

When the recorded window is larger than the Display Pane (and you choose not to shrink it by zooming out), you can move around the pane either by:

- using the vertical/horizontal scrollbars; ***or***
- clicking and dragging the image in the Display Pane (i.e., panning)

Disabling/enabling spotlight display

By default, the Display Pane highlights the selected step's detected objects (spotlight display).

To disable this spotlight feature:

1. On the menu bar, click **View**
2. Click **Show spotlight**

Detected objects are no longer highlighted in the Navigation Pane.

Diagram view

Diagram view displays the wizard flow in a diagram/flowchart structure. It shows where the wizard starts, ends, where fallbacks are applied, what kind of action takes the user to the next step, and more.

- Zooming & panning are available in Diagram view

Diagram view displays the following for each step:

- Thumbnail – including click position, indicator icons, step number, and step name
- Connectors – leading to and from each step and the steps it is connected to

Each action and function in the wizard is represented by a shape or connector of specific colors:

Shape + Caption	Function
	Wizard starting point
	When wizard ends successfully
	When wizard ends unsuccessfully
	When step starts
	When step ends successfully
+ fallback type	Fallback
+ button caption	Bubble button
	Block

TIP

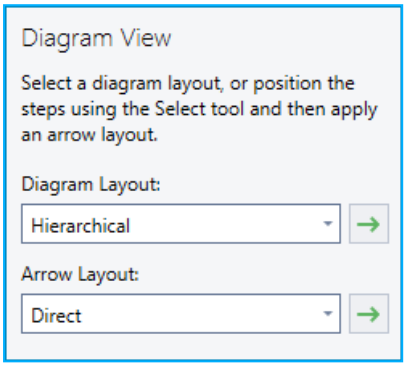
A legend of the shapes/connectors and their functions is also displayed in the Diagram view itself.

Diagram view options

Various view options are available within Diagram view:

- **Diagram Layout:** determines the layout of steps; the default view is **Hierarchical**
- **Arrow Layout:** Determines the layout of connectors; the default view is **Direct**

Change these options in the left-hand pane of Diagram view:



NOTE

These options only affect the view of the diagram. They do not affect wizard flow or functionality.

Storyboard view

Storyboard view displays the wizard as a sequence of thumbnails, but differs from Diagram view in that it does not display fallbacks and connectors. It allows you to focus on the sequence of individual steps and the action performed in each.

- Zooming & panning are available in Storyboard view

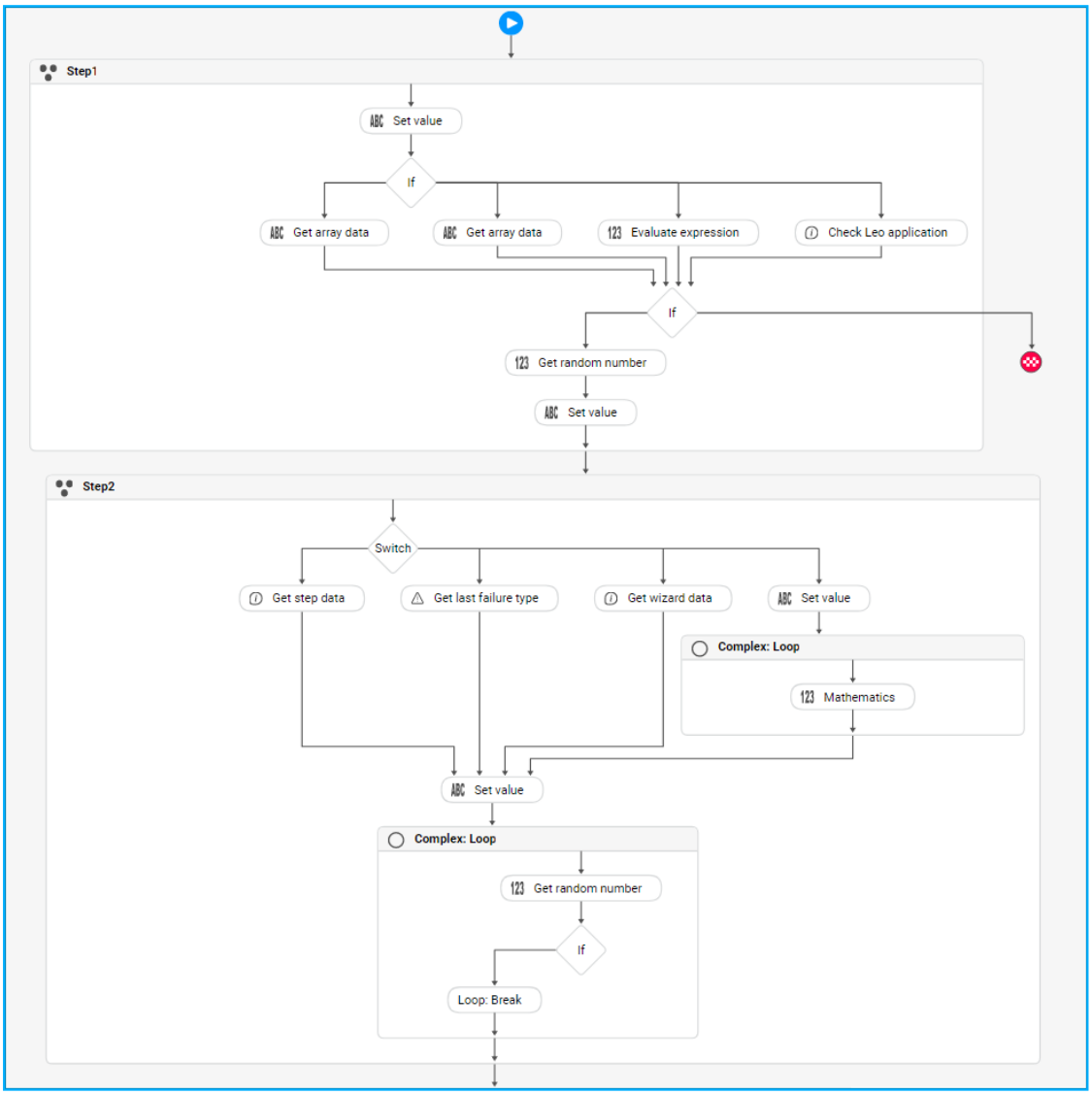
Advanced commands view

Advanced Commands view opens the Advanced Commands Editor. It displays the flow of each step, and enables you add and edit advanced commands for each step action.

Workflow view

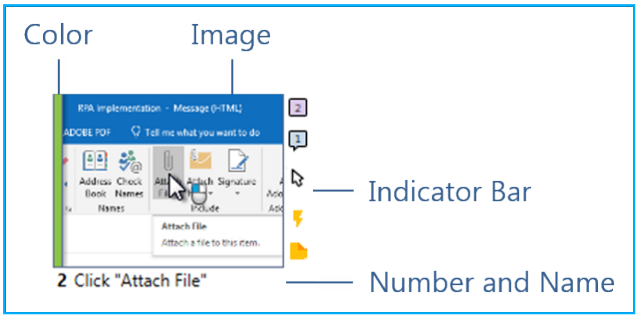
Workflow view displays the overall flow of your wizard along with all the details – including the actions taken in each step, logical operations (such as **Loop** and **If/Else**), **Fallbacks**, etc.

- Zooming & panning are available in Workflow view



The Navigation Pane

In the Navigation Pane, each step is represented by a thumbnail, comprised of the following elements:



- **Step number –** sequential step number assigned by Kryon Editor
- **Step name –** assigned by Kryon Editor when possible; editable by the RPA developer
- **Step image –** thumbnail of step, indicating detected object and action taken
- **Step color –** assigned by the RPA developer for organization/ease of identification (see Organizing Steps by Color)
- **Indicator bar –** indicating whether the step contains any of the following:
  - Core action type – **core action** icon (varies based on the type of action performed in the step)
  - Blocks – **blocks** icon (appears if the step contains blocks, including the number of bubbles)
  - Bubbles – **bubbles** icon (appears if the step contains bubbles, including the number of bubbles)
  - Advanced commands – **advanced commands** icon (appears if the step includes advanced commands)
  - Notes – **notes** icon (appears if the step includes notes)

When you select a step in the Navigation Pane, the full step image appears in the Display Pane.

Navigating to a specific step

While you can click steps to select or navigate them, you can also navigate to the **previous step**, **next step**, **first step**, or **last step** from the menu:

1. On the menu bar, click **Edit**
2. Click **Go to** and select the step to which to navigate

The relevant step is selected in the Navigation Pane and the full step image appears in the Display Pane.

Displaying disabled step thumbnails

By default, when a step has been disabled (i.e., marked **Do not play**), the step's thumbnail is hidden in the Navigation Pane – with only the step name and number visible

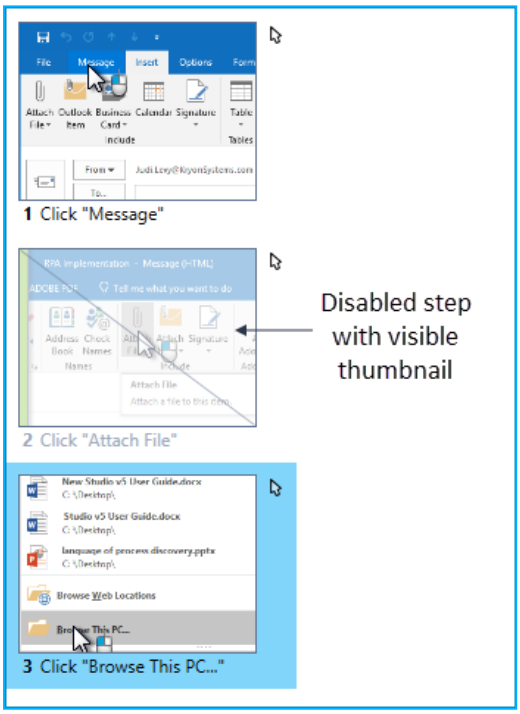
TIP

You can still see the contents of a disabled step in the Display Pane and Flow Pane by clicking on the step name/number

To view the thumbnails of disabled steps:

1. On the menu bar, click **View**
2. Click **Show disabled steps**

Disabled steps are expanded in the Navigation Pane to display their thumbnails:



The Flow Pane

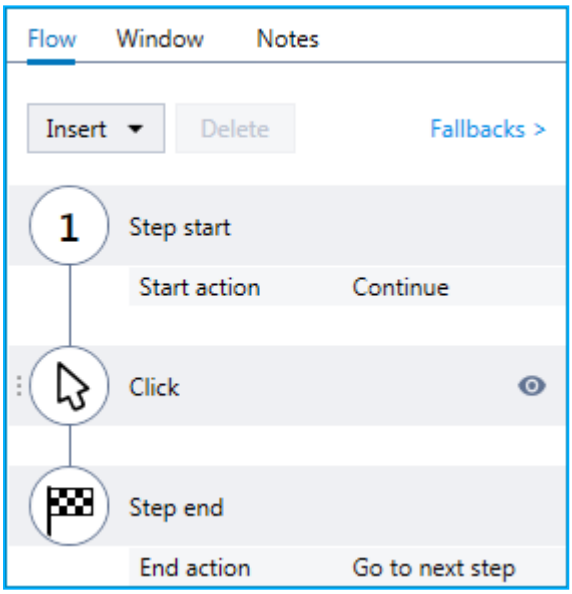
The **Flow Pane** contains the sequence of actions, layers of functionality, and additional properties of the selected step. It contains the following tabs:

- **Flow tab:** the primary Flow Pane display; shows the flow of actions in the step
- **Window tab:** contains all the window properties available for the recorded step
- **Notes tab:** enables RPA developers to add internal notes and comments to a step

Flow tab

The **Flow** tab displays the flow of actions within the selected step, and enables you add layers of functionality relating to these actions. When you record a step, the Flow tab automatically displays the three default step actions:

- Step start
- Core action; ***and***
- Step end



In addition to displaying the step flow, the **Flow tab** enables you to:

- Insert step actions
- Change action types
- Delete actions
- Change action events
- View action fallbacks
- Access the Properties Pane

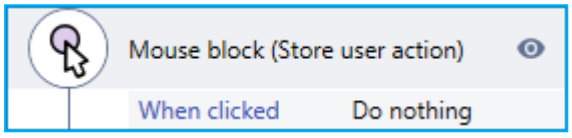
Step actions

Each wizard step is made up of a series of actions that occur in a sequence. These actions are at the heart of the wizard step and enable you to add functionality to the step.

Some actions are built-in by default, while others can be inserted by the RPA developer according to the process logic. The settings of all actions can be customized to fit the process needs.

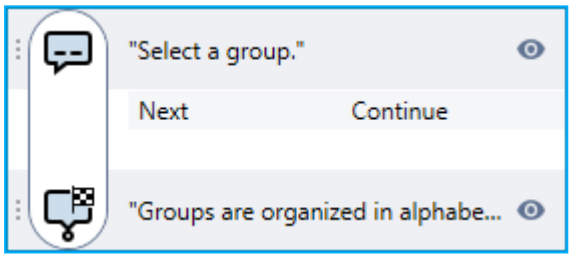
Actions are listed in the **Flow tab** in the order in which they occur, and can start either:

- one at a time (single action)



– or –

- at the same time as other actions (grouped actions)



NOTE

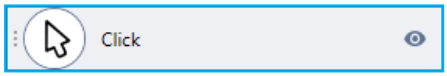
Grouped actions start at the same time, but do not necessarily end at the same time. These show/hide times can be determined in the action's Properties Pane.

Read more about the different types of step actions.

By default, all actions are set per step. Some actions can be set to work across more than one step (see Cross-step actions).

Every action in the step is represented in the step's **Flow tab**, and contains the following elements:

- **Visual indicator & label:** An icon and text label, representing the type of action (for the Step start action, the icon displays the step number)
- **Properties Pane:** Contains customizable properties for the action (hidden by default – to access the action's Properties Pane, either double-click the action or right-click it and select **Properties**)
- **Show/hide icon:** Click the action's icon to hide an action that prevents you from accessing actions beneath it in the Display Pane
- **Delete:** For removable actions, click the Delete button or right-click the action and select **Delete**
- **Drag Handle:** Actions that can be dragged and dropped onto a different position in the flow are indicated by a drag handle

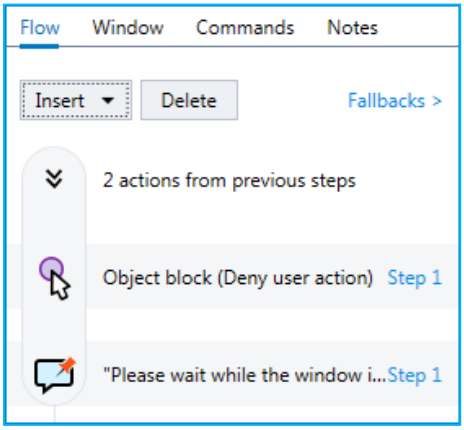


Cross-step actions

Cross-step actions are actions set to end at the end of a subsequent step in the wizard. In this way, these actions function across more than one step. The actions that can be used as cross-step actions are:

- Blocks
- Bubbles

Cross-step actions are listed at the top of the Flow tab and are collapsed by default. They can be edited by clicking the link to the step number in which the action began.



Fallbacks display

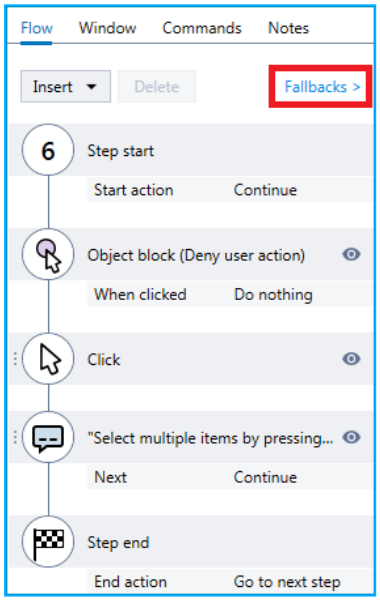
The **Fallbacks display** provides a quick view of all fallbacks for each action in the step's **Flow** tab.

> **NOTE**
>
> The **Step End** action does not have fallbacks.

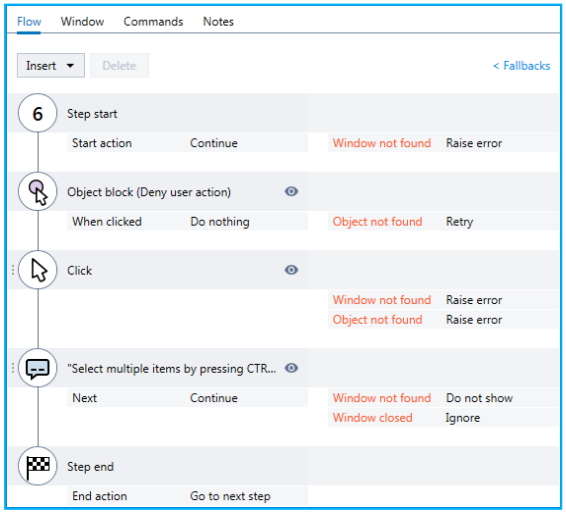
Read more about Fallbacks.

To access the **Fallbacks display**:

1. From the Flow tab, click **Fallbacks**

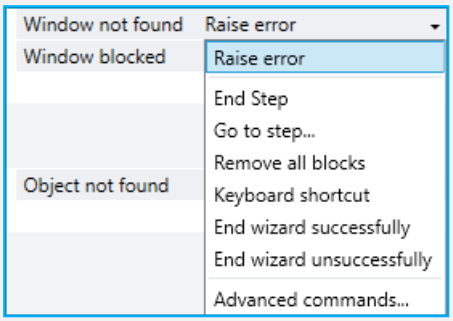


The Fallbacks display appears, listing the fallbacks for each action.



TIP

You can update the fallback command by clicking on the fallback dropdown list:



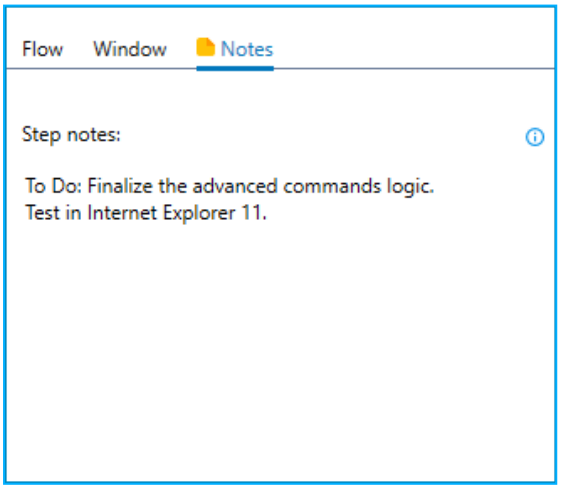
Window tab

Window detection and various additional window options are managed from the Flow Pane's **Window** tab.

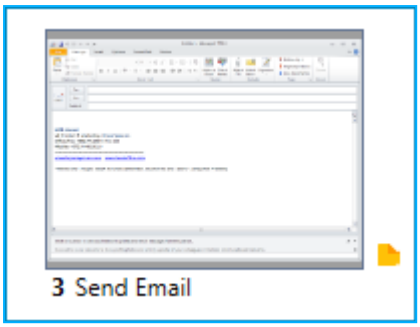
Notes tab

RPA developers can type internal notes and comments in a step by using the **Notes** tab. Notes are for internal reference only and will not appear when the wizard is played in Kryon Robot. Notes can only be viewed in the Wizard Editor or when the wizard is exported to a Word or PowerPoint file whose template contains the **Notes** field.

This is how the **Notes** tab appears in the Wizard Editor's Flow Pane (when notes are added to a step, the **Note icon** appears in the **Notes** tab's heading):



When a step contains notes, the Notes icon also appears in the step's thumbnail in the Navigation Pane:


3 Send Email

The Properties Pane

The **Properties Pane** displays all the properties and settings that can be customized for different aspects of the selected action or the window. The tabs in the **Properties Pane** vary depending on the selected action or window.

Accessing the Properties Pane

The **Properties Pane** is hidden by default, but can be easily accessed from the Flow Pane.

For a step action

To access the **Properties Pane** for a **step action**, do one of the following from the Flow tab:

- Double-click the action;
- Right-click the action and select **Properties**; ***or***
- Select the action, and from the menu bar, select **View > Property panel**

For a step's detected window

To access the **Properties Pane** for the step's detected window, do one of the following from the Window tab:

- Double-click **Window data**; ***or***
- Right-click **Window data** and select **Properties**

Available properties

The following sections describe the properties available for step actions and windows.

Properties tab

All actions have a **Properties** tab in the **Properties Pane**, which contains the main actions and settings available for that action. The properties vary depending on the selected action.

The **Properties** tab is indicated by the **Properties** icon .

Position tab

Some actions require detection of an object on the screen. Object detection is managed in the action's **Position** tab in the **Properties Pane**.

The **Position** tab is indicated by the **Position** icon .

Fallbacks tab

The **Fallbacks** tab is indicated by the **Fallbacks** icon .

See the Fallbacks section of this guide for detailed information about fallback events and defining fallback commands.

CHAPTER 5: Window Detection & Window Options

When a wizard is run, it uses data automatically collected during recording to determine the relevant window on which to act.

The following sections explain: (1) the various window properties automatically captured during recording; (2) the numerous options Kryon Studio offers for editing and adding to this data to ensure optimal results at wizard runtime; and (3) the various additional options for managing detected window behavior:

Window Detection 64

Window Properties for Desktop Applications 66

Window Properties for Web Applications 68

Editing Window Properties 69

Adding a Window Detection Rule 71

Managing Window Detection Rules 73

Adding a Set of Window Detection Data 76

Adding a Visual Object 78

Window Options 80

Window Detection

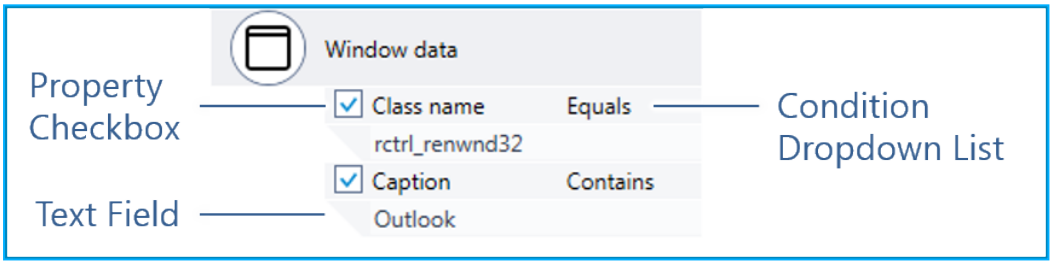
When you record a wizard, Studio stores a set of properties about each window on which an action was taken. **Window detection** is the name of the process that occurs when the wizard uses this information at runtime to determine the correct window on which to act. The data used for windows detection can be viewed and accessed for editing from the Window tab of the Flow Pane. The window properties collected for each step differ based on the type of application recorded:

- Window properties for desktop applications
- Window properties for web applications

For information about how to edit window detection properties, see Editing Window Properties.

Window properties

Each window property contains the following components:



- **Property checkbox:** Determines whether this property will be included or ignored during window detection
- **Condition dropdown list:** Options that determine which part of the text field will be included or ignored during window detection. The **condition dropdown list** contains the following options:
  - **Equals:** The property of the detected window must equal the exact text string appearing in the text field. **This is the default value.**
  - **Contains:** The property of the detected window must contain the exact text string appearing in the text field, either on its own or as part of a longer string (at the beginning, middle, or end). **This becomes the default value after you make a change to the property's text field**.
  - **Begins with:** The property of the detected window must begin with the exact text string appearing in the text field, either on its own or followed by other text
  - **Ends with:** The property of the detected window must begin with the exact text string appearing in the text field, either on its own or preceded by other text
  - **Use wildcards:** The property of the detected window must equal the exact text string appearing in the text field; however, the following characters can be used in the text string to replace any character or string of characters:

    - An asterisk (*) represents one or more characters (or no character)
    - A question mark (?) represents any single character
  - **Use regular expression:** The property of the detected window must match the pattern represented by the regular expression appearing in the text field
- **Text Field:** Contains the text string that the robot will look for in order to detect the window property, qualified by one of the conditions from the **condition dropdown list**

NOTE

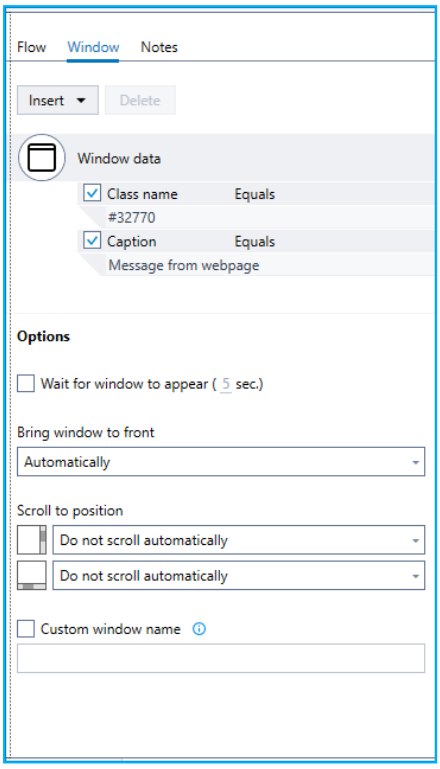
What is a *regular expression*?

Often, when you search for data in a text, you are looking for all the text that matches a certain pattern, rather than for specific text itself. A ***regular expression*** (often abbreviated ***regex***) is a sequence of characters that represents the pattern you are searching for.

- To learn more about regular expressions, see this article on the Microsoft Developer Network.
- For an online regex tester and reference, check out this website: regular expressions 101.

Window Properties for Desktop Applications

This is how Window tab of the Flow Pane appears for desktop applications:



Each set of window data collected for an action taken on a desktop application contains the following editable properties:

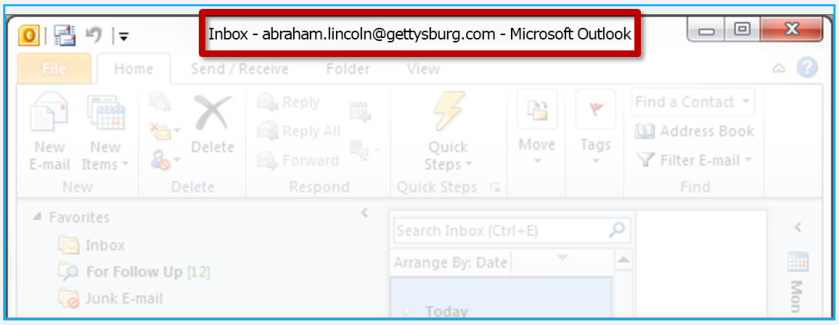
- **Class name:** A set of attributes defined by the application developer and used to generate a window on the user's screen. Every window is a member of a certain window class.
  - The window class name is captured automatically in the step's window properties:
    - the **Class name** checkbox is selected
    - the dropdown list is set to the **Equals** option; ***and***
    - the text field is filled in with the class of the window in which the step was performed

> **CAUTION**
>
> For some windows, the window class differs when you open that window in different operating systems or web browsers.
>
> For some applications, the window class changes every time you launch that window.

- **Caption:** The text that appears in the window title bar
  - The window caption is captured automatically in the step's window properties:
    - the **Caption** checkbox is selected;
    - the dropdown list is set to the **Equals** option; ***and***
    - the text field is filled in with the full caption of the window in which the step was performed

CAUTION

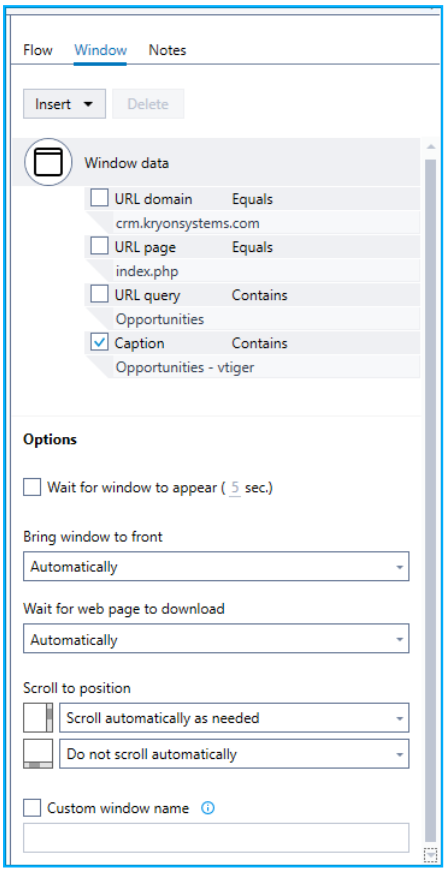
Some captions are user-specific:



For information about how to edit window detection properties, see Editing Window Properties.

Window Properties for Web Applications

This is how the Window tab of the Flow Pane appears for web applications:



Each set of window data collected for an action taken on a web-based application contains the following editable properties:

- **URL domain:** The domain name of a website, that is, a unique string that serves as a web address and identifies the owner of that address (e.g. docs.google.com).
  - By default, the URL domain checkbox is selected and the text field is filled in with the website domain name.
- **URL page:** The name of a page on the website.
  - By default, the URL page checkbox is selected and the text field is filled in with the web page name, if available.
- **URL query:** A string containing data transferred from the browser to the program that generates the web page.
  - By default, the URL query checkbox is selected and the text field is filled in with the query string, if available.
- **Caption:** The text that appears in the window title bar. For more information, see Caption.

For information about how to edit window detection properties, see Editing Window Properties.

Editing Window Properties

For information about the components of window properties, see Window properties.

To edit the selected step's window properties:

1. Go to the Window tab of the Flow Pane
2. Tick the checkboxes of properties you want the wizard to use to detect the window (i.e, active properties). Untick any checkboxes you do not want the wizard to use.
3. For each active property, select the desired condition from the condition dropdown list

TIP

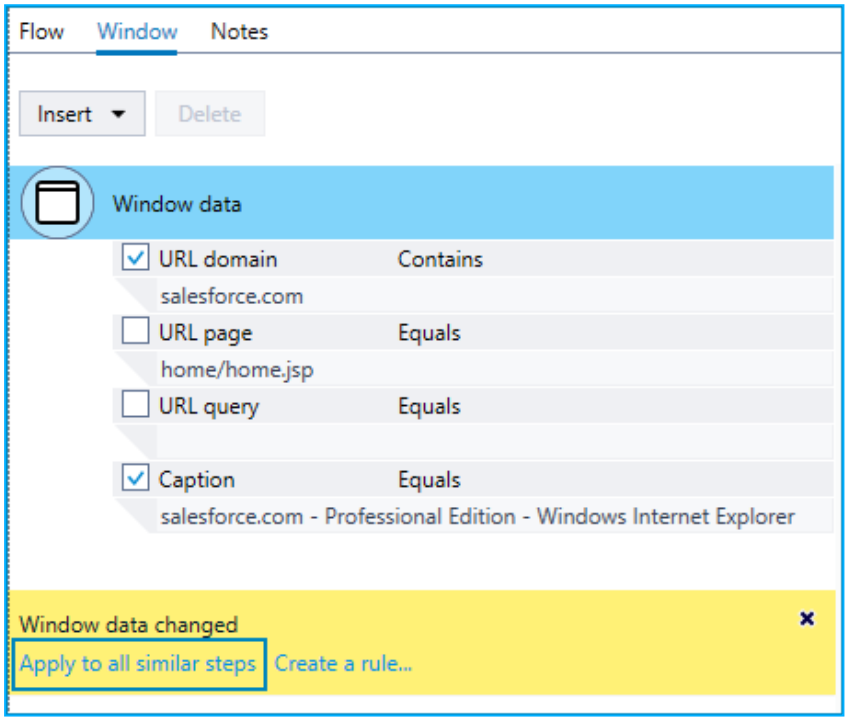
The **Contains** option is recommended for ensuring a generic setting that applies to all end users. Other options can be used for ensuring a more specific setting.

4. In each active property's text field, make sure that the text entered is as generic as possible and does not contain user-specific or environment-specific text that might fail the detection under different circumstances

NOTE

The text field is not case sensitive

5. When you change the data in the text field, the following occurs:
   - The value in the condition dropdown list changes from **Equals** to **Contains**
   - If this property is being changed for the first time, the **Window data changed** dialog appears, with one or more of the following options:
     - **Apply to all similar steps:** This option will appear if the window in this step is shared by other steps in the current wizard. Click the link if you want to apply the change you made to this property to all steps in the current wizard that share this window.



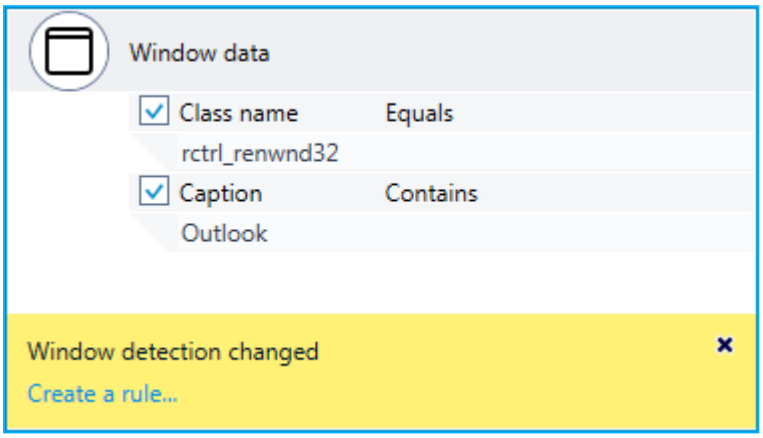
     - **Create a rule:** If you want to create a rule that applies the property to **all wizards** that share similar window detection values, follow the steps described in Adding a Window Detection Rule.

       For information about how to manage existing rules, see Managing Window Detection Rules.

Adding a Window Detection Rule

Window detection rules enable you to apply the same property conditions and text to all new wizards that share the same window detection properties (for example, a window caption in Microsoft Outlook that contains a specific username).

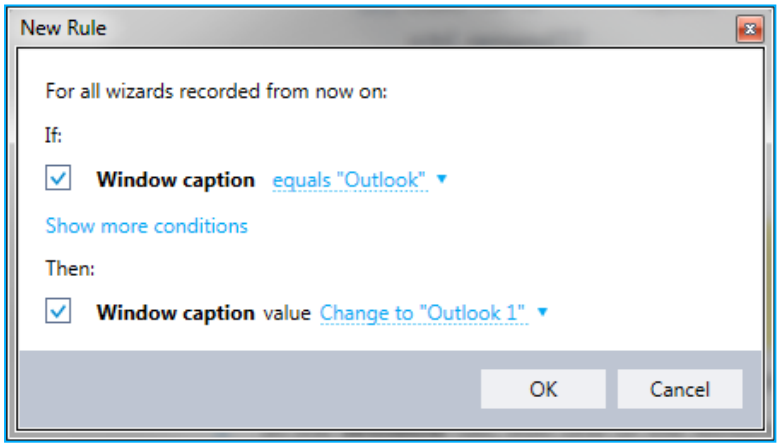
Rules are added from within a wizard you are currently working on in the Wizard Editor. The option to add a rule becomes available the first time you manually change a window property.



To add a window detection rule:

1. Click the **Create a rule** link that appears the first time you change a window property

   The **New Rule** dialog box appears



2. In the **New** Rule dialog box, select or clear the **If** checkboxes to add or remove rule conditions
3. Click a rule condition's dropdown list to select an operator, which qualifies the rule condition

EXAMPLE

If the recorded window's caption is `Inbox - abraham.lincoln@gettysburg.com - Microsoft Outlook`, you might select the `ends with Microsoft Outlook` operator to make the condition applicable to a broader range of scenarios.

4. Click the **Show more conditions** link to add rule conditions
5. Select or clear the **Then** checkboxes to add or remove rule actions
6. Click a rule action's dropdown list to select an operator, which qualifies the rule action
7. Click **OK** to apply the rule

The rule is added to the Kryon Studio Rule Manager and will now be applied to every new wizard recorded.

CAUTION

Window detection rules are applied only to new **wizards recorded after the rule was created**:

- Rules are not applied to wizards existing at the time the rule was created
- Rules are not applied to the wizard you are working on at the time the rule is created
  - For information about applying window detection properties to similar steps within the wizard you are currently working on, see Apply to all similar steps

For more information about viewing and managing rules in the Kryon Studio Rule Manager, see Managing Window Detection Rules.

Managing Window Detection Rules

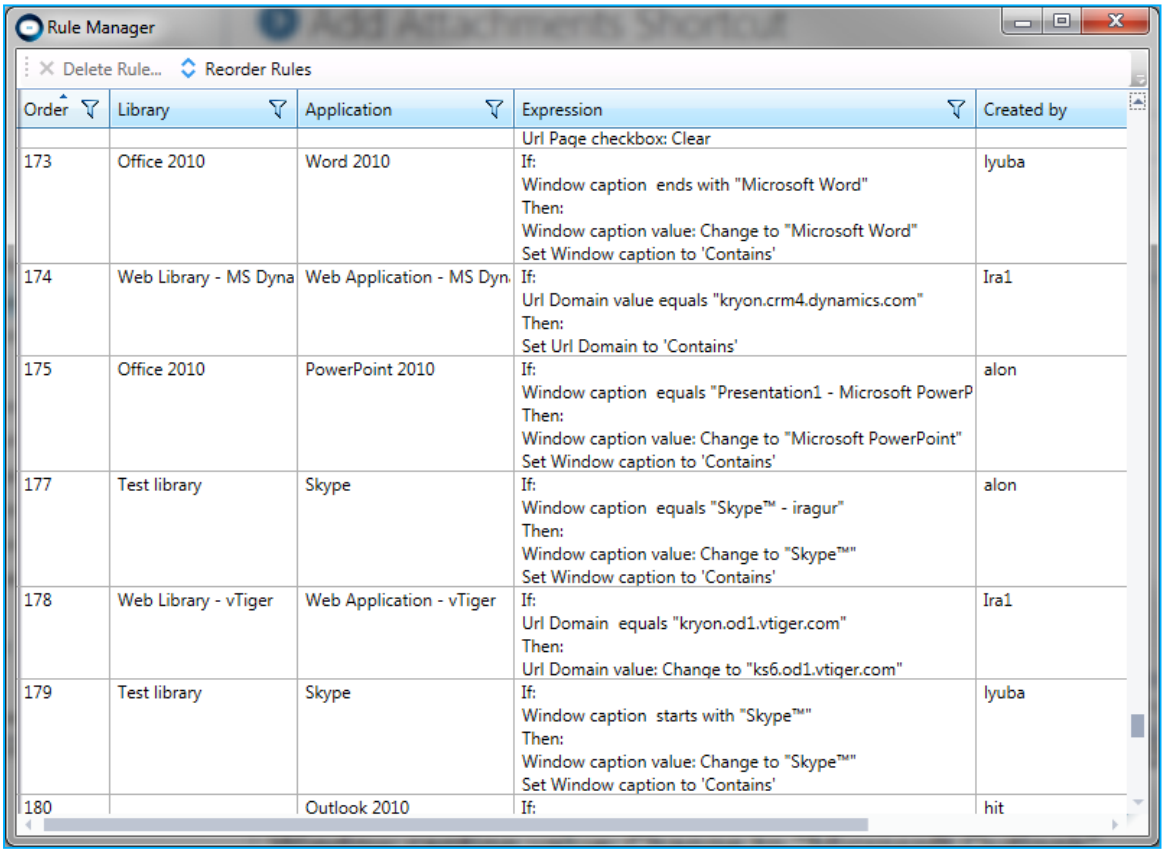
The Kryon RPA platform allows you to apply rules and conditions to the window settings within a wizard. To learn about how window detection rules are created, see Adding a Window Detection Rule.

The RULE MANAGER

The **RULE MANAGER** is where you can view and manage all the window detection rules created for your company. To access it, from the menu bar of the main Kryon Studio window, click **Tools > Rule manager**.

The **RULE MANAGER** contains the following columns:

- **Order:** The order by which the rules are applied
- **Library:** The library that the rule applies to
- **Application:** The application that the rule applies to
- **Expression:** The content of the rule
- **Created by:** The name of the user who created the rule
- **Creation date:** The date and time on which the rule was created

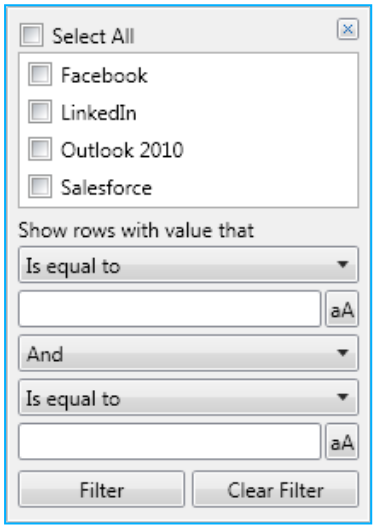


Filtering the RULE MANAGER table

The **RULE MANAGER** is a table of all rules that exist for your company. This table can be filtered by column, so that only rows containing specific types of information are shown in the manager, while rows containing other types of information are hidden.

Follow these steps to filter the table:

1. Click the Filter icon in the column heading of the attribute by which you want to filter the table
2. The following dialog box appears:



3. Select or clear the checkboxes of the column values that you want to display or hide in the table
4. From the first condition dropdown list, select the first condition that you want to apply to the filter, and type a value in the condition field

   A preview of the filtered table appears in the **RULE MANAGER**

> **TIP**
>
> To make your filter value case-sensitive, click the **Match Case** button aA.

5. Apply a second filter by continuing with step 6. Otherwise, go to step 8.
6. From **the And/Or** dropdown list, select a condition
7. From the second condition dropdown list, select the second condition that you want to apply to the filter, and type a value in the condition field

   A preview of the filtered table appears in the **RULE MANAGER**
8. Click **Filter** to apply the filter

The rule table is filtered by the conditions you set. The **Filter Applied** icon appears on the column heading by which you filtered the table

NOTE

When you close the manager dialog box, all filters are cleared so that when you reopen the manager, the table is unfiltered.

Reordering rules

Reordering rules affects the order in which the rules are applied for all wizards in the company.

Follow these steps to reorder rules:

1. From the **RULE MANAGER**, click **Reorder Rules**
2. At the top of the **RULE MANAGER**, the **Reorder Rules** toolbar appears
3. Select the rule that you want to move, and click the **Move Up** or **Move Down** icon to move the rule
4. Repeat step 3 for each rule that you want to move
5. Save your changes by clicking **Save Changes** on the **Reorder Rules** toolbar

TIP

To sort the table by a specific column, click the relevant column header.

Deleting a rule

When a rule is deleted, it is no longer available for to apply to any wizard in the company. However, deleting a rule does not affect existing wizards.

Follow these steps to delete a rule:

1. From the **RULE MANAGER**, select the rule that you want to delete, and click **Delete Rule**
2. From the confirmation dialog box that appears, click **OK**

   The rule is deleted from the table

Adding a Set of Window Detection Data

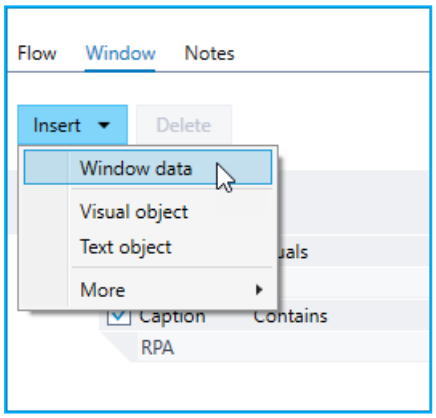
For some windows, the window class, caption, or URL changes when the wizard is played in a different environment than recorded. Kryon Studio enables you to add multiple sets of window detection data, such as window class or URL information, to a step to ensure that the robot detects the recorded window in different environments.

NOTE

When a step contains multiple sets of window detection data, the window will be detected if it matches all the active properties of **any** of the sets.

To add a set of window data to the selected step:

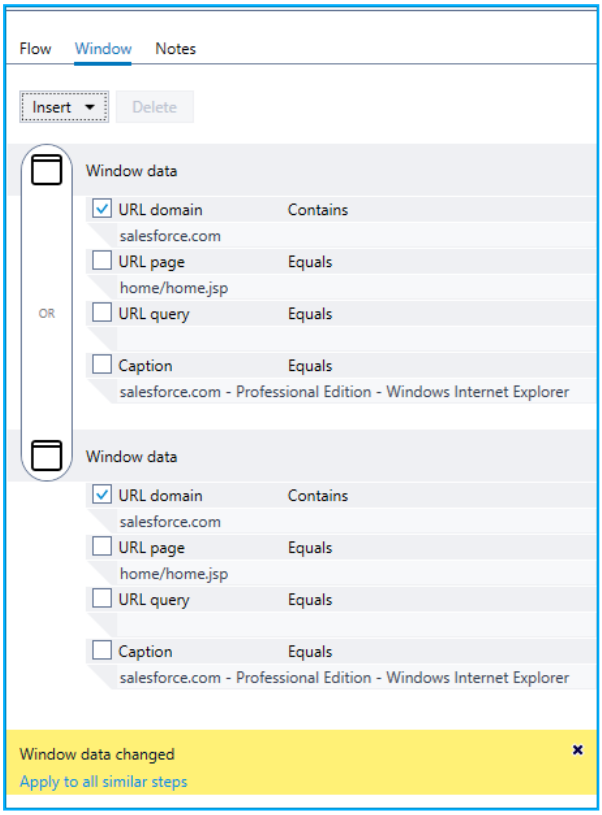
1. Go to the Window tab of the Flow Pane
2. Click the **Insert** button, and select **Window data**



A new set of window data appears, containing the relevant properties (based on whether the recorded windows is a desktop application or web application)

3. For each set of window data, add/remove the relevant details (see Editing Window Properties)

4. Results:
   - The window data is updated
   - If this window data is being updated for the first time, Studio prompts you to add a rule
   - If the window in this step is shared by other steps in the current wizard, Studio prompts you to apply the change to all similar steps



TIP

To remove a set of window detection data, click on the **Window data** heading for that set, then click the Delete button.

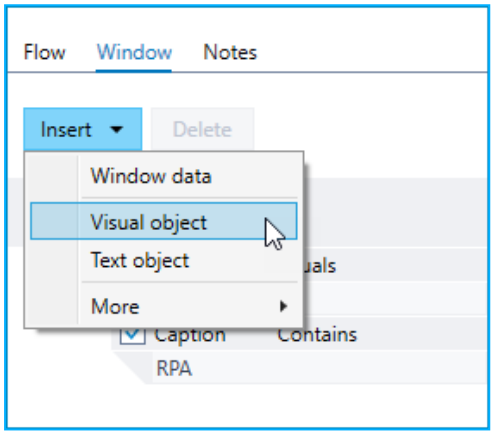
- When there is only one set of window data for the step, the Delete button will not be available.

Adding a Visual Object

In some applications, multiple windows share the same window class/caption. In such cases, the robot cannot rely on window properties alone to detect the correct window. In such cases, you can specify a visual object that appears in the window – giving the robot the information it needs to detect the correct window.

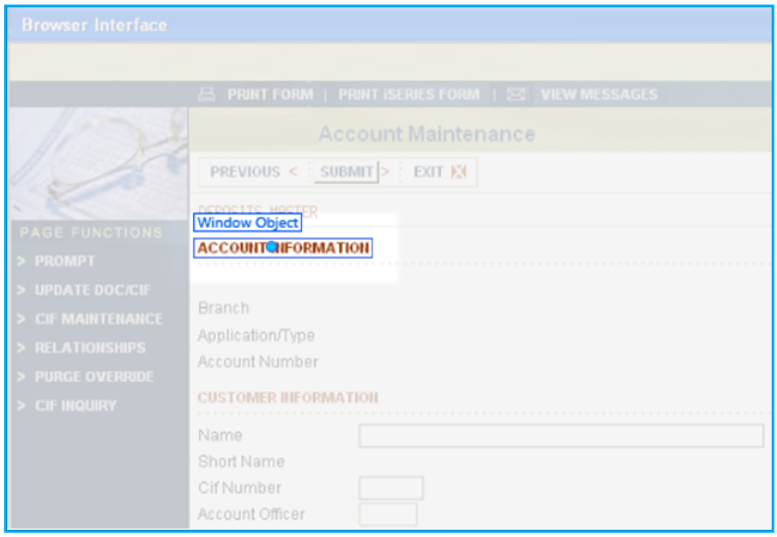
To add a visual object to the selected step's window detection properties:

1. Go to the Window tab of the Flow Pane
2. Click the **Insert** button, and select **Visual object**



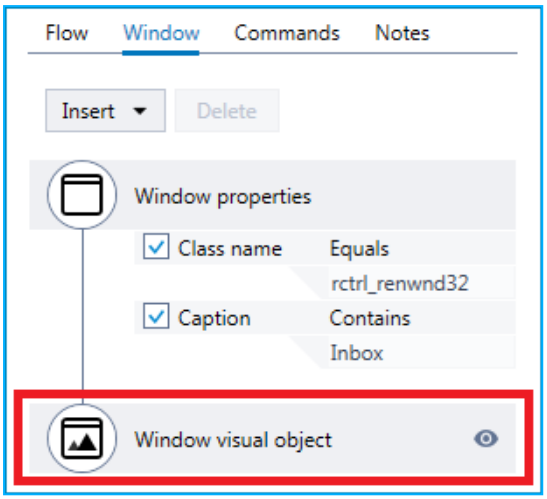
3. When you move the mouse into the Display Pane, the cursor will turn into a **plus (+)**
4. Click the object you want the robot to detect as a window object

   The **Window Object** box appears around the object



5. Drag the **Window Object** box to reposition it or its borders to resize it

The object is added to the step's window detection properties



TIP

You can add multiple window objects by repeating this procedure. Each additional window object is a fallback in case the preceding window object is not detected.

Window Options

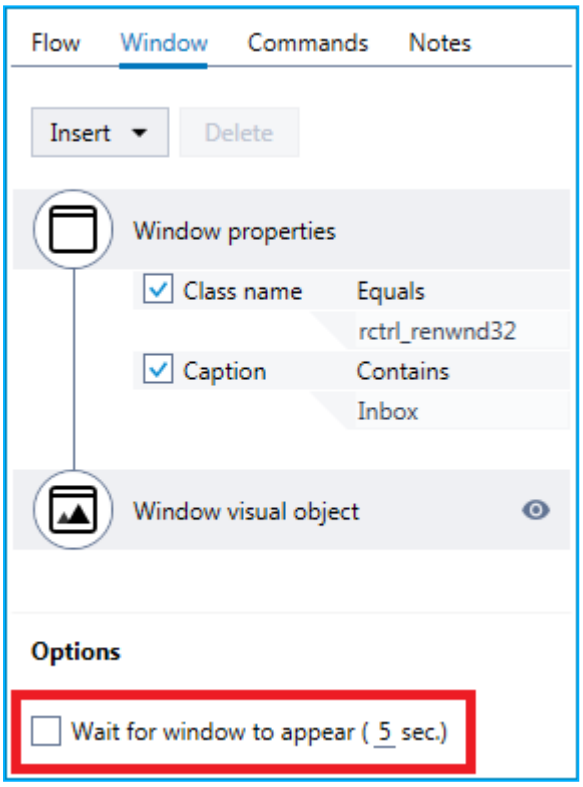
In addition its detailed options for managing window detection, the **Window** tab of the **Flow pane** offers additional options for managing window behavior.

Wait for window to appear

If you want the robot to wait some extra time for the step's specified window to appear before failing window detection, tick the **Wait for window to appear** checkbox and specify the maximum wait time. If the window does not appear by the time specified, the step's fallback command will be executed. If the window appears before the maximum time is up, the wizard will immediately continue without waiting the remaining time.

TIP

Utilizing this option can prevent wizard failure in case of slow systems or network/internet connections.



Bring window to front

The **Bring Window to Front** checkbox enables you to determine whether the robot should automatically activate the window on which the robot performs the action. The feature options are:

- **Automatically:** The robot automatically determines whether it should bring to front (i.e., activate) a window in order to perform the action. If the window is already activated, the robot does not attempt to activate it.
- **Always:** The robot always attempts to activate the window. This is useful for preventing cases in which the window might be unexpectedly deactivated by another window that has been brought to the front.
- **Never:** The robot never attempts to activate the window, and performs the action on the window only if it is already activated.

NOTE

Not applicable to sensors.

Wait for web page to download

In steps that contain a web browser window, the **Wait for web page to download** checkbox enables you to determine whether the robot should wait for the web page to download before starting the step.

The web page download options are:

- **Automatically:** The wizard waits while the page is loading. As soon as the page is downloaded, the wizard immediately starts the step. This is the default setting.
- **Always:** The wizard checks if the page has finished downloading. For pages that do not require downloading, this creates a slight delay of up to a few seconds before the step starts.
- **Never:** The wizard never waits for the page to download and immediately attempts to start the step.

TIP

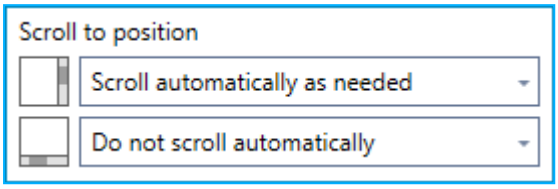
Utilizing this option can prevent wizard failure in case of slow systems or network/internet connections.

NOTE

Not applicable to sensors.

Scroll to position

In steps that contain scrollbars for the recorded window, you can determine whether the robot should automatically scroll to the step's click position if it is located outside the area shown in the user's application window. This option is set separately for vertical and horizontal scrollbars.



The automatic scrolling options are:

- **Scroll automatically as needed:** The robot detects whether the click position is located outside the area shown in the user's application window, and automatically scrolls up or down as needed to reach the click position.
  - This is the default setting for recorded web applications.
- **Do not scroll automatically:** The robot does not detect or scroll to the click position if it is located outside the area shown in the user's application window.
  - This is the default setting for recorded desktop applications.

Custom window name Attended/Hybrid Only

Allows you to customize the window name that appears in Paused messages to the end user.

CHAPTER 6: Object Detection

A action's detected object is the object that was clicked on during wizard recording. By default, this is the object that the robot will act on when the wizard is run. However, there are numerous methods by which a step's detected object can be moved, customized, or changed in order enhance robot performance and to ensure position accuracy. Additionally, in certain circumstances, detected objects can be used for purposes other than determining a step's core action position.

This chapter describes the basics of object detection. Detailed information about optimizing and customizing detected objects can be found in the Core Actions chapter of this guide.

In this chapter:

Detected Object Types 84

Detected Object Display 86

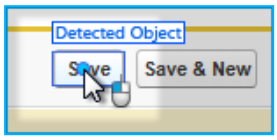
Object Detection Criteria 87

Reusing a Detected Object 90

Detected Object Types

There are several possible ways in which wizards make use of detected objects:

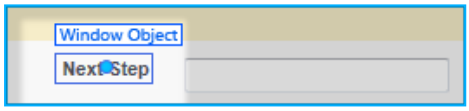
- **Core action:** The core action is recorded, and captured by default as a click, during the recording. There can be only core action per step.

  Core actions that use a detected object are:

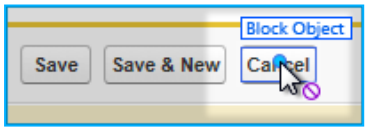
  - Click
  - Hover
  - Detect object
  - Read from screen

- **Window visual object:**

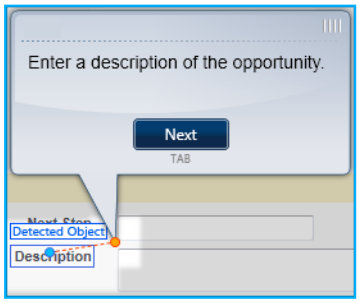
  Read more about window visual objects.

- **Object block:**

  Read more about object blocks.

- **Anchor object (for an anchored bubble):**

TIP

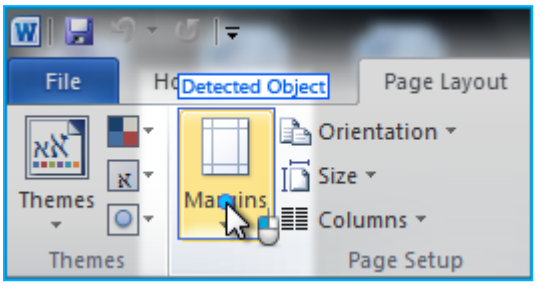
If you are unsure what each detected object in a step does, click on it in the main Wizard Editor window, and Studio will highlight the relevant action in the Flow Pane on the right. If the detected object belongs to an anchored bubble, for example, Studio will highlight that bubble action.

CAUTION

Any type of detected object – regardless of its function in the step and whether it is clicked or simply detected on the screen – must meet the object detection criteria.

Detected Object Display

This is how a detected object initially appears in the Wizard Editor:



However, once a wizard step is edited, the display of a detected object can include the following elements:

- **Detection box:** A blue bounding box that surrounds the detected object
- **Click:** A blue dot that (if no offset is used) represents the click position
- **Offset:** An orange dot created by offset (if used) that represents the click position
- **Highlight box:** An orange bounding box that represents the core action's highlight box position
- **Object block position:** A purple bounding box that represents a blocked area of the screen

Object Detection Criteria

The detection mechanism is designed to detect distinct GUI items such as icons, buttons, and text.

All detected objects that you expect the robot to detect must meet three criteria to ensure proper detection. A detected object must be:

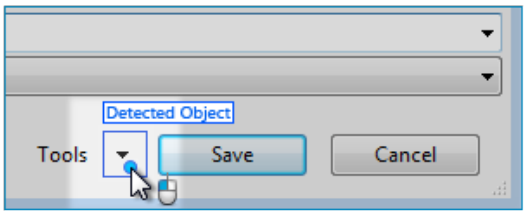
- Unique
- Static
- Clean

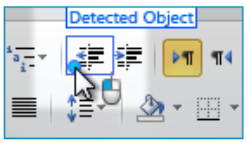
If an object does not meet the object detection criteria, use an offset.

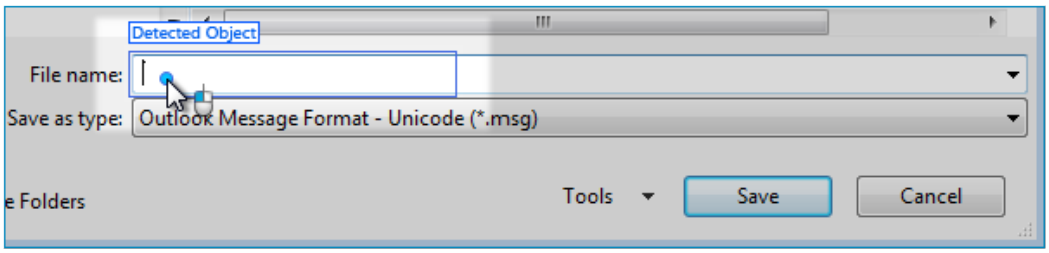
Unique

A unique object doesn't have any identical or near-identical siblings anywhere in the recorded window.

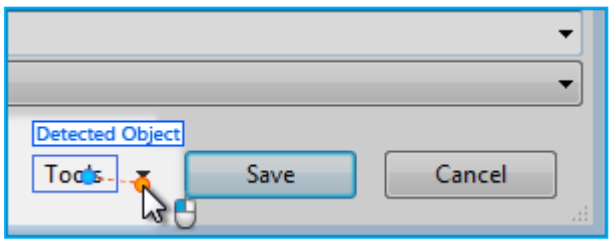
- Here are some examples of non-unique objects:







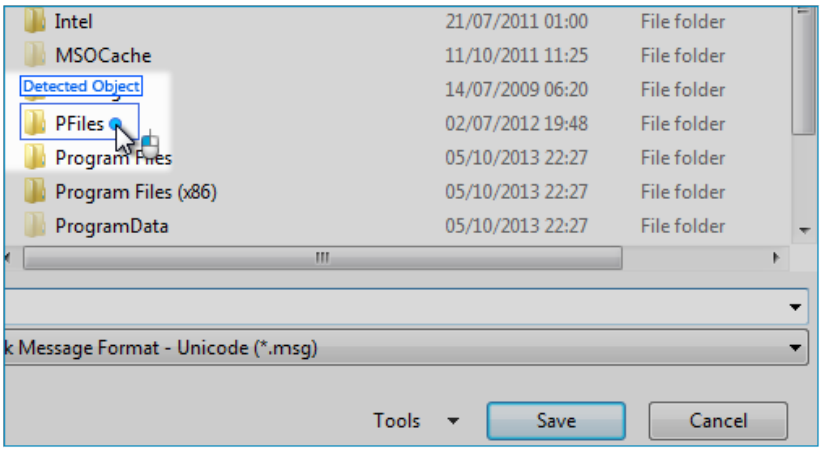
- And here is an example of a unique object:

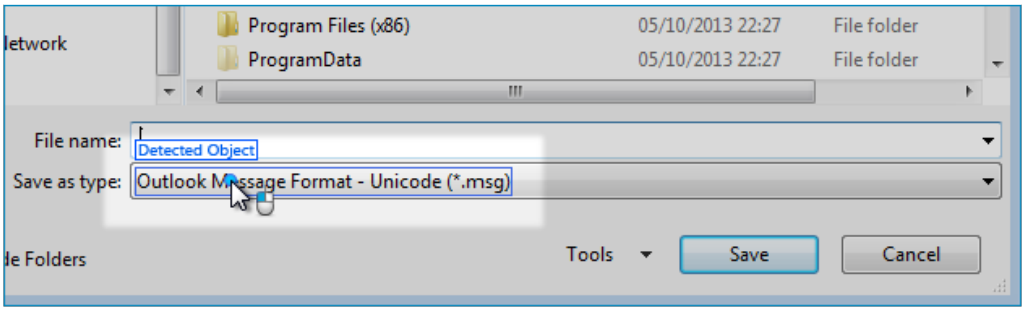


Static

A static object's appearance never (or rarely) changes

- Here are some examples of non-static objects:



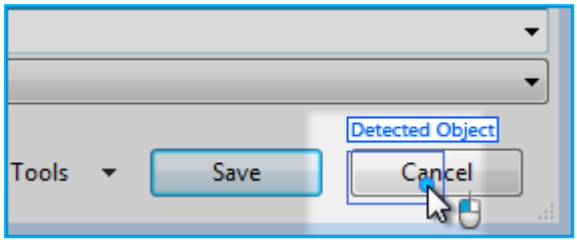


- And here is an example of a static object:

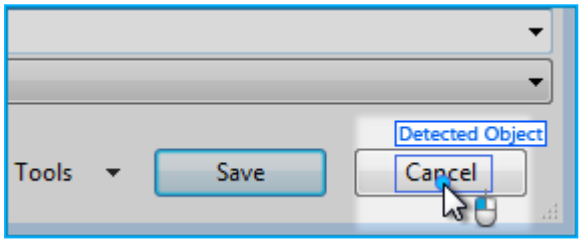
Clean

A clean object isn't chopped, doesn't include any blank areas, and doesn't include unncessary text, blank areas, lines or other shapes.

- Here is an example of a non-clean object:



- And here is an example of a clean object:

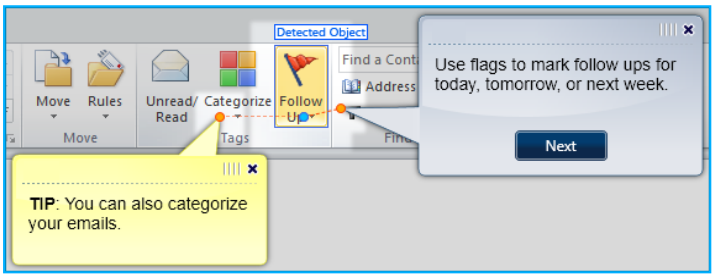


Reusing a Detected Object

Reusing detected objects ensures optimal robot performance. You can reuse an object either by copying it or by snapping one object over another. The robot detects the object once, and then reuses this detection data wherever the object has been reused in the wizard. This saves the time it would take the robot to detect the object over and over.

Reusing a detected object within the same step

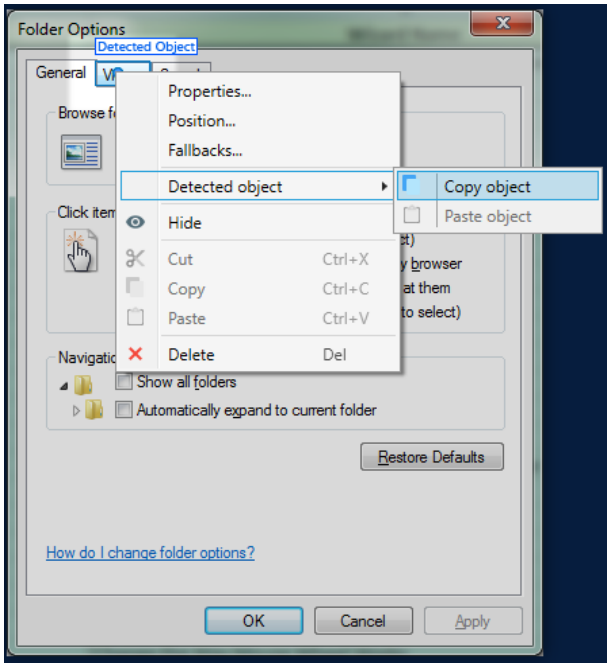
To reuse an object within the same step, snap one object's blue dot over another object's blue dot:



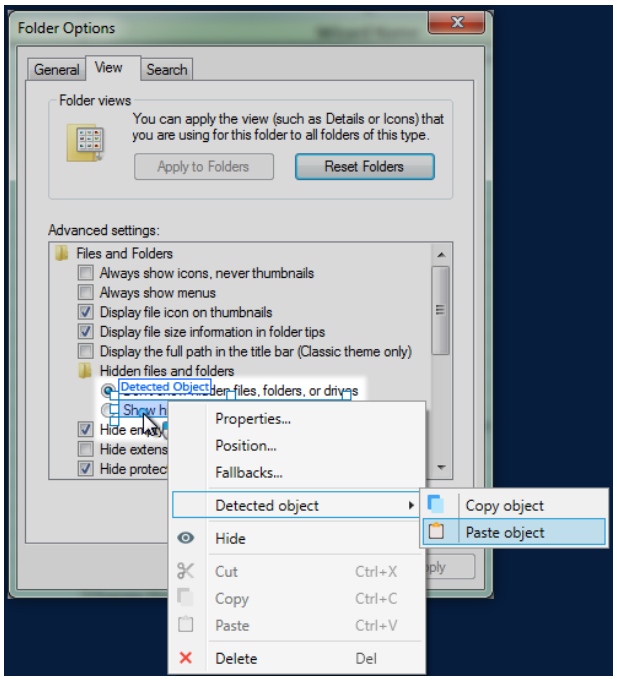
Reusing a detected object across different steps

To reuse an object across different steps:

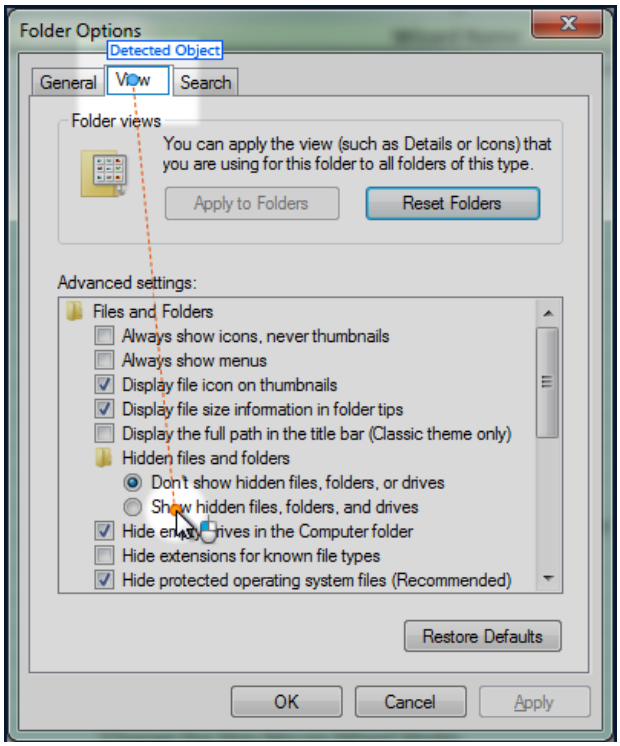
1. Copy the object in one step



2. Paste over the object in the second step



3. The detected object is now the same in both steps



CAUTIONS

When reusing a detected object across two or more steps, all steps must use: (1) the exact same screen; and (2) the exact same window size.

If you change the settings of a detected object **after** you have copied it to other step (s), the new settings are not automatically updated in the other step(s). To do so, you must modify object settings in the other step(s) manually.

CHAPTER 7: Editing Wizard Flow

When you finish recording a wizard, it appears in the Wizard Editor exactly as you recorded it. Editing a wizard consists of; (1) organizing your recording (editing wizard flow); and (2) adding layers of functionality that turn your basic recording into a robust automation tool (editing wizard steps).

The remainder of this chapter covers editing wizard flow (i.e., reordering and organizing) the steps of your wizard. Topics specific to editing wizard steps themselves are covered in later chapters.

In this chapter:

Moving a Step 94

Copying or Duplicating a Step 95

Appending Steps 96

Deleting a Step 97

Adding or Editing an Embedded Wizard 98

Organizing Steps by Color 101

Undoing & Redoing Changes 102

Moving a Step

Studio enables you to reorder the flow of a wizard by dragging or cutting and pasting steps.

To move a step:

1. In the Navigation pane, select the step that you want to move
2. Do one of the following:
   a. Click and drag the step, and drop it in its new location
   b. Cut and paste the step as follows:
      - Cut the step by doing one of the following:
        - On the Wizard Editor menu bar, select **Edit > Cut**
        - On the toolbar, click the **Cut** icon
        - Press `CTRL+X`
      - In the Navigation pane, select the step after which you want to paste your step
      - Paste the step by doing one of the following:
        - On the menu bar, select **Edit > Paste**
        - On the toolbar, click the **Paste** icon
        - Press `CTRL+V`

Copying or Duplicating a Step

Copying a step

Studio enables you to copy steps within a wizard or from one wizard to another.

To copy a step:

1. In the Navigation pane, select the step that you want copy
2. Copy the step by doing one of the following:
   - On the menu bar, select **Edit > Copy**
   - On the toolbar, click the **Copy** icon
   - Press `CTRL+C`
3. In the Navigation pane, select the step after which you want to paste your step
4. Paste the step by doing one of the following:
   - On the menu bar, select **Edit > Paste**
   - On the toolbar, click the **Paste** icon
   - Press `CTRL+V`

Duplicating a step

You can duplicate a step only within the same wizard.

To duplicate a step:

1. In the Navigation pane, select the step that you want to duplicate
2. Press `CTRL+D`

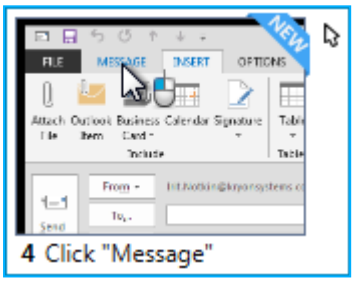
Appending Steps

Studio enables you to add steps to an existing wizard. Use the same best practices for appending as you would for recording. Appended steps are edited as regular steps.

To append steps to a wizard:

1. In the Navigation pane, select the step ***after*** which you want to append your step
2. Do one of the following:
   - On the menu bar, select **Wizard > Append Recording**
   - On the toolbar, click **Append Recording**
3. Continue with Recording a Wizard | Start recording | step 3

The steps you appended appear in the Navigation pane after the selected step, with a **New** label in the thumbnail's top-right corner:



NOTE

The **New** label disappears when you select a different step.

Deleting a Step

Studio enables you to delete any steps that are not required for the wizard flow.

CAUTION

Step deletion is permanent and cannot be reversed. To recreate a deleted step, you must append the step by re-recording it.

Delete a step from a wizard by doing the following:

1. In the Navigation pane, do one of the following:
   - Right-click the step and select **Delete**
   - Select the step and press **DELETE**
2. In the confirmation message that appears, click **OK** to permanently delete the step

CAUTION

Step deletion cannot be undone, even if you have not saved your changes.

Adding or Editing an Embedded Wizard

An embedded wizard is a regular wizard inserted as a building block into the flow of another wizard, and its steps are reused by other wizards. Its steps are integrated into the containing wizard's flow of steps.

NOTES

- Embedded wizards are not available for sensors.
- A wizard cannot be embedded into itself.

Adding an embedded wizard

To add an embedded wizard into the flow of a currently open wizard:

1. In the current wizard's Navigation pane, select the step ***after*** which you want to embed the wizard
2. On the menu bar, select **Insert > Embedded Wizard**
3. From the **Catalog** dialog box that appears, select the wizard that you want to embed

> **TIP**
>
> To search for a specific wizard, click **Find** or press `CTRL+F` and then type your search query.

4. Results:
   - The wizard you selected is embedded as a step into the containing wizard's flow.
   - The embedded wizard's thumbnail appears in the Navigation pane, and its core action is indicated by the **Embedded** icon
   - The embedded wizard's control pane appears in the Wizard Editor's Display pane

> **TIP**
>
> To preview an embedded wizard's steps, from its control pane, click the **Sample screenshots** dropdown list

Editing an embedded wizard

To edit an embedded wizard:

1. From the containing wizard's Navigation pane, select the embedded wizard step
2. In the embedded wizard's control pane, click **Edit wizard**

The embedded wizard is opened for editing in a new **Wizard Editor** window

TIP

You can also edit the embedded wizard by opening it directly from the catalog in the main Kryon Studio window.

CAUTION

Changing it once changes it everywhere!

Note that when you edit an embedded wizard, your changes are made to the embedded wizard itself, **NOT ONLY** in the containing wizard.

This means that the changes you make to an embedded wizard will affect all other wizards into which it is embedded and/or sensors from which it is launched.

To check for other wizards/sensors in which the embedded wizard is used, check its **EMBEDDED WIZARDS** tab.

Organizing Steps by Color

Steps in a wizard can be assigned colors to help keep them organized and make them easier to identify.

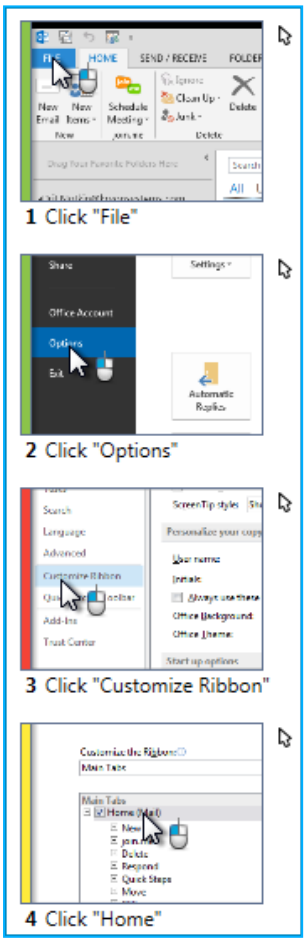
TIP

Assign colors to steps to help keep track of wizard structure (e.g., to follow sub-processes, created by decision points and fallbacks, branching out of the main wizard structure).

To assign a color to a step:

1. In the Navigation Pane, right-click the thumbnail of the step to which you want to assign a color and select **Color**
2. Select a color from the list

The color you selected appears along the left margin of the step's thumbnail:



Undoing & Redoing Changes

Most changes you make while editing a wizard can be undone and/or redone until you close the wizard.

CAUTION

Once you close a wizard, changes you have made to the wizard can not be undone.

To undo or redo an action while editing a wizard, do one of the following:

- On the menu bar, select **Edit > Undo** or **Edit > Redo** (as applicable)
- On the toolbar, click either the **Undo** icon or **Redo** icon (as applicable)
- Press `CTRL+Z` to undo or `CTRL+Y` to redo an action

CHAPTER 8: Editing Wizard Steps

Once the steps of the wizard are in the correct order, the second goal in editing a wizard is to edit the steps themselves to adjust and/or add to their functionality. This process involves working with the following mandatory and/or optional components of each step:

- **Step actions**

  Each wizard step is made up of a series of actions that occur in a sequence. These actions are at the heart of the functionality of each wizard step:

  - **Step Start:** the step's opening action; inserted by default and cannot be moved or removed
  - **Core Action:** the step's primary action (i.e., the interaction with the object detected during recording); inserted by default, but **CAN** be moved or removed
  - **Step End:** the step's closing action; inserted by default and cannot be moved or removed
  - **Blocks**: always occur immediately after the **Step Start** action; optional action that can be inserted and removed, but cannot be moved. Multiple blocks can be added to a single step.
  - **Bubbles:** occur either before or after the **Core Action**: optional action that can be inserted, moved, or removed. Multiple bubbles can be added to a single step.
- **Fallbacks**
- **Advanced commands**

The remainder of this chapter covers the **Step Start** and **Step End** step actions. Succeeding chapters deal with each of the other step actions.

In this chapter:

Step Start 104

Step End 110

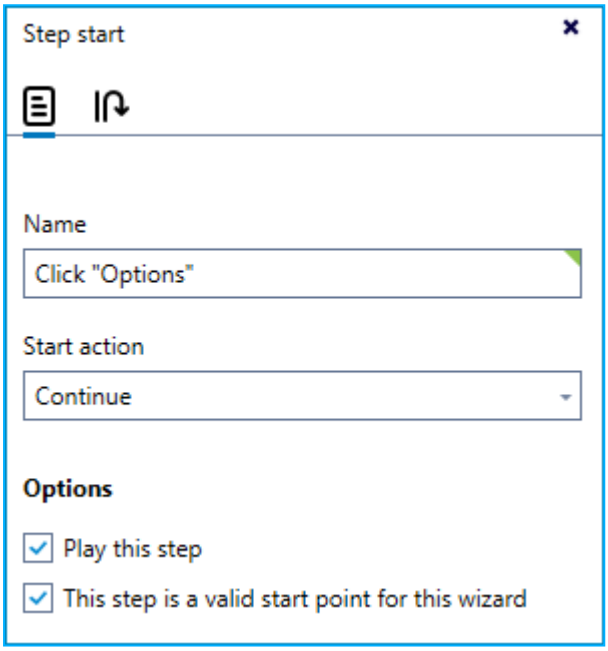
Step Start

Step start is a mandatory action that is added by default. It is triggered at the beginning of the step, and cannot be moved or removed.

Step start enables you to view the step's opening event, and to set an opening command for the selected step by selecting an option from the **When starts** dropdown list.

Step start properties

Step Start contains the following properties:

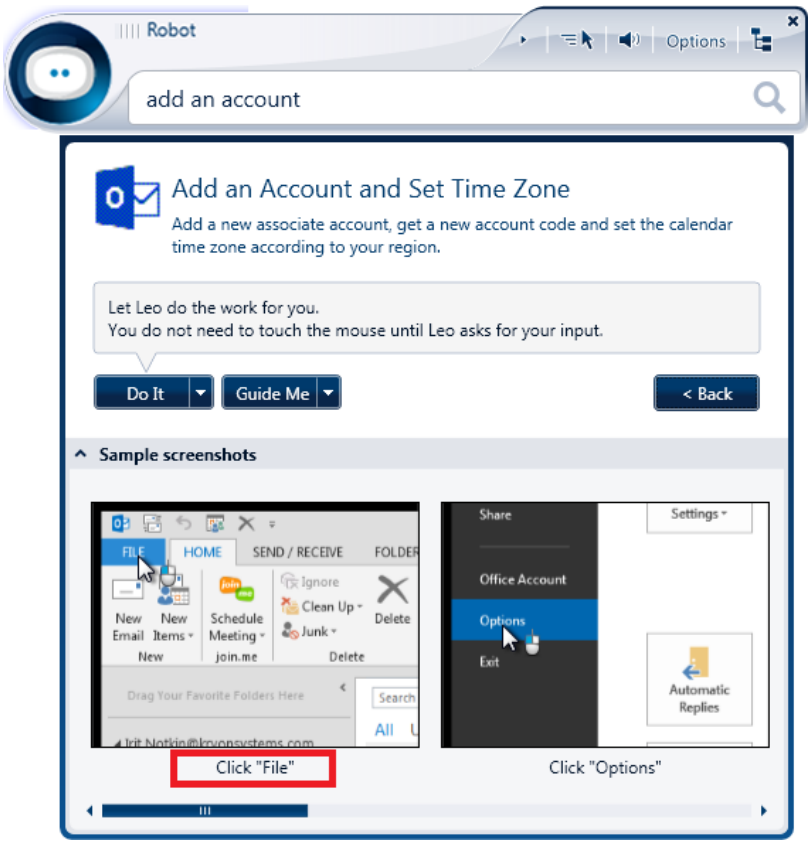


- Name: a label that identifies the step
- Step start action: determines which command to trigger when the selected step begins
- Options: additional options related to **Step start**

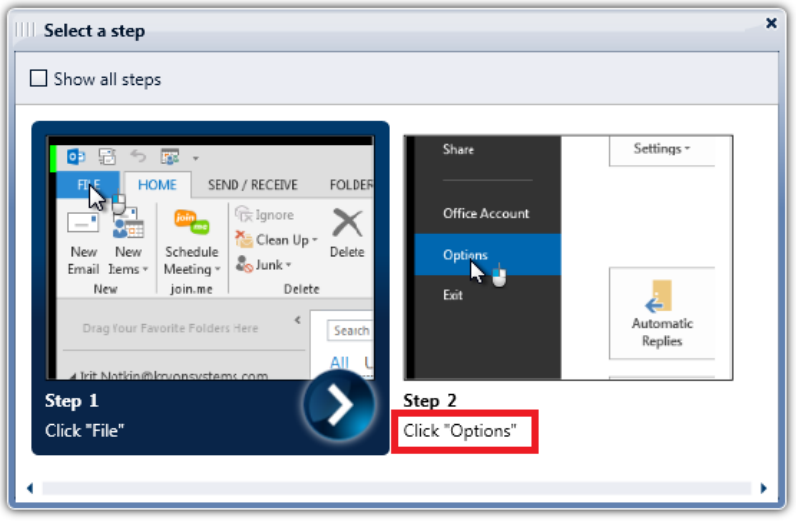
Name

Step names are labels that identify a wizard step according to its purpose. Step names are generated automatically by Studio (when possible) or left blank (when not), and can be customized by the RPA developer. Each step name is visible to end users in the following locations:

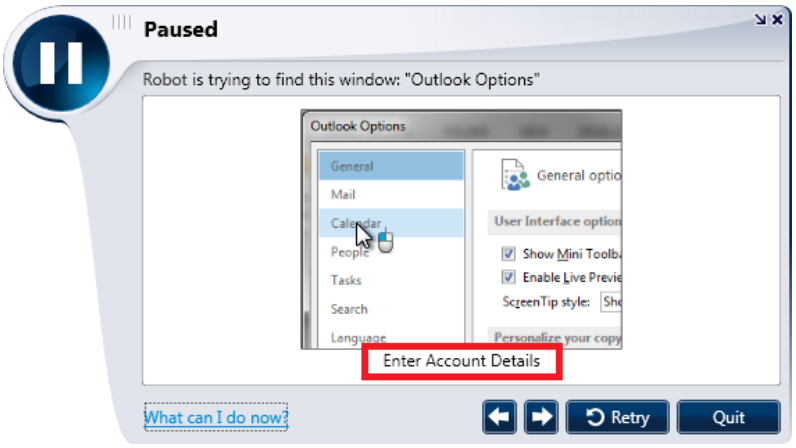
- **Published Wizards in Kryon Robot:**
  - Sample screenshots in the search bar or catalog:



  - **Do It/Guide Me** from a selected step:



- Paused step image:



- **Wizards Exported to Documents:** Wizards exported to Word and PowerPoint documents whose template includes the step name

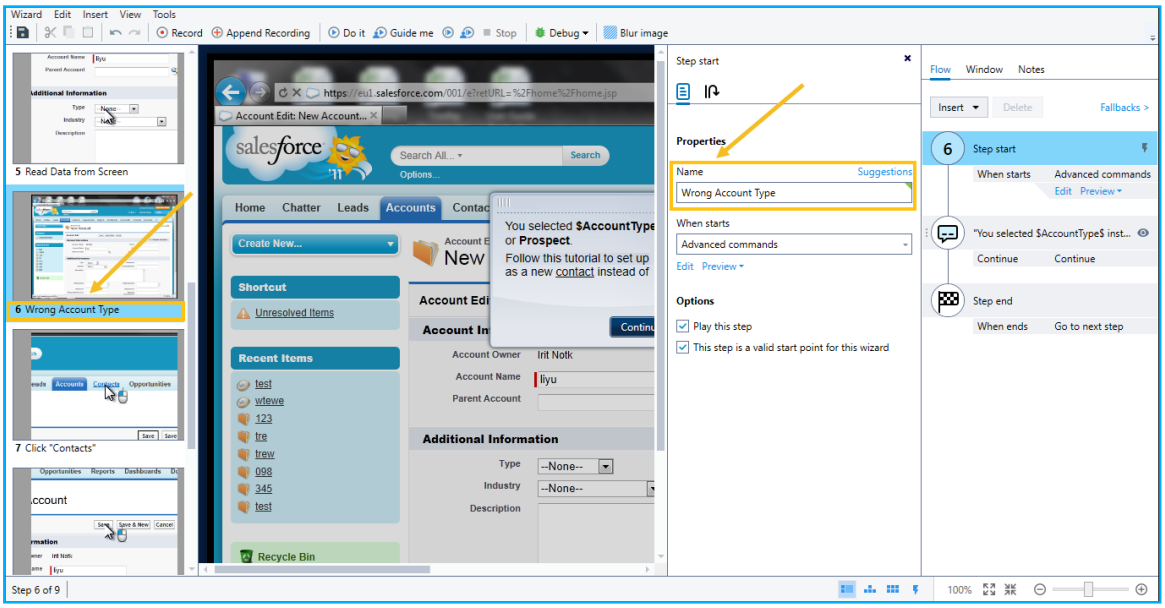
NOTE

Sensor step names are not visible to end users in Kryon Robot.

To customize the name of the selected step:

1. From the Flow Pane, access the **Step start** Properties Pane
2. In the **Name** field of the Properties tab, type a name for the step

The updated step name appears in the Navigation Pane under the step thumbnail:



Step start action

The **Step start** action (labeled **When starts**) enables you to set one of the following commands with which to open the selected step:

- **Continue:**When the step begins, the wizard continues the flow to trigger other actions in the step in the order determined by the RPA developer.
- **Advanced commands:** When the step begins, the wizard triggers a predefined set of advanced commands. When the advanced commands are completed, the wizard continues by executing the next actions in the step flow.

To set the selected step's **Step start** action:

1. In the Flow tab of the Flow Pane, select the desired command from the **When starts** dropdown list that appears under **Step start**

Options

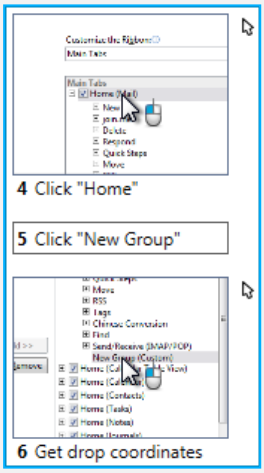
The following sections describe additional **Step start** options available from the Properties tab of the Properties Pane.

Enable/disable step

The **Play this step** checkbox enables you to disable a step without removing it from the wizard. This is useful when you do not need to play a step in a wizard but want to keep it for future use or reference.

NOTES

- After recording, all steps are enabled by default other than steps containing recorded scrolling or drag-and-drop actions. For additional information, see Scrolling & dragging.
- By default, disabled steps are hidden in the Navigation Pane (as shown below). For information about how to view them, see Displaying disabled step thumbnails.



- In Diagram View, disabled steps are always completely hidden and cannot be shown.

Valid wizard start points

Studio enables you to determine whether the selected step will serve as a valid starting point for the wizard, that is, whether the user can choose to start the wizard from this step. This option is controlled from the checkbox labeled **This step is a valid start point for this wizard**.

Step start fallbacks

The **Step start** fallback events are:

- Window not found
- Window blocked

NOTES

- **Step start** fallbacks are not available for embedded or launched wizards
- The Fallbacks tab of the **Step start** Properties Pane is indicated by the **Fallbacks** icon

Step End

Step end is a mandatory action that is added by default. It is triggered at theend of the step, and cannot be moved or removed.

Step end enables you to view the step's closing event, and to set a closing command for the selected step by selecting an option from the **When ends** dropdown list.

Step end properties

The **Step end** action (labeled **When ends**) enables you to set one of the following commands with which to end the selected step:

- **Go to next step:** When the step is completed, the robot goes to the next step in the wizard flow (this is the default action)
- **Go to step:** When the step is completed, the robot goes to the step you specify
- **Remove all blocks:** Dismisses all blocks and cancels their functionality for that run of the wizard or sensor
- **Keyboard shortcut:** When the step is completed, the wizard triggers a predefined keyboard key combination (e.g., `ALT+P` to trigger the print command)

  NOTES

  - The **Keyboard shortcut** action is not applicable to mouse steps.
  - When the **Keyboard shortcut** action is selected, an option appears in Properties Pane, allowing you to determine whether the wizard will accept input in any language (default) or only the language in which the wizard was recorded

- **End wizard successfully:** When the step is completed, the robot exits the wizard and reports the ending as successful.
- **End wizard unsuccessfully:** When the step is completed, the robot exits the wizard and reports the ending as unsuccessful.
- **Advanced commands:** When the step is completed, the robot triggers a predefined set of advanced commands. When the advanced commands are completed, the robot continues the wizard flow in the order determined by the RPA developer.

To set the selected step's **Step end** action:

1. In the Flow tab of the Flow Pane, select the desired command from the **When ends** dropdown list that appears under **Step end**

CHAPTER 9: Core Actions

A core action is the action automatically recorded in a step when you use either your mouse or keyboard while recording on the target application.

In this chapter:

Core Action Types 112
Changing, Restoring, or Deleting a Core Action 115
Core Action Properties 118
Core Action Position 121
Core Action Position: Detected Object 123
Core Action Position: Fixed Position 128
Core Action Position: Relative Position 129
Core Action Position: Offset 130
Core Action Position: Highlight Box 133
Detection Match Configuration 135
Core Action Fallbacks 141

Core Action Types

The core action types are:

- Mouse actions:
  - Click
  - Hover
  - Detect object
  - Read from screen
- Keyboard
- Wait

Click

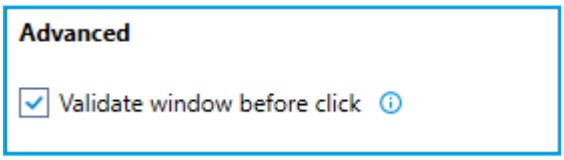
The **Click** core action is the default for steps recorded when the mouse is clicked. It reflects the type of mouse click made during recording: single, right-click, left-click, double-click, or other.

NOTE

Not applicable to sensors.

Validate window before click

By default, before any **Click** action, the wizard validates the current window is the same as the window detected at the beginning of the step. In order to deactivate that validation, clear the **Validate window before click** checkbox in the Position tab of the **Click** action's Properties Pane.



Hover

The **Hover** core action moves the cursor on the screen to where a click was recorded in the step, but it does not perform the actual click action. This can be useful, for example, when the mouse needs to hover over menu options in order to open a sub-menu (as is often the case for websites).

NOTE

Not applicable to sensors.

Detect object

The **Detect object** core action confirms the existence of the object clicked during the recording, but it does not move the cursor or perform the actual click. This can be useful, for example, when you need to ensure that a certain object is visible on the screen before continuing the process (e.g., the **Developer** tab in Microsoft Office).

Read from screen

The **Read from screen** core action instructs the robot to read data from a field on the screen and use that data as a variable in one or more Advanced Commands. The field value is read in real time when the wizard is run.

Read from screen can be useful, for example, when the flow of the wizard depends on the value of a specific field that appears on the screen.

For detailed information, see the chapter Read from Screen.

Keyboard

The **Keyboard** core action is the default for steps recorded when key(s) are pressed on the keyboard. The action records the actual sequence of keys pressed (which is often replaced during wizard runtime by the value of a variable set using Advanced Commands).

NOTE

Not applicable to sensors.

Wait

The **Wait** core action causes the wizard to wait for one of the following events to occur:

- **Wait for block removal:** The wizard waits for all blocks to be activated or removed before continuing the wizard flow. As long as a block is not activated or removed, the **Wait** action persists. Once a block is activated, the wizard flow continues.

  NOTE

  Wait for block removal is a valid core action only if the step contains blocks. For more information about blocks and how to use them, see the Blocks chapter of this guide.

- **Wait for window close:** The wizard waits for the step's detected window to close – either indefinitely or for a specified maximum amount of time. If the window is not closed within the time specified, the wizard continues to the next action in the flow.
  - By default, the **Max wait** checkbox ticked, and the maximum wait is set to 5 seconds
  - If the **Max wait** checkbox is not ticked and the wizard waits indefinitely
- **Wait for object disappearance:** Waits for the step's detected object to disappear – either indefinitely or for a specified maximum amount of time. If the object does not disappear within the time specified, the wizard continues to the next action in the flow.
  - By default, the **Max wait** checkbox ticked, and the maximum wait is set to 5 seconds
  - If the **Max wait** checkbox is not ticked and the wizard waits indefinitely

Changing, Restoring, or Deleting a Core Action

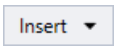
Since **Recorder** captures captures only mouse clicks and keystrokes when recording, Studio gives you the option to change a step's core action if you need your wizard to execute a different core action type.

NOTE

In any step, the core action can be deleted, replaced with another core action, or restored if it was previously deleted. Note, however, that each step can have at most **only one core action**.

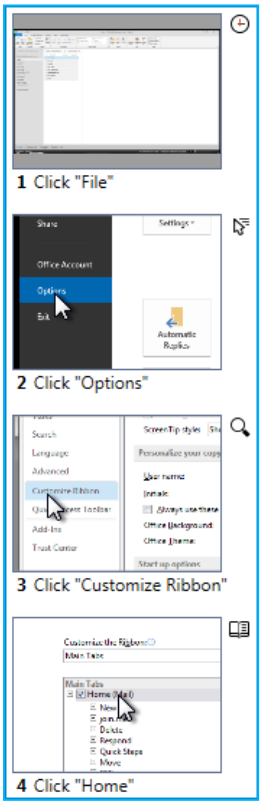
Changing or restoring a core action

To change or restore the selected step's core action:

1. Go to the Flow tab of the Flow Pane
2. Click the **Insert** button 

3. Select the desired core action from the dropdown list

The step core action is updated with the selected action, and the action icon appears in the step thumbnail's upper-right corner.



NOTE

Disabled core actions

- A Click core action can not be replaced by a Keyboard core action
- A Keyboard core action can not be replaced by a Click core action or Wait for object disappearance

TIP

Don't forget to give it a try!

If you change the core action for any step in the wizard, make sure to run the wizard to verify that the modified step works properly.

Deleting a core action

Deleting the core action keeps the step enabled and retains all other functionality layers in the step, while disabling the core action. This is useful, for example, when the step's purpose is to inform users or prompt them to perform the action instead of the robot.

EXAMPLE

When you delete a rule in Microsoft Outlook, Outlook prompts you to confirm the deletion. If you record a wizard for deleting a rule, you might want to provide users with a choice instead of the robot performing the deletion for them. Instead of having the robot click **OK**, you could add a bubble to the step instructing the user to confirm the deletion. In this scenario, you would delete the step's core action so that the bubble appears but the robot does not click **OK** for the user.

To delete the selected step's core action:

1. Go to the Window tab of the Flow Pane
2. Right-click the core action and select **Delete**.

The core action is deleted from the step and the detected object is no longer displayed in the Display Pane.

NOTE

While the detected object is no longer displayed, Studio retains the detected object data. In this way, if you later elect to restore the step's core action, the detected object is also restored.

Core Action Properties

From the Properties tab of the Properties Pane, you can customize the following properties (based on core action type, as indicated):

NOTE

The following core action types have no customizable properties... in fact, they have no Properties tab in the Properties Pane at all (but you will find other tab(s) there):

- Detect object
- Wait for block removal

Customizable properties: Click & Hover

Do-it mouse movement

Studio enables you to determine how to move the mouse cursor to its click position on the screen. The mouse movement options are:

- **Shortest:** The robot automatically executes the shortest path from the user's current cursor position to the click position. This is the default setting.
- **Vertical > Horizontal:** The robot moves the cursor in a straight line up/down from the current cursor position and then right/ left to the click position
- **Horizontal > Vertical:** The robot moves the cursor in a straight line right/left from the current cursor position and then up/down to the click position

TIP

This option can be useful when the mouse needs to hover over options in order to display menus.

NOTE

Not applicable to sensors.

Guide-me bubble text

When a wizard is played in **Guide me** mode, a default **Guide-me bubble** appears for each step, anchored to the orange highlight box that surrounds the relevant click position on the user's application. By default, the **Guide-me bubble text** is `Click inside the box`. This text can be customized for each step as follows:

1. From the selected step's core action Properties tab, click the **Guide-me bubble text** dropdown list
2. Select **Custom...**
3. In the field that appears, enter the desired text

The custom text appears for this step when the wizard is played in **Guide me** mode.

NOTE

Not applicable to sensors.

Customizable properties: Read from screen

For detailed information about the customizable properties of the **Read from screen** core action, see the chapter Read from Screen.

Customizable properties: Keyboard

Typing speed

Determines the speed at which the keystrokes will be entered at runtime: automatic (default) or at the speed at which they were entered during recording.

Keyboard language

Determines whether the wizard will accept input in any language or only the language in which the wizard was recorded (default).

NOTE

The **Keyboard** action's Properties tab includes a field that displays the actual sequence of keys typed during recording. This key sequence is not editable. (A step of this type is often replaced during wizard runtime by the value of a variable set using Advanced Commands).

Customizable properties: Wait

The Wait for window close and Wait for object disappearance core actions include a property that allows you set the maximum time the wizard will wait before it proceeds to the next action in the flow.

- By default, the **Max wait** checkbox is set to 5 seconds
- If the **Max wait** checkbox is not ticked and the wizard waits indefinitely

NOTE

Not applicable to the Wait for block removal core action.

Core Action Position

The Kryon RPA Platform uses a proprietary, patented visual detection algorithm to detect objects on the screen and identify the location in which to perform each step's core action.

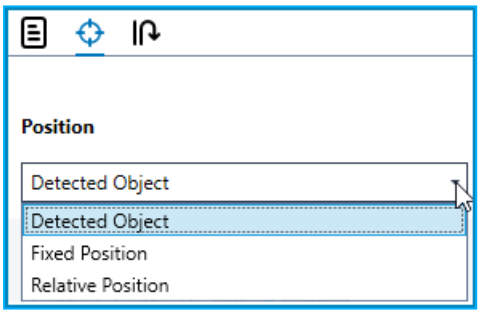
NOTE

For a description of how robots detect objects on the screen, see the Object Detection chapter.

From the Position tab of the Properties Pane, you can optimize object detection, thereby improving performance in terms of speed and accuracy.

Position dropdown list

By default, robots use the Kryon visual detection algorithm identify the object to use for each step. The **Position** dropdown list allows you to disable this default visual detection method (on a per-step basis) and set the robot to click at a fixed or relative distance from the window borders. The following methods are available for determining core action position:



- **Detected Object:** The robot detects the object to click by its visual appearance – an image or text in the object, regardless of its position on the screen. This is the default setting.
- **Fixed Position:** The robot clicks a position based on a fixed distance from a selected corner of the window. The fixed distance is determined by the recorded click position.
- **Relative Position:** The robot clicks a position based on proportion of window size. The proportion is determined by the recorded click position.

Certain customizable properties on the **Position tab** vary based on the selected method of determining core action position. For detailed information, see:

- Core Action Position: Detected Object
- Core Action Position: Fixed Position
- Core Action Position: Relative Position

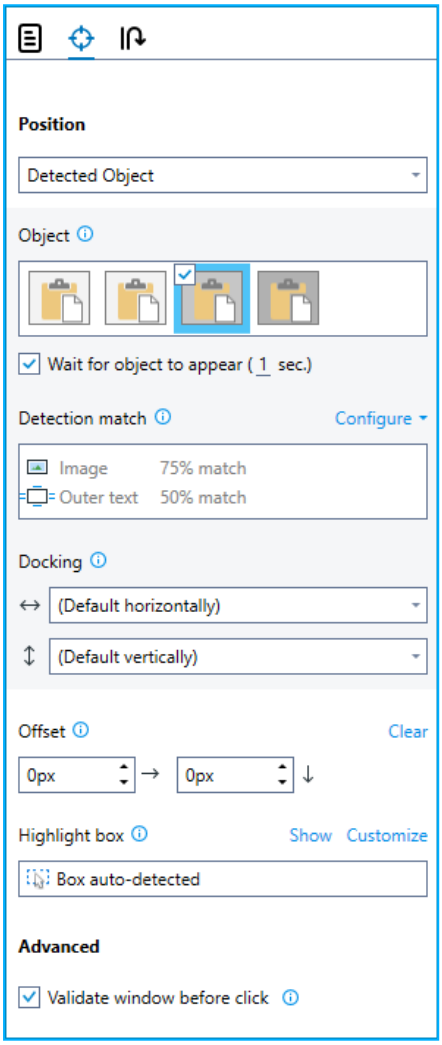
Certain customizable properties are available no matter which method is selected. For detailed information, see:

- Core Action Position: Offset
- Core Action Position: Highlight Box

Core Action Position: Detected Object

The following are the customizable properties specific to the **Detected Object** method of determining core action position:

- Object
- Wait for object to appear
- Detection match
- Docking

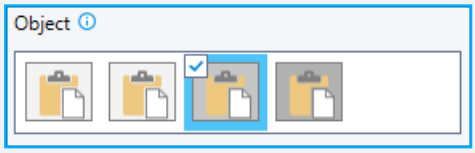


Object

When a wizard is recorded, **Recorder** captures a series of images of the click position. The images are captured at very short intervals, to capture all possible variations in the object appearance. Each image represents a different state of the object to detect, whether it is idle, hovered or clicked. By default, the object state captured immediately after the click is the selected image (called **mouse down**); however, you have the option to select the object variation the robot will use.

EXAMPLE

In Microsoft Outlook 365, the **Paste** button changes appearance depending on whether it is idle, hovered over, or clicked. The selected object below (indicated by a check mark and blue border) is the **mouse down** version of the button – as it appears just after it is clicked:



Selecting an object image variation

To select a variation of the object image:

1. Access the core action's Position tab
2. In the **Position** dropdown list, verify that **Detected Object** is selected
3. In the **Object** section, click on the image variation you want the robot to detect

TIP

Roll your cursor over an image to view a tooltip about the object state that the image represents.

The selected variation will appear with a check mark and a blue border. When the wizard is run, the robot will search for an image on the user's screen that matches the wizard's object image in order to detect the object.

Customizing the detected object

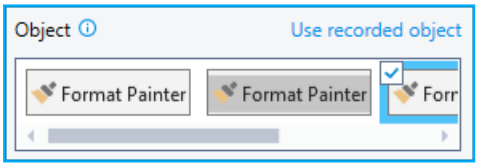
Alternatively, you can customize the detected object completely by selecting a different object in the step's Display Pane.

To customize the selected step's detected object:

1. In the **Display Pane**, the step's current detected object is highlighted by a blue bounding box and label, and contains a blue dot that represents the mouse click position
2. Drag and resize the bounding box around the object you want the robot to detect, and move the blue dot within the box to reposition the mouse click

> **NOTE**
>
> While you are dragging the bounding box, a highlighted square marks the area of the window within which you can select a new detected object.

The image of the detected object you selected and any available variations appear in the **Object** section, along with the **Use recorded object** link.



To remove the customized object and revert to the originally recorded object, click the **Use recorded object** link.

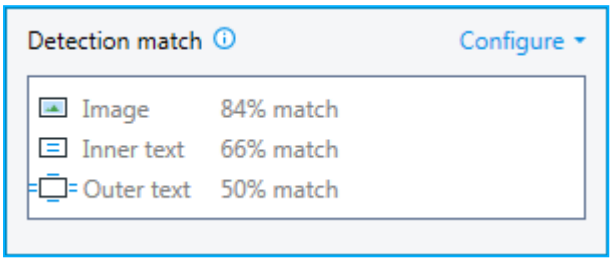
Wait for object to appear

If you want the robot to wait some extra time for the core action's detected object to appear before failing object detection, tick the **Wait for object to appear** checkbox and specify the maximum wait time. If the object does not appear by the time specified, the core action's fallback command will be executed. If the window appears before the maximum time is up, the wizard will immediately continue without waiting the remaining time.

> **TIP**
>
> Utilizing this option can prevent wizard failure in case of slow systems or network/internet connections.

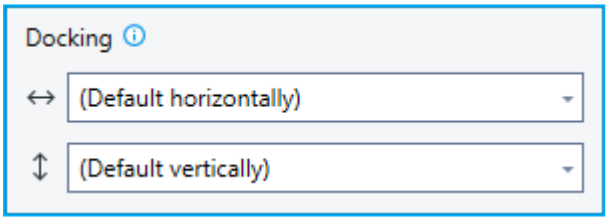
Detection match

When a wizard is run and the robot looks for the click position, it analyzes the screen for image/text that match the captured image/text selected in the Object section. The detection is based on the minimum match threshold for images and text, which appear in the **Detection match** section. For full details on how to customize the default thresholds, see Detection Match Configuration.



Docking

Object docking tells the robot to which window border (or center) the detected object is docked or pinned. Docking focuses the area in which the robot searches for the detected object, so it can be helpful when performance speed or accuracy needs to be improved.



An object can be docked:

- **Horizontally:** The object is located at a fixed distance from the left or right window border or from the window's horizontal center
- **Vertically:** The object is located at a fixed distance from one of the top or bottom window borders or from window's vertical center
- **Both horizontally & vertically**

By default, no specific docking position is selected.

NOTE

If the object is not found within the expected area, the robot searches the entire screen.

TIP

How do you know if an object is docked?

To discover if an object is docked within an application window, go to the application window itself (outside of Studio) and resize the window by dragging it's borders. Pay attention to how resizing the window affects the distance between the detected object and the window's borders and center.

Horizontal docking:

- If the distance between the object and the left border does not change when dragging the side borders, the object is **docked left**
- If the distance between the object and the right border does not change when dragging the side borders, the object is **docked right**
- If the distance between the object and the window's horizontal center does not change when dragging the side borders, the object is **docked to center**

Vertical docking:

- If the distance between the object and the top border does not change when dragging the top/bottom borders, the object is **docked to top**
- If the distance between the object and the bottom border does not change when dragging the top/bottom borders, the object is **docked to bottom**
- If the distance between the object and the window's vertical center does not change when dragging the top/bottom borders, the object is **docked to center**

Editing an object's docking settings

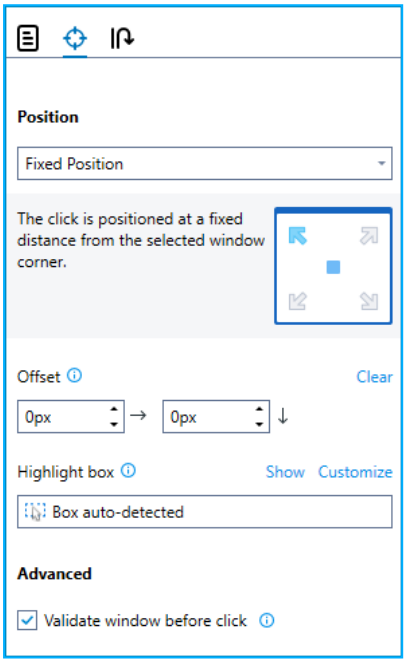
To edit an object's docking settings:

1. Access the core action's Position tab
2. In the **Position** dropdown list, verify that **Detected Object** is selected
3. In the **Docking** section:
   a. Select the desired **horizontal docking** from the horizontal docking dropdown list ↔
   b. Select the desired **vertical docking** from the vertical docking dropdown list ↕

Core Action Position: Fixed Position

When you use the **Fixed Position** method of determining core action position, the robot clicks a point on the screen that is a fixed number of pixels (both horizontally and vertically) from a selected corner of the window (let's call this the ***anchor corner***). The distance from the anchor corner is determined by the location that was clicked when the wizard was recorded.

The Position tab of the Properties Pane appears as follows when the **Fixed Position** method is selected from the position dropdown list:



Changing the anchor corner

The selected (i.e., blue) arrow in the image points to the anchor corner. The upper-left corner is selected by default.

To change the anchor corner:

1. Click on the arrow that points to the anchor corner you wish to use

The arrow you clicked on is selected and will be used as the anchor corner when the wizard is run.

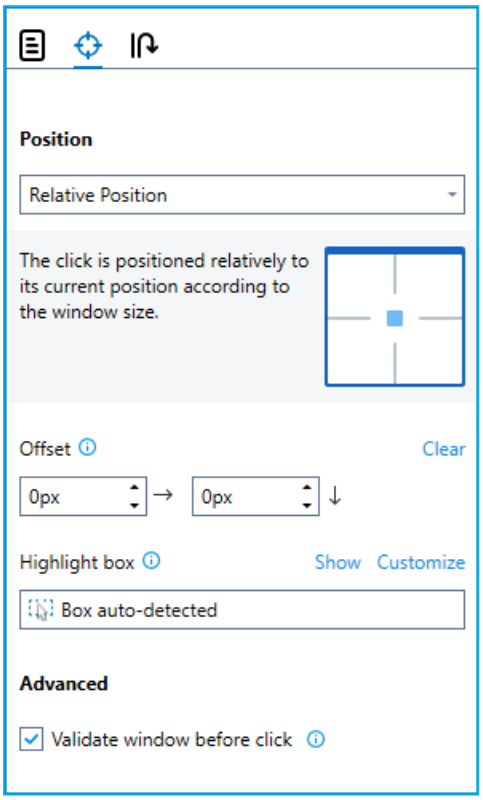
CAUTION

When using the **Fixed Position** method, test that the robot is able to detect the click position when the window is resized or when a different environment is used. Otherwise, the wizard might either click in the wrong position or show an error message.

Core Action Position: Relative Position

When you use the **Relative Position** method of determining core action position, the robot clicks a position based on proportion of window size. The proportion is determined by the recorded click position. There are no customizable properties specific to the **Relative Position** method.

The Position tab of the Properties Pane appears as follows when the **Relative Position** method is selected from the position dropdown list:



EXAMPLE

Using Relative Position

I record a wizard on a window sized 1280 pixels wide x 1024 pixels high. During the recording, I click on an object located 10% to the right and 20% below the center of the window – i.e., 128 pixels to the right and 205 pixels below the center.

If I use the **Relative Position** method of determining core action position, when I run the wizard on a window sized 1024 pixels x 768 pixels, the click will still occur 10% the right and 20% below the center of the window – now 102 pixels to the right and 154 pixels below the center.

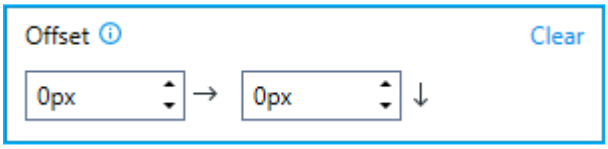
Core Action Position: Offset

> **NOTE**
>
> This **Offset** section is not applicable to the Read from screen core action. Instead, see **Field label position (field offset)**.

At times, a wizard may need to interact with an object that is not unique (such as an empty text field or dropdown arrow) or one whose appearance may change (such as a dropdown list). Despite the fact that this is the object the robot needs to act on, the recorded click or keystroke on this object doesn't provide enough information to allow for accurate object detection.

In such cases, edit the step using an **offset**. This instructs the robot to detect a more easily identifiable object during runtime, then to perform the core action a fixed number of pixels (horizontally and vertically) away from it.

The **offset settings** for current step's core action appear in the Position tab of the Properties Pane. By default, offset is set to 0 pixels (both horizontally and vertically):



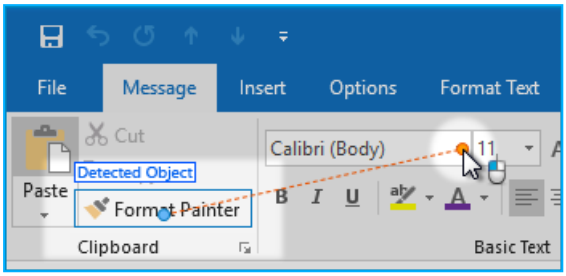
Adjusting offset

To adjust the offset:

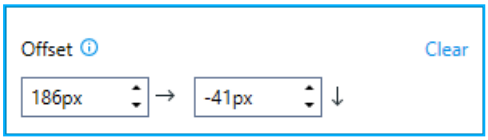
1. In the Display Pane, hover your mouse over the blue dot that identifies the detected object until it turns into a 4-way arrow
2. Drag the dot to the object you want the robot to detect instead

   The blue dot and the **Detected Object** label move to the new location
3. Hold down the <SHIFT> key and drag the dot again to the object you want the robot to act on

4. Results:
   - An orange dot is created (and appears over the object the robot will act on)
   - The icon representing the click action has moved with the orange dot



   - The **offset settings** as appear in the Position tab show the position of the orange dot relative to the blue dot



Horizontal setting:

- Positive number → orange dot is right of the blue dot (i.e., click position is to the right of the detected object)
- Negative number → orange dot is to the left of the blue dot (i.e., click position is to the left of the detected object)

Vertical setting:

- Positive number → orange dot is below the blue dot (i.e., click position is below the detected object)
- Negative number → orange dot is above the blue dot (i.e., click position is above the detected object)

TIPS

- You can directly adjust (or fine tune) offset position by manually changing the offset setting numbers in the Position tab
- After adjusting offset, be sure that the highlight box is properly placed around the click target (and adjust as necessary)

Removing offset

To remove the offset:

1. Click the **Clear** link in the **Offset** section of the Position tab

Results:

- The offset settings in the Position tab are reset to 0 px horizontal, 0 px vertical
- The orange dot representing the offset is removed from the Display Pane

CAUTION

Check the detected object!

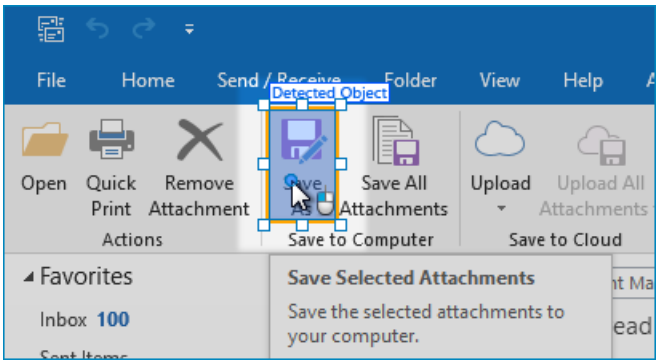
When you clear the offset, the orange dot is removed, but any change you made to the position of the detected object (the blue dot) remains. To revert to using the originally recorded object, click the **Use recorded object** link in the **Object** section of the Position tab.

Core Action Position: Highlight Box

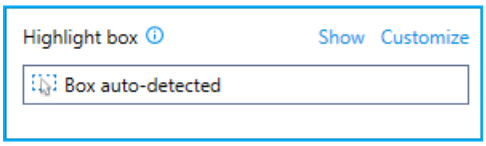
NOTE

Applicable to **Click** and **Hover** core actions only.

When users play a wizard in Guide Me mode, an orange highlight box appears around each object that the user is instructed to click. By default, the highlight box location is determined by the click position, and automatically adjusts itself to the object's borders.



The **highlight box** settings for current step's core action appear and can be edited in the Position tab of the Properties Pane:



Viewing highlight box location

To view the current location of the highlight box:

1. Go the Position tab of the Properties Pane
2. In the **Highlight box** section, click **Show**

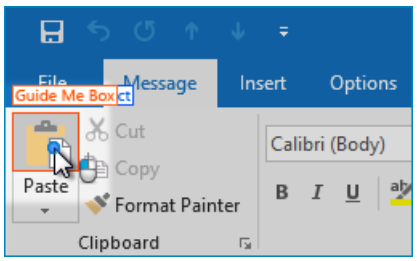
In the Display Pane, the highlight box appears temporarily at the location in which it will appear to the end user.

Adjusting highlight box location

To adjust the location of the selected step's highlight box:

1. Go the Position tab of the Properties Pane
2. In the **Highlight box** section, click **Customize**

   In the Display Pane, the highlight box appears in its current location with a **Guide Me Box** label:



3. Drag and resize the box as necessary

TIP

Adjusting the location of the highlight box is especially relevant for steps in which the click position has been adjusted by an offset, in which case the highlight box might need to be changed or expanded.

Restoring the default highlight box location

To restore the selected step's default highlight box location:

1. Go the Position tab of the Properties Pane
2. In the **Highlight box** section, click **Delete (use default)**

The highlight box is restored to its default location.

Detection Match Configuration

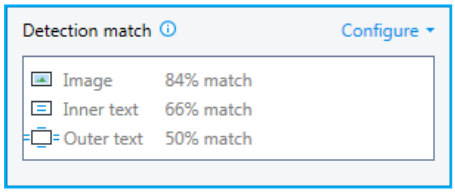
The **Detection match** configuration menu contains settings that allow you to adjust the flexibility of the robot's detection capabilities (in other words, to be more or less accepting of images that differ from the recorded image). This menu also allows you to view the results of the latest wizard run and whether it succeeded or failed. The menu includes the following tabs:

- Detection Method: Allows you to determine the image/text detection method, the detection match threshold(s), and the size of the search area
- Detection Behavior: Allows you to include color inversions, limit detection to a specific color and font weight, and optimize detection when screen blocking is activated
- Last Run Results: Shows you the match percentages detected the last time the wizard or sensor was run from Kryon Studio (useful for optimization and debugging purposes)

Accessing the Detection match configuration menu

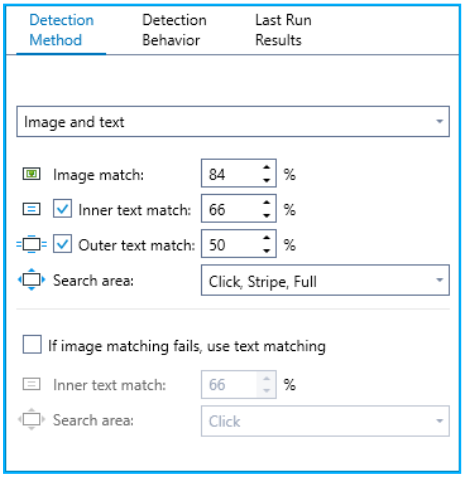
To access the **Detection match** configuration menu:

1. Access the core action's Position tab
2. In the **Position** dropdown list, verify that **Detected Object** is selected
3. At the top right of the **Detection match** area, click **Configure**



Detection Method

The **Detection Method** tab allows you to determine the image/text detection method, the detection match threshold, and the size of the search area.



Method

- **Image and text:**The robot attempts to identify the click position using: (1) the detected object image; (2) the text inside the image; and (3) the text around the image (default setting)
- **Text only:** The robot attempts to identify the click position using only the text inside the detected object image

NOTE

If all else fails...

When the **image and text** method is selected, you can elect to use the **text only** method in case of image match failure. To do so, tick the **If image matching fails, use text matching** checkbox, and configure the desired settings for this alternate method: inner text match and search area.

Match threshold(s)

Match threshold represents the accuracy the robot will require when matching the detected object at runtime to the object detected during recording.

- The higher the match threshold percentage, the closer the match required
- If the specified match threshold(s) are not met, the step will fail object detection

Once you select a detection method, you can set an individual **match threshold** for each element used by the robot in identifying click position.

Image and text

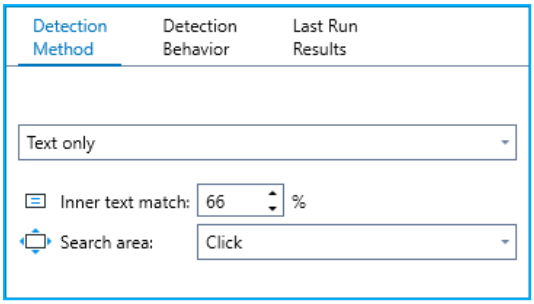
When the **image and text** method is selected, you can configure the following **match thresholds:**

- **Image match:** The visual image of the object and of any text in it
  - Default threshold = 84%
  - If the image is small, increasing the default percentage is recommended

> **CAUTION**
>
> Decreasing the image match threshold below 70% is not recommended, and in some cases restricted. If the minimum threshold is less than 70%, the robot might click the wrong area of the screen, in which case you must increase the percentage so that the wizard will run properly.
>
> If you use 70% or above and the robot is still unable to detect the correct click position, adding a fallback is recommended.

- **Inner text match:** OCR (image-to-text conversion) for text within the detected object (e.g., a sentence on the screen or text inside a button, such as `Continue`)
  - Default threshold = approximately 60% to 80%, depending on text length
  - You can elect not to use inner text match by clearing its checkbox
  - If the clicked image does not contain text, the inner text match checkbox will be automatically cleared
- **Outer text match:** OCR for text surrounding the detected object (used to confirm the location of the object when there are similar images nearby)
  - Default threshold = 50%
  - You can elect not to use outer text match by clearing its checkbox
  - The outer text match threshold affects only a handful of scenarios and, in most cases, won't be used
  - If outer text could be dynamic, clearing this checkbox is recommend

Text only



- **Inner text match:** OCR (image-to-text conversion) for text within the detected object (e.g., a sentence on the screen or text inside a button, such as `Continue`)
  - Default threshold = approximately 60% to 80%, depending on text length

Search area

The **Search area** setting instructs the robot which area(s) of the screen to include when searching for the detected object:

- **Click:** Search is limited to the click area only
- **Click, Stripe:** First the click area is searched. If the click search fails, the search area is expanded to the stripe in which the object is docked
  - **Stripe search** is applicable only if either vertical or horizontal docking is set. If both or neither direction is set, stripe search is not applicable.
- **Click, Stripe, Full:** First the click area is searched. If the click search fails, the stripe area is searched. If both the click and stripe search fail, the search area is expanded to the entire screen.

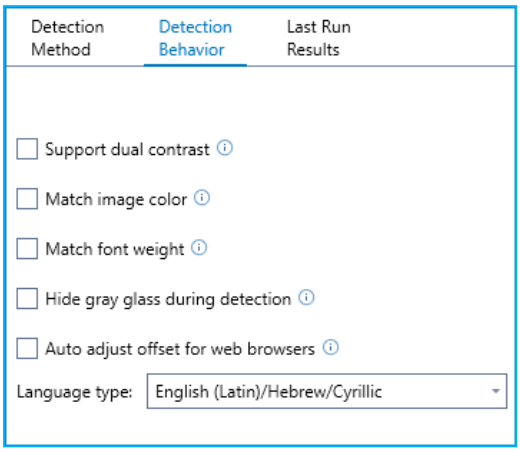
TIP

Smaller search = better performance

For the best performance, use the smallest search area that consistently produces accurate results. If the object location never (or rarely) changes, limiting the search to the click area improves robot speed. If the object location is expected to change, you can expand the search area to include the entire window to maximize the robot's opportunity to detect it.

Detection Behavior

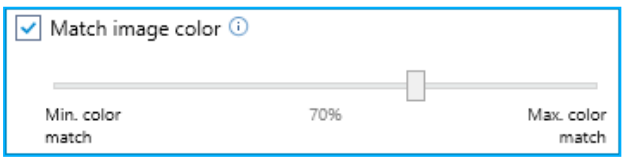
The **Detection Behavior** tab allows you to customize various additional settings to maximize object detection accuracy.



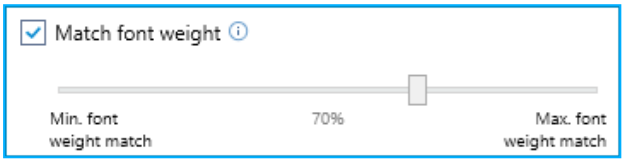
- **Support dual contrast:** Ensures detection if an object's colors are inverted

> **EXAMPLE**
>
> This tab, when idle (that is, not clicked), displays dark text on light background: Contacts
>
> The tab's colors get inverted when clicked, resulting in light text on dark background: Contacts
>
> The support dual contrast checkbox allows the robot to detect the object in either idle/clicked state.

- **Match image color**: Ensures detection of an object only with the same image color as that recorded. Allows you to prevent detection of objects with similar, but not identical colors, and to exclude minor color variations.
  - When this setting is enabled, a color match percentage is also specified (default = 70%). The higher the percentage, the closer the match required. Adjust this setting to increase or decrease the required color match between the recorded object and the object detected at wizard runtime.



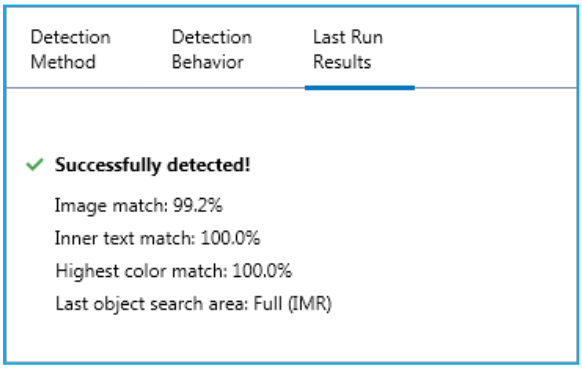
- **Match font weight:** Ensures detection of text with the same font weight as that recorded. Allows you to prevent detection of objects whose font weight does not match that of the recorded font (e.g., the robot will detect an object when its text is bold, but ignore it if it is not).
  - When this setting is enabled, a font-weight match percentage is also specified (default = 70%). Adjust this setting to increase or decrease the required font-weight match between the recorded object and the object detected at wizard run time.



- **Hide gray glass during detection:** For rare cases in which the gray glass effect used for screen blocking might interfere with object detection, this setting enables you to temporarily disable the effect only for the duration of object detection
  - Relevant only for steps in which screen blocking is activated
- **Auto adjust offset for web browsers:** Adjusts offset during runtime to account for the web browser's zoom-level
- **Language type:** Identifies the language type of text in the recorded object to improve accuracy of the text match at runtime

Last Run Results

The **Last Run Results** tab displays the match percentages detected the last time the wizard was played from Kryon Studio. This data can be used for optimization and debugging purposes by helping you determine the desired match percentages and fix detection errors.



Core Action Fallbacks

The fallback events for core actions are:

- Window not found
- Object not found

The default fallback command for both of these events is `Raise error`.

To change the default fallback command for either or both of these events, follow the instructions in Applying Fallbacks.

CHAPTER 10: Read from Screen

In this chapter:

Using Read from Screen 143
Read from Screen: Position 144
Read from Screen: Value Type 148
Read from Screen: Variable 156

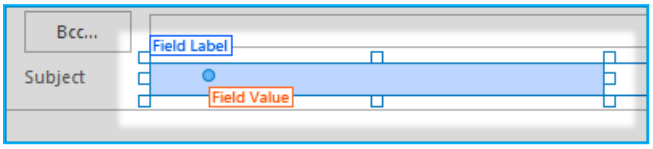
Using Read from Screen

Using **Read from screen** and maximizing its accuracy requires providing the robot with a specific set of instructions:

1. Changing the step's core action to **Read from screen**.
2. Identifying a ***static*** text near the read location that the robot uses to identify the location from which it should read. This is called the field label.
3. Identifying a specific location (field) on the screen from which the robot should read the value. This location is identified by the field box.
4. Identifying the type of value the robot should read. This is called the value type.
5. Specifying the variable into which the robot should store the value it has read. The wizard can then use this value later in one or more Advanced Commands.

Read from Screen: Position

Once you change a step's core action to **Read from screen**, in the Display Pane, the blue text reading **Detected Object** changes to read **Field Label**, and an additional orange text reading **Field Value** appears.



Field label & field value

Two locations must be set in order for the robot to read from the correct position on the screen:

- The **field label** is the ***static*** text near the read position that the robot uses to identify the location from which it should read; it is indicated by a blue dot and the **field label box** (a blue bounding box)
- The **field value** represents the actual position from which the robot should read; it is indicated by the **field box** (an orange bounding box that is initially hidden)

NOTE

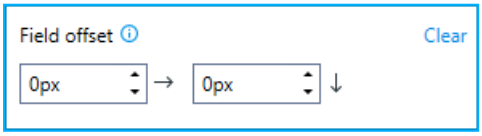
Why can't I see the field box?

By default, the orange **field box** is hidden because it is in the same location as the blue **field label box**. If and when you customize the location of the **field box**, it will become visible.

Field label position (field offset)

The first location to be set for **Read from screen** is the position of the **field label**. By default, the **field label box** and the **field box** are in the same location, so you must adjust their relative locations using **field offset**.

The **field offset settings** appear in the Position tab of the Properties Pane. By default, field offset is set to 0 pixels (both horizontally and vertically):



Adjusting field offset

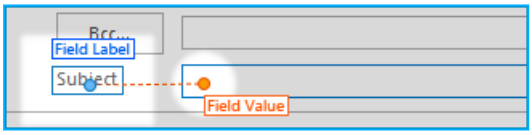
To adjust field offset:

1. In the Display Pane, hover your mouse over the blue dot that identifies the **field label** until it turns into a 4-way arrow
2. Drag the dot to the text you want the robot to use as the **field label**

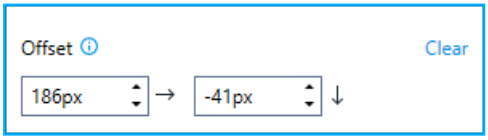
   The blue dot, the **field label** box, and the text reading **Field Label** move to the new location

> **NOTE**
>
> While you are dragging the dot, you may notice that the words **Field Value** move as well. Not to worry, we will adjust this next.

3. Hold down the <SHIFT> key and drag the dot again to the location from which you want the robot to read
4. Results:
   - An orange dot is created (and appears over the location from which the robot will read)
   - The text reading **Field Value** has moved with the orange dot



   - The **field offset settings** as appear in the Position tab show the position of the orange dot relative to the blue dot



Horizontal setting:

- Positive number → orange dot is right of the blue dot (i.e., field value is to the right of the field label)
- Negative number → orange dot is to the left of the blue dot (i.e., field value is to the left of the field label)

Vertical setting:

- Positive number → orange dot is below the blue dot (i.e., field value is below the field label)
- Negative number → orange dot is above the blue dot (i.e., field value is above the field label)

TIPS

- You can directly adjust (or fine tune) the field offset position by manually changing the field offset setting numbers in the Position tab

Removing field offset

To remove field offset:

1. Click the **Clear** link in the **Field offset** section of the Position tab

Results:

- The field offset settings in the Position tab are reset to 0 px horizontal, 0 px vertical
- The orange dot (representing the **field value** location) is removed from the Display Pane

Field value position (field box)

The location of the **field box** is automatically detected and adjusted when you follow the **field offset** process as described above. You can view and edit this location as required.

Viewing field box location

To view the current location of the **field box**:

1. Go the Position tab of the Properties Pane
2. In the **Field box** section, click **Show**

The orange **field box** appears temporarily in the Display Pane.

Adjusting field box location

To adjust the location of the **field box**:

1. Go the Position tab of the Properties Pane
2. In the **Field box** section, click **Customize**

   The orange **field box** appears in the Display Pane.

3. Drag and resize the box as necessary

Restoring the default field box location

To restore the default **field box** location:

1. Go the Position tab of the Properties Pane
2. In the **Field box** section, click **Delete (use default)**

The **field box** is restored to its default location.

Additional Read from screen position options

The additional options and settings available in the Read from screen Position tab are applicable to the position of the **field label**. The usage of these options is the same as that for other core actions – including the selected method in the position dropdown list, object, wait for object to appear, detection match, docking, and fixed position anchor corner).

Read from Screen: Value Type

A Kryon Robot reads specific types of values from the fields on the screen (e.g., identifying numbers/dates, recognizing free-form texts, determining whether a field or checkbox is empty, etc.). For each **Read from screen** action, the robot reads one type of value.

Selecting the Read from screen value type

To select the value type for the selected **Read from screen** core action:

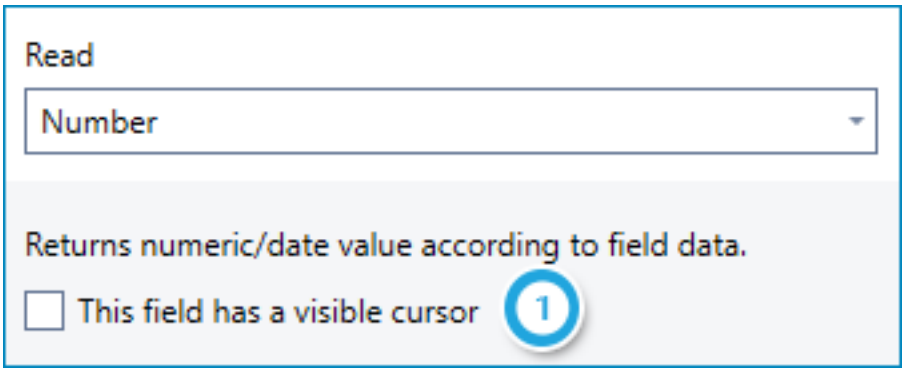
1. Go the Properties tab of the Properties Pane
2. Select the value type from the **Read** dropdown list

The additional properties to be specified vary based on the value type selected.

Number

The robot uses Kryon's visual technology to read a number or date from the specified field, recognizes it using optical character recognition (OCR), and places it into a variable.

- Accepted input:
  - The numbers `0 - 9`
  - The characters `\ + - / . ,`
- If the first or last character is not an accepted input (e.g., currency sign), the wizard removes it and places the resulting number in the variable
  - For example: `€12.3` or `12.3%` becomes `12.3`
- If any character other than the first or last is not an accepted input, the variable is returned as empty



1. Tick this checkbox if the field to be read contains a visible or flashing cursor when selected

Text: Native

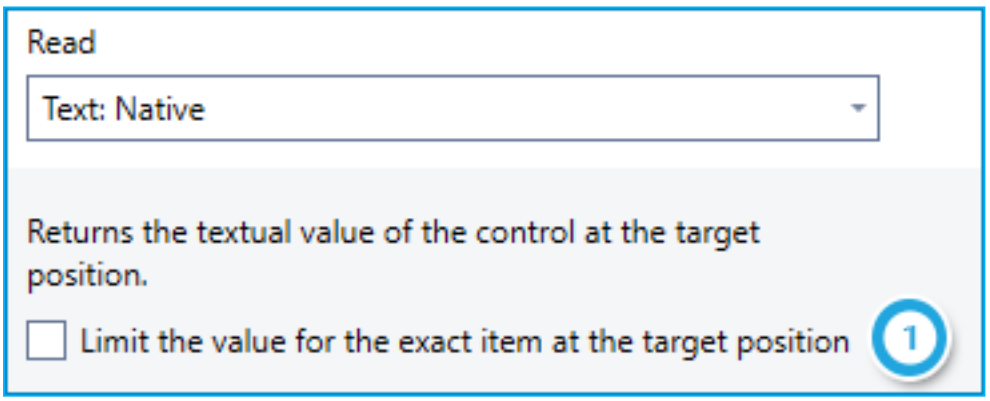
The robot reads any text from the specified field and places it into a variable.

> **NOTE**
>
> **Text: Native** is the recommended value type for free-form text when it is supported by the target application. When it isn't supported, Text: OCR can be used.

Support for the **Text: Native** value type depends on the behind-the-scenes technology of the target application/web browser. In order to work, it requires that the target application expose its memory processes. Some applications and web browsers do, while others don't. **This must be tested during the content development process.**

Known technologies that do **NOT** support **Text: Native**:

- Java
- Microsoft Silverlight
- Adobe AIR
- WebKit
- Google Chrome
- Mozilla Firefox



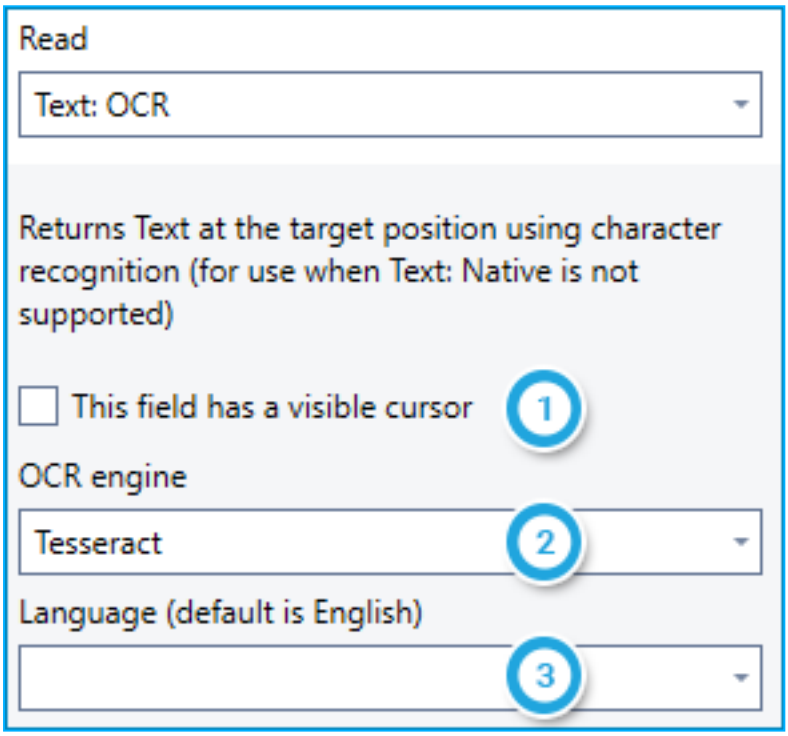
1. By default, if the text field at the target location contains a table or a tree, the entire table or tree will be returned in the variable. Tick this checkbox to limit the returned text to just the cell or line at which the field box is located.

Text: OCR

The robot uses visual technology to read any text from the specified field, recognizes it using OCR, and places it into a variable.

NOTE

Text: OCR is the value type to use for free-form text when Text: Native is not supported by the target application. For texts that contain only numbers or texts for which all possible values are known in advance, the Number and Text: Predefined values value types, respectively, will likely produce more accurate results.



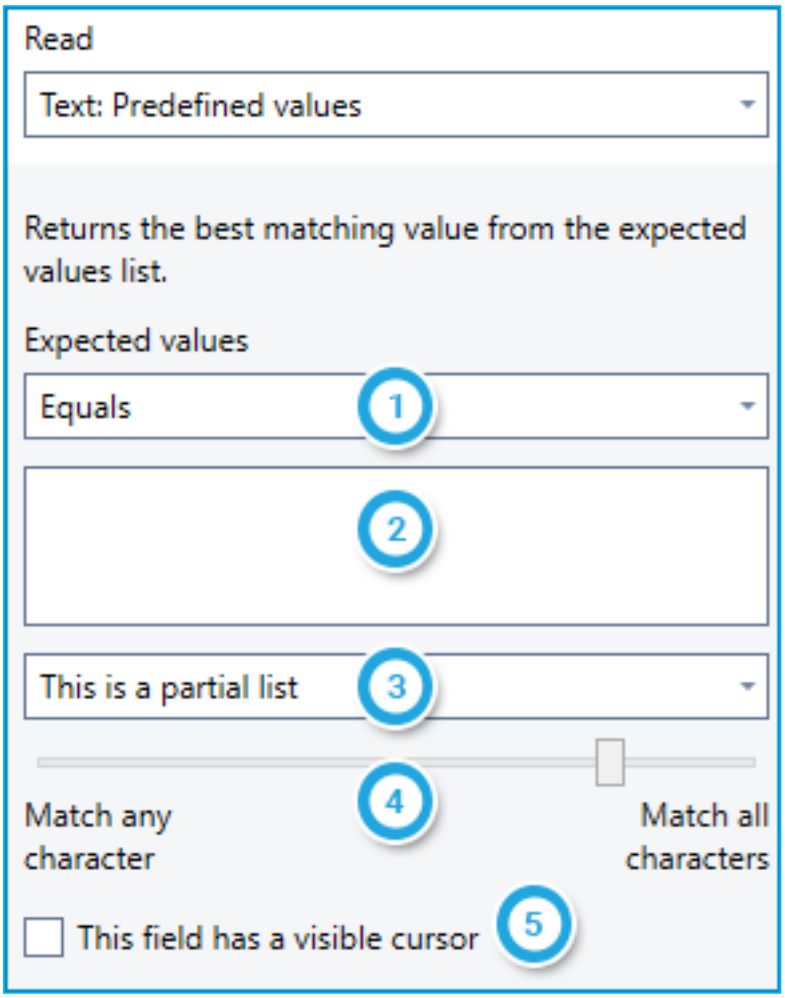
1. Tick this checkbox if the field to be read contains a visible or flashing cursor when selected
2. The current OCR engine is Tesseract
3. Select the language of the text to be recognized

Text: Predefined values

The robot uses visual technology to read any text from the specified field, uses OCR to match it to a value within a predefined list, then returns the matched value into a variable.

NOTE

The **Text: Predefined values** option may be used to provide better accuracy than Text: OCR when all possible values for the specified field are known in advance and Text: Native is not supported.



1 **Condition dropdown:** In the list of predefined values 2, you can enter either all or part of each possible value. In this dropdown list, select whether you want the wizard to match and return:

- the whole value (`Equals`); ***or***
- a part of each value (`Starts with`, `Ends with`, `Contains`)

EXAMPLE

Let's say –

- your expected values include `South Carolina`, `Georgia`, and `Alabama`
- the value appearing on screen is `North Carolina`
  - If you select `Equals` from the condition dropdown list → no exact match (see 3 for an explanation of the value that will be returned in the variable)
  - If you select `Ends with` from the condition dropdown list → the value returned in the variable will be `Carolina` (note that the value returned will **NOT** be `North Carolina`)

2 Expected Values:

Enter the list of possible values with which the robot will attempt to match the value read from the screen – one value per line

- Acceptable values include any combination of:
  - individual items using Latin, Cyrillic, or Hebrew alphabets; ***and***
  - variable(s) whose values are predefined lists

3 Full/partial list dropdown:

- If, in the list of predefined values (2), you provide **ALL** possible values, select `This is the full list`
  - The value returned in the variable will **ALWAYS** be the closest match from the list (whether or not there is an exact match)
- If you provide only **SOME** of the possible values, select `This is a partial list`. The **match threshold slider** (4) will then appear.
  - The value returned in the variable will the closest value from the list **IF** it is within the match threshold you specify. If the closest value is not within the match threshold, the returned variable will be empty.

4 **Match threshold slider** (appears **ONLY** if `This is a partial list` is selected in 3):

Specify the accuracy the robot should require when comparing the value read from the screen to the list of predefined values 2

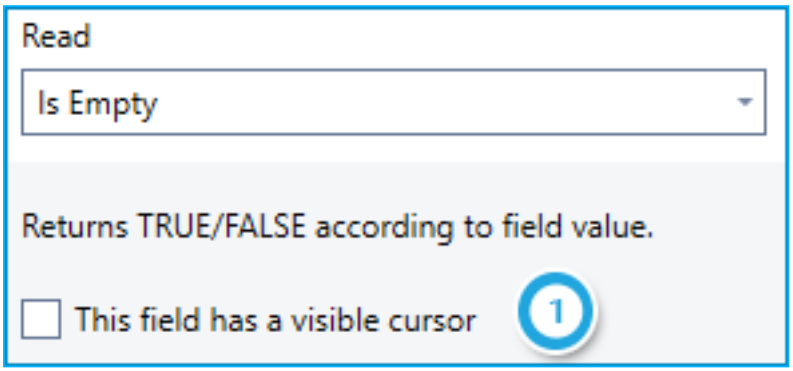
- The higher the match threshold percentage, the closer the match required
- If the closest value in the list does not meet the specified match threshold, the returned variable will be empty

5 Tick this checkbox if the field to be read contains a visible or flashing cursor when selected

Is empty

The robot determines if the specified field is blank or contains a value of any type, and returns `TRUE` or `FALSE` into the variable as follows:

- Empty field = `TRUE`
- Non-empty field = `FALSE`

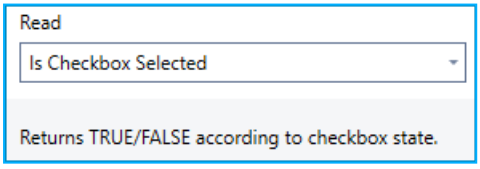


1 Tick this checkbox if the field to be read contains a visible or flashing cursor when selected

Is checkbox selected

The robot determines if the specified checkbox is ticked or cleared, and returns `TRUE` or `FALSE` into the variable as follows:

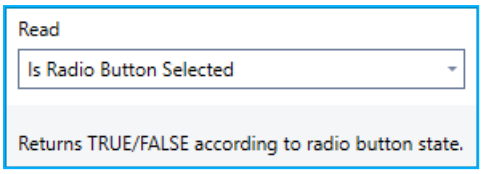
- Ticked checkbox = `TRUE`
- Cleared checkbox = `FALSE`



Is radio button selected

The robot determines if the detected radio button is selected or cleared, and returns `TRUE` or `FALSE` into the variable as follows:

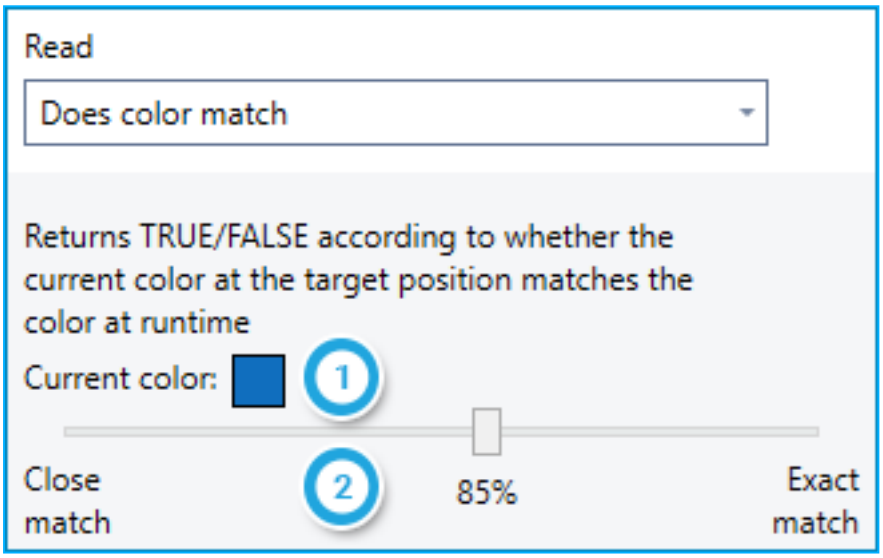
- Selected radio button = `TRUE`
- Cleared radio button = `FALSE`



Does color match

The robot determines if the color at the field value location at wizard runtime matches the designated color, and returns `TRUE` or `FALSE` into the variable as follows:

- Color matches within the specified threshold = `TRUE`
- Color does **NOT** match within specified threshold = `FALSE`



1. Displays the color at the current field value location. This color is determined by:
   - the color that existed at the field value location when the wizard was recorded; ***or***
   - the color at the field value location after editing (if the field value location was moved)

   If the field value location is moved, the color displayed here changes accordingly.
2. Set the slider to the precision of the color match required in order for the robot to return a value of `TRUE` :
   - The higher the percentage, the closer the color match required
   - Default threshold = 85%

Read from Screen: Variable

Once the robot reads a value from the screen, it stores that value in a variable which can then be used in Advanced Commands.

Setting the Read from screen variable

To specify the variable in which the **Read from screen** value should be stored:

1. Go the Properties tab of the Properties Pane
2. In the **Return result in variable** field, type a name for the variable or select an existing variable from the dropdown list

CHAPTER 11: Blocks

Attended/Hybrid Only

In this chapter:

Blocks: Overview 158

Adding & Deleting Blocks 160

Block Properties 161

More About Object Blocks 164

Block Duration 166

Blocks: Overview

A block is an optional step action that can be used to sense when a user performs particular action (the "triggering action") on the target application, such as clicking on a certain button or entering specific keyboard combination. When the block detects this action, it performs the logical sequence defined by the RPA developer.

The following are the major considerations when using a block:

- Whether to allow/deny/store the triggering action (Behavior)
- What the wizard should do next (When clicked/pressed)
- If/how/when the block is removed (Block duration)

Block types

You can choose from various types of blocks – varying by the triggering action and resulting screen area/key combination blocked:

- **Screen block:** When the user clicks the mouse in the target application, the entire screen is blocked – regardless of the applications currently open or active on the screen
- **Window block:** When the user clicks the mouse in the target application, the active application window is blocked
- **Object block:** When the user clicks the mouse over a predefined area of the target application, that area is blocked
    - Certain properties (namely, block position and fallbacks) are relevant only to **object blocks**. For additional information, see More About Object Blocks.
- **Key block:** When the user types a predefined key combination affecting the target application, the resulting action is blocked

Why use a block?

Blocks have a wide array of uses, depending on organizational needs. These uses can be classified into two main purposes:

- To ensure user compliance with defined processes
- To augment application functionality

EXAMPLE

After filling in a form, the user clicks **OK**. To help ensure that form is completed correctly before it is submitted, you can use a block to:

- Sense when the user clicks **OK**
- Hold the click before allowing it to go through to the application (while the wizard validates that the form is completed correctly)
- Based on the result of the validation, decide whether to: (1) release the click to the application; or (2) deny it and prompt the user to fix any incorrect data

Development

In Studio, the RPA developer would:

1. Record the form window in a sensor/wizard
2. Apply a block to the form's **OK** button to sense the user's click
3. Apply a read from screen validation on the form fields

Wizard runtime

In the user's environment:

1. The user completes the form and clicks **OK** to submit it
2. When the user's click on **OK** is sensed, the block stores the click to memory, not letting it go through to the application
3. The wizard/sensor runs a read from screen validation to ensure that the form was completed correctly
4. If the form was completed correctly, the wizard releases the user's click and allows it to go through to the application (i.e., resulting in the form being submitted)
5. If the form was not completed correctly, the wizard prompts the user to fix it, then restarts the block process from step 1

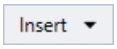
Adding & Deleting Blocks

When a block is added to a step, it always appears immediately after step start.

- Blocks can be added and removed, but cannot be moved within the step's flow
- Multiple blocks can be added to a single step

Adding a block

To add a block to the selected step:

1. Go to the Flow tab of the Flow Pane
2. Click the **Insert** button 

3. Select the desired block type from the dropdown list
4. For an object block, move your mouse over the Display Plane and click in the location of the block you want to create
    - You can customize the precise location and size of an object block after it has been created. See Object block position.

The block appears in the Display Pane and in the Flow tab

Deleting a Block

To delete a block from the selected step, do one of the following:

- From the Flow tab of the Flow Pane, select the block and click

- From the Flow tab of the Flow Pane, right-click the block and select **Delete** from the menu
- From the Display Pane, right-click the block and select **Delete** from the menu

Block Properties

The following block properties can be customized from the Properties tab of the Properties Pane:

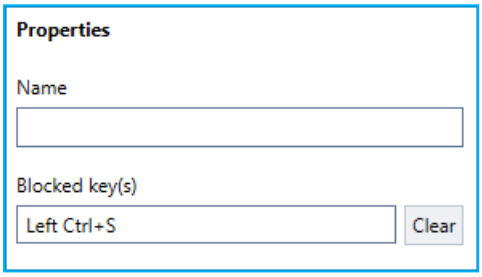
Name

Enter a block **Name** to serve as a reference for the RPA developer in Kryon Studio. The **Name** is not visible to end users in Kryon Robot.

Blocked key(s)

Enter the key or key combination for the block to intercept (e.g., `CTRL+S`)

- Click the **Clear** button to clear your choice of blocked key(s)



> **NOTE**
>
> The **Blocked keys(s)** property is available for (and applicable to) key blocks only.

Behavior

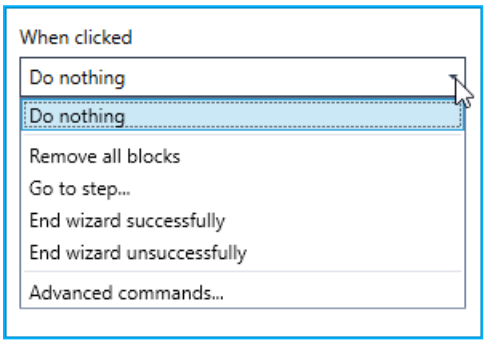
Select a radio button in the **Behavior** section to determine what the block will do when it intercepts the click/key combination:

- **Deny user action:** Denies the click/key combination from getting through to the application; it is dismissed as if it did not occur
  - For screen and window blocks, there are additional options that allow you to customize what the user sees when the click is denied

- **Store user action:** Stores the click/key combination to memory in order to allow or deny it at a later time, according to a predefined logic
- **Allow user action:** Release the click/key combination so that it gets through to the application

When clicked/pressed

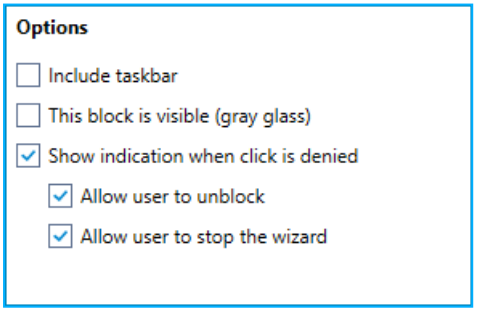
Select an option from this dropdown list to determine how the wizard should proceed once the user action is either denied, stored, or allowed. The default option is **Do nothing**.



NOTE

This property is called **When pressed** for key blocks and **When clicked** for all other block types.

Additional options



Include taskbar

Elect whether to include the Windows task bar in the blocked area of the screen

NOTE

The **include taskbar** option is available for screen blocks only.

Denied click options

Select one or more of these additional options when the behavior of a screen or window block is set to deny the user action –

- **This block is visible (gray glass):** The user sees the entire block area as "grayed out" when the wizard is run
- **Show indication when click is denied:** When the click is intercepted and denied, the user receives a notification. You also have the option to allow the user to:
  - Remove the block
  - Stop the wizard

NOTE

Denied click options are available for screen and window blocks only.

More About Object Blocks

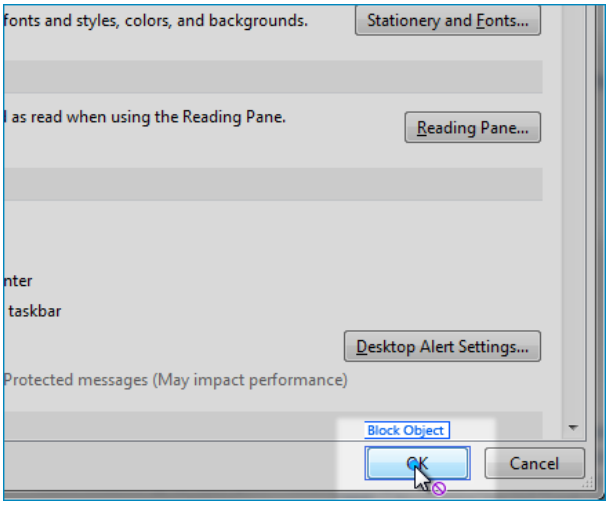
The following properties (and, therefore, tabs in the Properties Pane) are relevant only to object blocks:

- Position
- Fallbacks

Object block position

Object block display

In Studio, the position of an **object block** is visible in the Display Pane, indicated by a blue dot and the purple block icon :



At wizard runtime, object blocks are:

- invisible when run from Kryon Robot
- indicated by a purple bounding box when run from Kryon Studio

Object blocks & object detection

Object block position is determined using object detection. Therefore, an object block's position properties are essentially identical to core action position properties – including the ability to apply an offset.

NOTE

In the context of object blocks, the detected object is called the **Block Object**.

Viewing & customizing block location

The precise size and location of an object block is automatically detected when you create the block and/or apply an offset. You can view and edit this location as required.

To view the current location of the block:

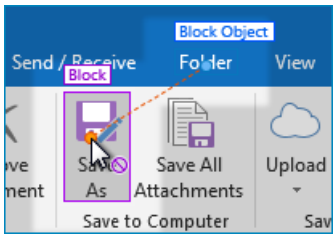
1. Go the Position tab of the Properties Pane
2. In the **Surrounding box** section, click **Show**

The **surrounding box** appears temporarily (in orange) in the Display Pane.

To adjust the location of the block:

1. Go the Position tab of the Properties Pane
2. In the **Surrounding box** section, click **Customize**

   The purple surrounding box appears in the Display Pane.



3. Drag and resize the box as necessary

Restoring the default block location

To restore the default block location:

1. Go the Position tab of the Properties Pane
2. In the **Surrounding box** section, click **Delete (use default)**

The block is restored to its default location.

Object block fallbacks

The fallback event for an object block is:

- **Object not found**

The default fallback command for this event is `Retry`.

To change the default fallback command, follow the instructions in Applying Fallbacks.

Block Duration

Once a block is added, it remains in effect for the remainder of the wizard run until it is removed. In other words, the block remains in place even after the end of the step in which it was added.

Removing blocks

Blocks are removed through a command added to the wizard by the RPA developer. There are several possible locations for adding the **Remove all blocks** command:

- When clicked/pressed block property
- Step end action
- Fallback command
- Advanced commands

CHAPTER 12: Bubbles

Attended/Hybrid Only

In this chapter:

Bubbles: Overview 168
Bubbles & Mouse/Keyboard Control 169
Working with Bubbles 171
Bubble Types 176
Bubbles: General Properties 180
Bubbles: Show/Hide Properties 182
Bubble Buttons 186

Bubbles: Overview

Bubbles are textual descriptions or instructions that appear on the user's screen when running a wizard. Bubbles are used to provide information to the user or to prompt the user to perform an action.

Bubbles can have one or more of the following functions:

- Instructing users to perform actions that the robot will not perform for them, such as typing user-specific information or deleting data
- Informing users about what is happening in a step, and providing important information or tips
- Providing decision points to alternative scenarios in a wizard flow
- Granting control to the user over the mouse and keyboard

The properties of each bubble include configurable buttons, display settings, narration, and the bubble's position on the screen. Bubble properties are managed from the bubble's Properties Pane.

Bubbles can be deleted, copied, or moved to other steps in a wizard or to other wizards in Kryon Studio. You can add multiple bubbles to a step and choose where to position them in the screen.

Bubbles & Mouse/Keyboard Control

As an RPA developer, the main question to ask yourself when setting the properties of a bubble is: What propels the flow forward? Who controls the bubble's timing and decides when to close it – the user, or the robot?

If the bubble timing is user-controlled, this means that while the bubble is displayed, the user is given control over the mouse and keyboard. This allows the user to freely click on, hover over, and type in the application for the duration of the bubble action.

If the bubble is robot-controlled, this means that the robot has control over the mouse and keyboard while the bubble is displayed. The user does not have control to perform any actions on the application but must allow the robot to control it.

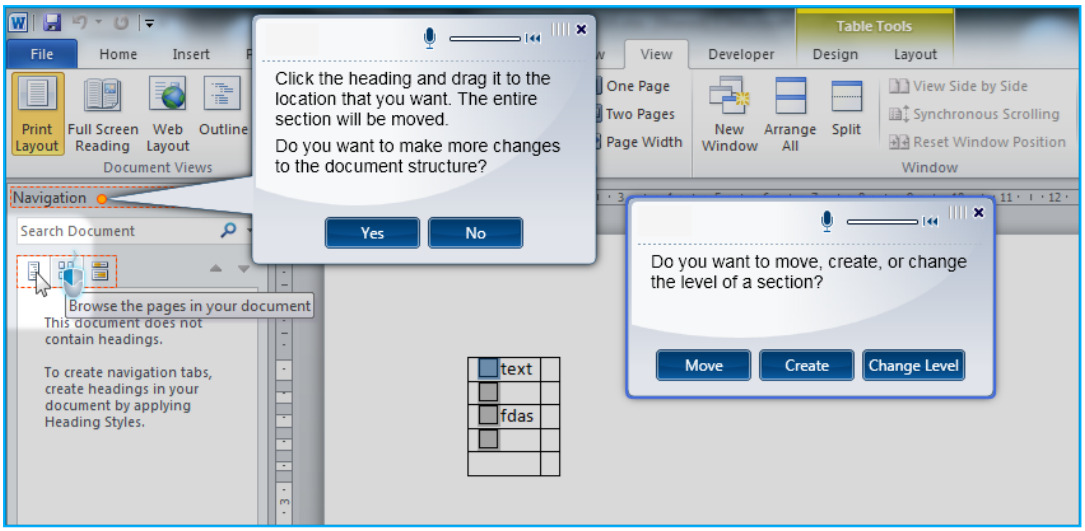
User-controlled timing

A bubble is user-controlled when it is set to hide On button click.

A user-controlled bubble appears before or after the step's core action or another bubble, and disappears when the user clicks one of its buttons. Once the button is clicked and the bubble disappears, the robot resumes control of the mouse and keyboard, and proceeds to the next action in the wizard flow.

User-controlled bubbles:

- Contain interactive buttons
- Can contain input fields
- Appear one at a time in a wizard, in the order defined by the RPA developer
- Cannot appear at the same time as the core action or another user-controlled bubble
- Can appear at the same time as robot-controlled bubbles
- Can be anchored or non-anchored
- Can proceed when the user clicks on/hovers over the anchored object



Robot-controlled timing

A bubble is robot-controlled when it is set to hide by any option other than On button click.

A robot-controlled bubble does not pause the wizard at runtime. The robot continues to work while the bubble is displayed, and the user is not given control of the mouse or keyboard. If the user attempts to use the mouse or keyboard while the bubble is displayed, the wizard is paused.

A robot-controlled bubble can be timed to disappear after a few seconds, or when another bubble is closed.

Robot-controlled bubbles:

- Do not contain buttons or input fields
- Are useful for tips and explanations that do not require user interaction
- Cannot appear at the same time as the core action
- Can be timed to disappear after a few seconds
- Can be set to hide at the end of a subsequent step
- Can appear at the same time as other bubbles or tooltips
- Can be anchored or non-anchored

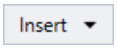
Working with Bubbles

Bubbles can be added to a step either before or after the core action.

- Bubbles can be added, removed, and moved within the step's flow
- Multiple bubbles can be added to a single step

Adding bubbles

To add a bubble to the selected step:

1. Go to the Flow tab of the Flow Pane
2. Click the **Insert** button 

3. Select the desired bubble type from the dropdown list
4. Move your mouse over the Display Plane and click in the location of the block you want to create
    - You can move and resize a bubble after it has been created

The bubble appears in the Display Pane and in the Flow tab.

NOTE

You can add an unlimited number of bubbles to a single step.

Editing bubble text

To edit the text in a bubble:

1. In the Display Pane, double-click the bubble whose text you want to edit

    The bubble is enabled for editing, and the **Text Editing** toolbar appears above the bubble:

2. Type and format the bubble text as required
3. Click anywhere outside the bubble to exit text editing mode

Inserting an image into a bubble

To insert an image into a bubble:

1. In the Display Pane, double-click the bubble into which you want to insert the image
2. In the **Text Editing** toolbar, click the **Insert image** button and select the image to insert

The image is inserted into the bubble.

ink

Edit image size first!

Once inserted into a bubble, the image size cannot be changed. So be sure to resize the image using an image editor before inserting it into the bubble.

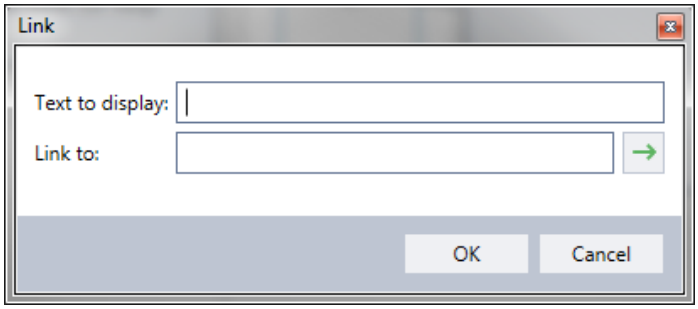
Inserting a hyperlink into a bubble

You can insert links into bubbles to help users access additional information that is relevant to the wizard.

To insert a hyperlink into a bubble:

1. In the Display Pane, double-click the bubble into which you want to insert the link
2. In the **Text Editing** toolbar, click the **Insert link** button and select the image to insert

   The **Link** dialog box appears:



3. In the **Text to display** field, type the text that you want to display for the link
4. In the **Link to** field, type or paste the link URL
   - To test the link (i.e., go to the URL you entered), click the button
5. Click **OK**

The link is inserted into the bubble.

Moving and resizing bubbles

Bubbles can be resized and moved as follows:

- To resize the bubble, click the bubble and then click and drag the bubble borders.
- To move the bubble, click and drag the bubble to a different location.
- To move a bubble anchor, click and drag the anchor object to a different location.

Hiding/showing bubbles in the display pane

In certain steps, you may find that bubbles are clustered in a single location on the screen, making them difficult to see and edit in the Display Pane. To view a bubble that is behind another bubble, you can either move the bubble, bring it to the front (or send other bubbles to back), or hide the bubble in the Display Pane.

NOTE

Hiding a bubble in the Display Pane does not delete it. The bubble remains active and visible to end users in Kryon Robot during wizard runtime.

To hide or show a bubble in the Display Pane:

1. From the Flow tab of the Flow Pane, click the **Show/Hide** button for the bubble that you want to show or hide

NOTE

Bubbles are shown in the Display Pane by default, with the **Show/Hide** button enabled.

Bringing a bubble to the front

To bring a bubble to the front for editing purposes, do one of the following:

- From the Flow tab of the Flow Pane, click on the bubble that you want to bring to the front;
- From the Display Pane, click any area of the bubble that is visible; ***or***
- From either the Display Pane or the Flow Pane, right-click the bubble and select **Bring to Front** from the menu
    - You can similarly send any bubble to the back by right-clicking it and selecting **Send to Back** from the menu

NOTE

Bringing a bubble to the front (or sending it to the back) does not change the order in which it appears when the wizard is run. Bubbles are displayed during runtime in the order which they appear in the Flow Pane. To learn about changing this order, see Reordering bubbles within a step.

Reordering bubbles within a step

Bubbles in a step appear to the user in the order in which they are listed in the step flow, either before or after the core action. You can reorder bubbles/core action within in the step.

To reorder bubbles/core action within a step:

1. From the Flow tab of the Flow Pane, click and drag the bubble/core action to the desired location within the flow.

Moving bubbles among steps/wizards

You can move bubbles within a wizard and/or from one wizard to another by cutting and pasting them.

To move bubbles among steps/wizards:

1. In either the Display Pane or the Flow Pane, click the bubble to select it
2. Cut the bubble by doing one of the following:
    - On the Wizard Editor menu bar, select **Edit > Cut**
    - On the toolbar, click the **Cut** icon
    - Press `CTRL+X`
3. Select the step to which you want to move the bubble (either in the current wizard or another wizard)

4. Paste the bubble in its new location by doing one of the following:
   - On the menu bar, select **Edit > Paste**
   - On the toolbar, click the **Paste** icon
   - Press `CTRL+V`

Copying bubbles

You can copy bubbles within a wizard and/or from one wizard to another. Copying bubbles is useful when you want to create new bubbles based on existing bubbles.

To copy a bubble:

1. In either the Display Pane or the Flow Pane, click the bubble to select it
2. Copy the bubble by doing one of the following:
   - On the menu bar, select **Edit > Copy**
   - On the toolbar, click the **Copy** icon
   - Press `CTRL+C`
3. Select the step to which you want to copy the bubble (either in the current wizard or another wizard)
4. Paste the bubble in its new location by doing one of the following:
   - On the menu bar, select **Edit > Paste**
   - On the toolbar, click the **Paste** icon
   - Press `CTRL+V`

Deleting bubbles

Studio allows you to delete any bubble from a wizard.

To delete a bubble:

1. In either the Display Pane or the Flow Pane, click the bubble you want to delete
2. Do one of the following:
   - On the menu bar, select **Edit > Delete**
   - In the Flow Pane, click the

button

NOTE

You can undo/redo bubble deletion as required until you save or close the wizard (at which point deletion becomes permanent).

Bubble Types

There are four types of bubbles available in Kryon Studio:

- Bubble
- Anchored bubble
- Tooltip
- Help button

Bubble

The standard **bubble** is a robot- or user-controlled bubble that floats on the window, not pointing to any specific area.

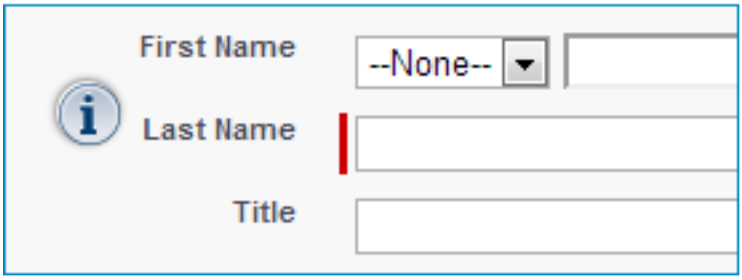
Anchored bubble

An **anchored bubble** is a robot- or user-controlled bubble with an anchor that points to a specific area of the window (often referred to as a "callout").

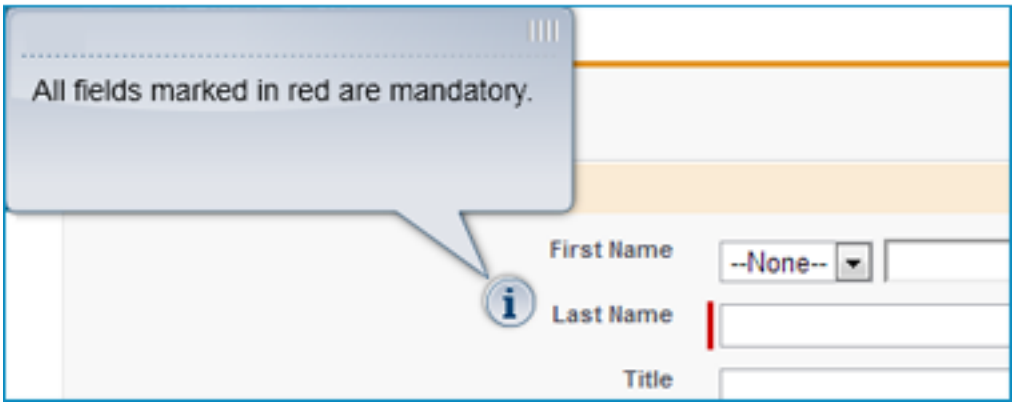
Tooltip

A **tooltip** is robot-controlled bubble (icon) that informs the end-user of additional information or validation errors. The tooltip appears as an icon on the screen, attached to an anchored bubble that appears when the user hovers over the icon with the mouse cursor.

Default:



On hover:



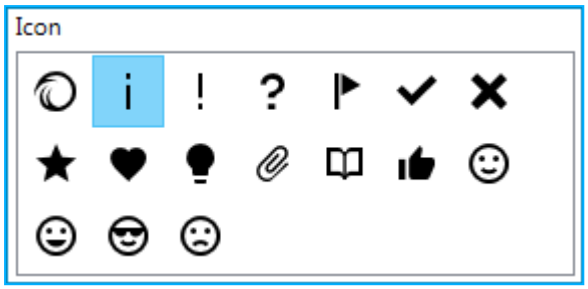
Though a tooltip's timing is robot-controlled, the user must be allowed control of the mouse while the tooltip is displayed so that he can hover over the tooltip icon and view the bubble itself. Therefore, tooltips must be timed to appear in one of the following ways:

- With a previous user-controlled bubble; ***or***
- In a step whose core action is Wait

Tooltips:

- Do not contain buttons or input fields
- Are useful for tips and explanations that users can expand if needed
- Cannot appear at the same time as the core action
- Can appear at the same time as other bubbles or tooltips
- Cannot be timed to disappear after a few seconds
- Can be set to hide at the end of a subsequent step
- Must be anchored (and cannot have a transparent anchor)

- Have an icon that can be selected by the RPA developer



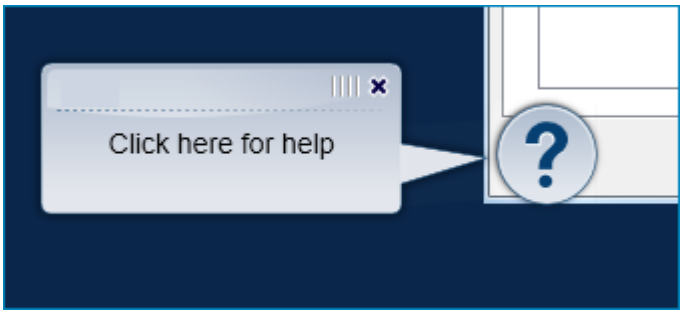
CAUTION

Tooltips can only be displayed with other bubbles, but never on their own

- **Exception:** A tooltip can be used on its own in a step whose core action is **Wait**

Help button

A **help button** is a robot-controlled bubble (icon) that allow the user to simultaneously display or hide all other robot-controlled bubbles and tooltips on the screen. By default, a help button hides all robot-controlled bubbles that precede it until it is clicked. It appears as an icon on the screen, anchored to a hint bubble that appears when the user hovers over the icon with the mouse cursor:

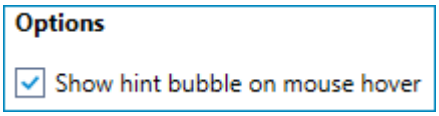


Though a help button's timing is robot-controlled, the user must be allowed control of the mouse while the button is displayed so that he can hover over the button and view its hint bubble. Therefore, help buttons must be timed to appear in one of the following ways:

- With a previous user-controlled bubble; ***or***
- In a step whose core action is Wait

Help buttons:

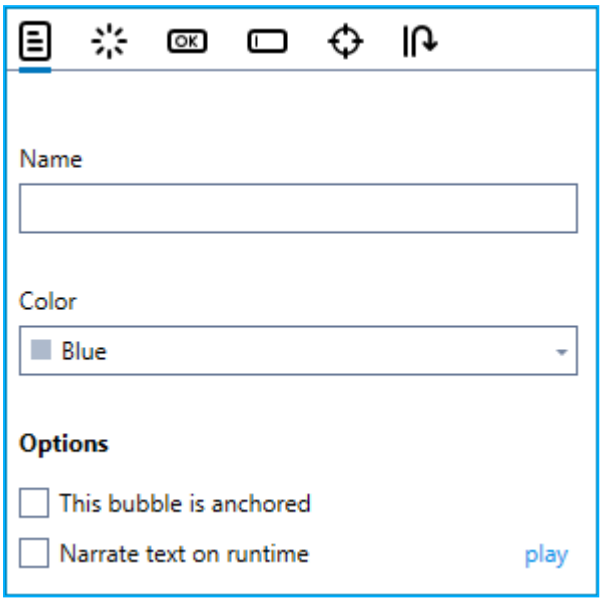
- Have anchored hint bubbles with no buttons or input fields
  - A help button's anchored hint bubble can be disabled



- Are useful for tips and explanations that do not require user interaction
- Cannot appear at the same time as the core action
- Can only appear with other robot-controlled bubbles and/or tooltips
- Can only be timed to disappear at the end of a step or with a preceding bubble
- By default, cause all preceding robot-controlled bubbles and tooltips to be hidden until clicked
- Can be moved around the screen by the end-user during wizard/sensor runtime
- Have an icon that can be selected by the RPA developer

Bubbles: General Properties

The following general bubble properties can be customized from the Properties tab of the Properties Pane:



Name

Enter an optional **Name** to serve as a reference for the RPA developer in Kryon Studio. The **Name** is not visible to end users in Kryon Robot.

Color

Select the background color of the bubble from the dropdown list.

- Determines the color of the bubble as it appears in Studio and in Kryon Robot at wizard runtime.

Anchored bubble options

This bubble is anchored

Tick/clear the **This bubble is anchored** checkbox to determine whether the bubble is an anchored bubble.

- For anchored bubbles, continue by setting the anchor object position.

NOTE

The initial state of this checkbox is determined by the bubble type selected when the bubble was added.

Use transparent anchor

Tick/clear the **Use transparent anchor** checkbox to determine whether an anchored bubble's callout anchor is invisible to the end user at wizard runtime.

- This checkbox is visible only for anchored bubbles.
- When this checkbox is selected, the bubble's anchor remains visible in the Wizard Editor's Display Pane but is invisible when the wizard is run.

TIP

Why use a transparent anchor?

If you want a bubble to look like a non-anchored bubble but behave like an anchored bubble (e.g., be hidden when an object on the screen changes or is covered by a window), you can use an anchored bubble with a transparent anchor. The bubble will have the same Show/Hide properties as any anchored bubble, but will appear as non-anchored to the user.

Narrate text on runtime

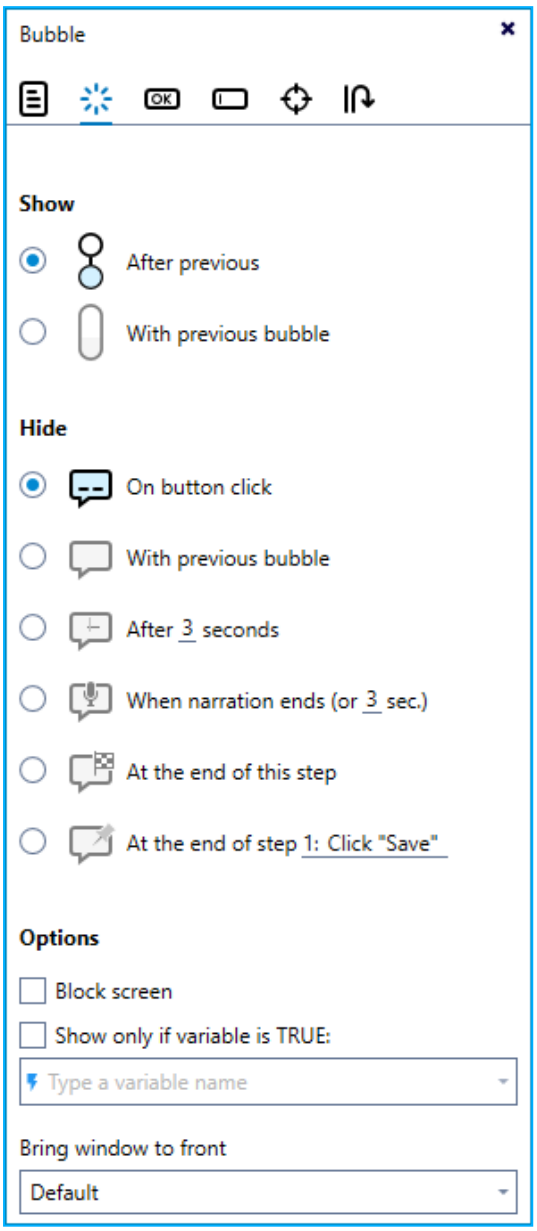
Tick/clear the **Narrate text on runtime** checkbox to determine whether bubble text will be automatically read aloud when the bubble appears at wizard runtime.

- To exclude a portion of bubble text from being narrated, edit the bubble text by adding angle brackets (<>) around the text to exclude

To test the bubble's narration, click the **play** link to the right of the checkbox.

Bubbles: Show/Hide Properties

Bubble **Show/Hide** properties determine the timing and conditions for showing/hiding bubbles and bubble anchors. They are managed from the **Show/Hide** tab of the Properties Pane:



Show

Choose from these options that determine when to show the bubble:

- **After previous:** The bubble appears after the previous bubble is closed
- **With previous bubble:** The bubble appears along with the previous bubble

Hide

Choose from these options that determine when to hide the bubble:

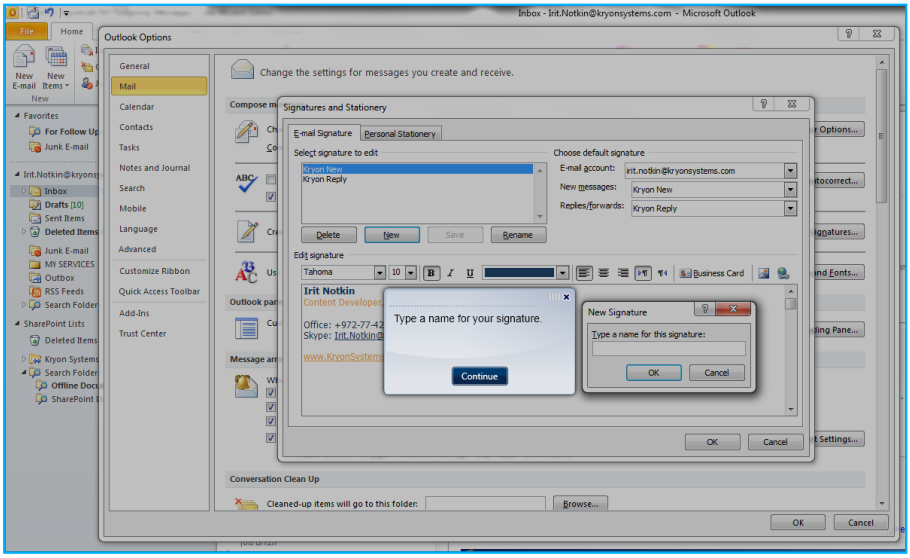
- **On button click:** Hide the bubble when the user clicks one of the bubble buttons (applies to bubbles and anchored bubbles only)
- **With previous bubble:** If the bubble is shown with the previous bubble, hide both bubbles at the same time
- **After [X] seconds:** Hide the bubble after the number of seconds specified (applies to bubbles and anchored bubbles only)
- **When narration ends:** Hide the bubble when the audio narration stops; or, if narration is disabled for the bubble, hide the bubble after the number of seconds specified (applies to bubbles and anchored bubbles only)
- **At the end of this step:** Hide the bubble at the end of the current step
- **At the end of step [X]:** Hide the bubble at the end of the specified subsequent step (applies to bubbles, anchored bubbles, and tooltips only)

NOTE

If the bubble is set to hide **On button click**, this means it has user-controlled timing. All other **Hide** options apply only to bubbles with robot-controlled timing.

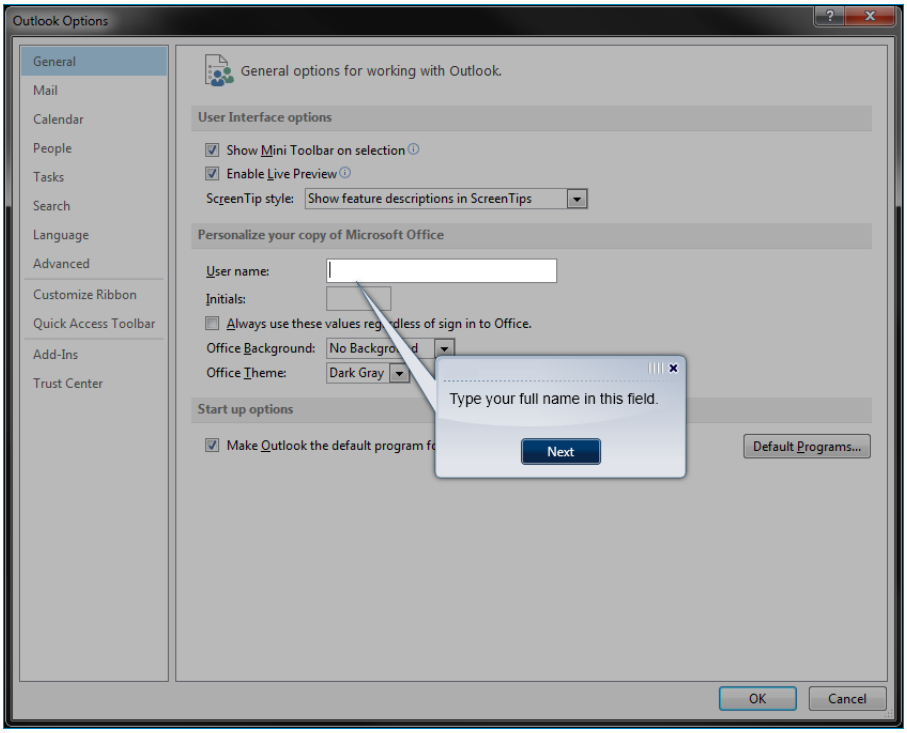
Block Screen

Tick this checkbox to block the user's entire screen for as long as the bubble is displayed (i.e., the user cannot click anything on the screen around the bubble). During runtime, the user will see a grayed-out screen around the bubble:



This option is available only for bubbles set to hide on button click.

If the bubble is anchored and its highlight box is enabled, the entire screen around the highlighted box is blocked:



Show only if variable is TRUE

Tick this checkbox to display the bubble only if a specific variable's value is `True`. If the variable's value is `False`, the bubble is not displayed to the user.

NOTE

The specified variable would have acquired its value through the application of a logical sequence earlier in the wizard (generally, using advanced commands or Read from screen).

EXAMPLE

You want a bubble to appear to the end user only if a particular field on the user's screen is empty. (For purposes of this example, let's say that we're checking a form's date field.)

To make this happen, you could:

1. Determine if the date field is empty by setting the step's core action to read from screen using the Is Empty value type
2. If the date field is empty, the value `True` will be returned in the Read from screen variable (let's say that we've assigned this variable the name Date Field Empty)
3. Add a bubble after the core action and set the bubble's **Show/Hide** properties to show the bubble only if the Date Field Empty variable has a value of `True`

Bring window to front

Select an option from the dropdown list to determine whether the robot should automatically activate the window to which the bubble is related:

- **Default:** The robot automatically determines whether it should bring the window on which the bubble appears to the front of the screen. If the window has already been brought to the front, the wizard does not attempt to bring it to the front.
- **Always:** The robot always attempts to bring the window to the front of the screen. This is useful for preventing cases in which the window might be unexpectedly deactivated by another window that is brought to the front.
- **Never**: The robot never attempts to bring the window to the front of the screen. The robot shows the bubble if it detects the window, regardless of whether or not the window is brought to the front.

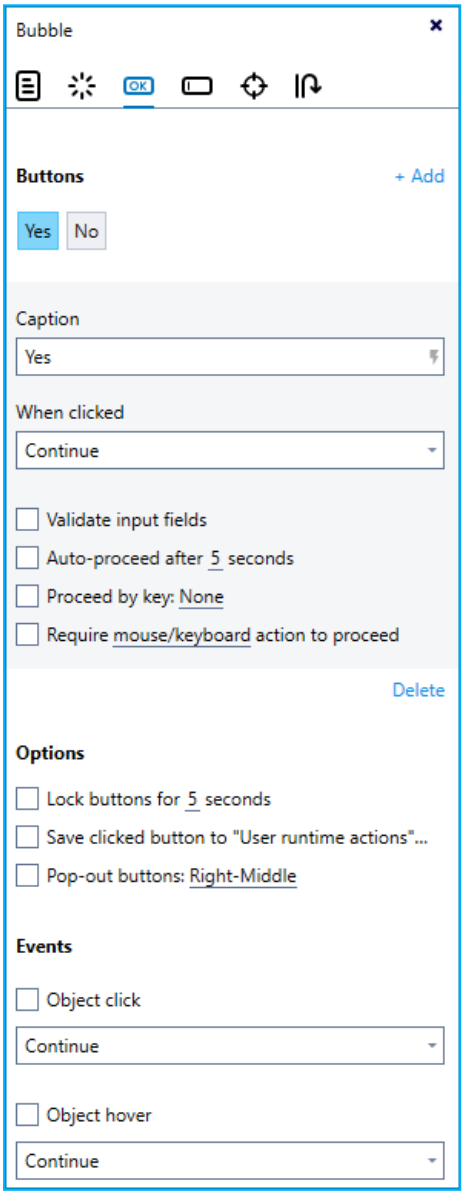
Bubble Buttons

User-controlled bubbles contain interactive buttons that determine how to continue the wizard flow (i.e., each button is assigned a command that tells the wizard what action to perform when the user clicks it).

A bubble can contain an unlimited number of buttons, but none are required.

- If a bubble does not contain at least one button, it is considered robot-controlled, and you can configure its behavior by showing/hiding the bubble under specific conditions in the step
- By default, a bubble contains one **Next** button, set to continue to the next step action when clicked

Bubble button properties are managed from the **Buttons** tab OK of the Properties Pane:



Adding buttons to a bubble

To add a button:

1. In the **Buttons** tab of the Properties Pane, click the **+ Add** link

A button is added to the bubble, with the caption **New button**.

Deleting buttons

To delete a button:

1. In the **Buttons** tab of the Properties Pane, select the button you want to delete
2. Click the **Delete** link

The button is deleted from the bubble.

Caption

The caption is the text that appears on the button. Be sure to give each button within a bubble a ***unique*** caption.

When clicked

From the **When clicked** dropdown box, select a command to be assigned to the button:

- **Continue:** When the button is clicked, the wizard continues to the next step action. This is the default setting.
- **Go to step:** When the button is clicked, the wizard goes to the step in the wizard flow that you specify. This command is useful when there is a decision point in the wizard flow and the user must choose between several paths.
- **End wizard successfully:** When the button is clicked, the wizard ends with a scenario reported as successful. This command is useful when you want to end the wizard early in the flow, or to create a decision point that splits the wizard flow into multiple paths. If the wizard is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's success scenario as defined by the RPA developer.
- **End wizard unsuccessfully:** When the button is clicked, the wizard ends with a scenario reported as unsuccessful. Unsuccessful is not considered a failure, but rather an ending that the RPA developer chose to design as an alternative ending to the wizard. This command is useful when you want to end the wizard before the end of the flow, or to create a decision point that splits the wizard flow into multiple paths. If the wizard is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's alternative scenario as defined by the RPA developer.

- **Advanced commands:** When the button is clicked, the wizard triggers a predefined set of advanced commands. When the advanced commands are completed, the wizard continues by executing the next step action.

Single button options

The following options can be configured for each individual button in a bubble:

Validate input fields

Tick/clear the **Validate input fields** checkbox to determine whether to enforce field validation.

Auto proceed after [X] seconds

Tick/clear the **Auto proceed** checkbox to determine whether the robot should automatically click the bubble button if the user doesn't click a button within a few seconds after the bubble appears on his screen.

- When using this option, enter the number of seconds the robot should wait before proceeding
- During runtime, this option is indicated by a countdown indicator on the button:

Proceed by key

Tick the **Proceed by key** checkbox to activate a keyboard shortcut for the button.

- When this option is used, the user can "press" the bubble button by pressing the keyboard key assigned to the button (or by clicking it with the mouse, as any regular button)
- The keyboard shortcut keys that can be assigned are **TAB** or **ENTER**
- During runtime, this option is indicated by the keyboard shortcut name under the button:

Require mouse/keyboard action to proceed

Tick the **Require mouse/keyboard action to proceed** checkbox to require user interaction with the application before allowing him to click the button to close the bubble (i.e., the button is disabled until the user performs a mouse click or keystroke in the application).

- When using this option, select whether to require a mouse click, a keyboard stroke, or either
- During runtime, this option results in the following if the user attempts to click the button before performing a click or keystroke in the application:
  - The bubble text jiggles and the bubble stays in place
  - The button command does not occur
  - A yellow bar appears under the bubble, providing users with the option to continue despite not having interacted with the application:

Group Button Options

The following options can be applied collectively to all buttons in a bubble:

Lock buttons for [X] seconds

Tick the **Lock buttons for [X] seconds** to disable the bubble buttons for a configurable number of seconds after the bubble appears on the user's screen. (i.e., the user cannot click the button during that time).

- When using this option, enter the number of seconds the buttons should remain locked
- During runtime, this feature is indicated by grayed-out buttons and a countdown indicator at the top of the bubble:

Save clicked button to user runtime actions

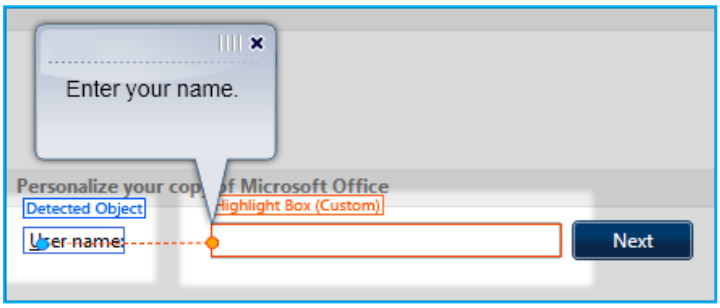
Tick this checkbox is selected to report usage statistics for any button clicked in that bubble by the user. This report is available in the Kryon Report Generator under **User Runtime Actions**.

Pop-out buttons

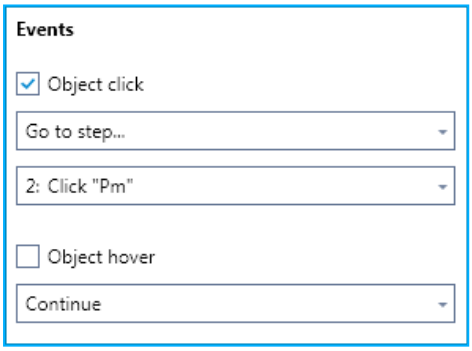
Applicable to anchored bubbles only.

Tick the **Pop-out buttons** checkbox to move the bubble's buttons from the bubble onto the screen, providing a more intuitive experience when user input is required.

- When using this option, select the desired button location from the dropdown list (relative to the anchored object's highlight box)



Bubble events



For each anchored bubble that is set to hide On button click, you can optionally add commands for when certain events occur:

- **Object click:** The user clicks the bubble's detected object
- **Object hover:** The user hovers above the bubble's detected object

To add commands for bubble events:

1. Tick the checkbox(es) for the event(s) to which you want to assign a command
2. From the dropdown box, select the command you want to assign to the event
    - The available commands are same as the When clicked button commands

CHAPTER 13: Fallbacks

In this chapter:

What is a Fallback? 192
Fallback Events 193
Fallback Commands 194
Applying Fallbacks 195
Fallback Usage & Examples 196

What is a Fallback?

Each action in a step depends on window detection and/or object detection. If the expected window or object is not found, the robot's default behavior is to raise an error.

A **fallback** is a backup action you specify so that, in case this occurs, the wizard will continue uninterrupted instead of raising an error.

TIP

Best practice

For each action, you can ***and should*** provide a **fallback**.

A **fallback** consists of 2 parts –

- Fallback event: the event that triggers the backup action
- Fallback command: the backup action that is carried out when the fallback event occurs

NOTE

The following actions do not have fallbacks:

- Step end
- Screen block
- Window block
- Key block

Fallback Events

Depending on the step action for which you are defining the fallback, you can specify a fallback command for the following events:

- **Window not found:**
  - The robot attempts to detect the window in which the action is performed. If the robot fails to detect the window, it triggers this fallback. The default fallback is an error message.
  - For **Step start** actions, the robot attempts to detect the window in which the step occurs before the step starts. If the robot fails to detect the window, it performs a fallback. The default fallback is an error message.
  - For **Bubble** actions, the robot attempts to detect the window on which to run the step, after the step starts and before the core action is performed. This might occur because the bubble precedes the core action and blocks the window, or because the user closes the window while the bubble is shown. If the robot fails to detect the window, it performs a fallback. The default fallback is an error message.
- **Window blocked:** This fallback event handles dialog boxes opened by the application that block access to the recorded window
- **Window closed:** This fallback event handles scenarios in which the recorded window was initially detected successfully, but was closed in the middle of the step
- **Object not found:** The robot attempts to detect the click position before it performs the action. If the robot fails to detect the click position, it performs a fallback. The default fallback is an error message.
- **Object changed:** This fallback event is available for anchored bubbles only
- **Object hidden:** This fallback event is available for anchored bubbles only
- **Wizard not found:** This fallback event is available for embedded or launched wizards only
- **Wizard ended unsuccessfully:** This fallback event is available for embedded or launched wizards only

Fallback Commands

There are a number of commands available to handle each fallback event:

- **Raise error:** An error message appears on the wizard bar. Not applicable to sensors or embedded wizard steps
- **End step:** The step ends and the wizard continues with its When step ends event
- **Go to step:** The wizard goes to a specific step in the flow. Not applicable to embedded wizard steps
- **Remove all blocks:** Dismisses all blocks and cancels their functionality for that run of the wizard or sensor
- **Keyboard shortcut:** The wizard types a keyboard combination entered by the RPA developer (e.g., `ALT+P`)
- **Advanced commands:** The wizard triggers a predefined set of advanced commands. When the advanced commands are completed, the wizard flow continues in the order determined by the RPA developer
- **End wizard successfully:** The robot exits the wizard. If the wizard is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's success scenario as defined by the RPA developer.
- **End wizard unsuccessfully:** The robot exits the wizard. If the wizard is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's failure scenario as defined by the RPA developer
- **Fail sensor:** The sensor ends and is and activated again based on recurrence settings, date settings, and whether a predefined visual object is detected. If a visual object is detected, the sensor is re-launched when the object is detected. **Not applicable to wizards.**

In addition to backing up failed scenarios, fallbacks can be used to create more complex wizards, which are capable of performing more than just linear procedures. You can use fallbacks to create alternative paths for the wizard flow, which are activated according to what the user chooses to do.

Applying Fallbacks

To apply a fallback to a step action:

1. Select the step to which you want to add the fallback
2. From the Flow Pane, do one of the following:
   - Access the **Fallback display**
   - Access the relevant action's **Properties Pane**; then locate the relevant fallback event in the Fallbacks tab
3. From the fallback event's dropdown list, select a fallback command

The fallback is applied to the step action.

Fallback Usage & Examples

The following sections describe the available fallbacks and usage examples:

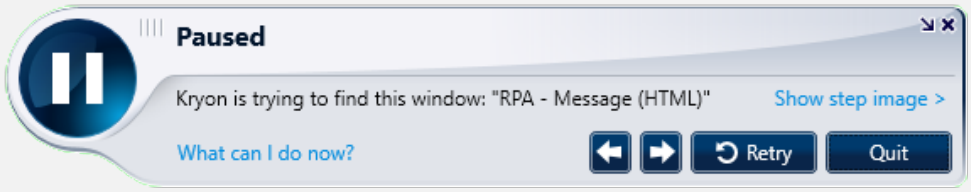
- Raise error
- End step
- Go to step
- Keyboard shortcut
- End wizard successfully
- End wizard unsuccessfully

Raise error

When the **Raise error** fallback command is triggered for a core action, the **Paused** message is raised in the wizard bar, describing why the wizard was paused.

EXAMPLE

You recorded a wizard for creating a new mail message and typing its content. If, while playing the wizard, the user changes the subject of the message before the wizard reaches this step in the flow, the message's window caption is changed. In this scenario, the robot looks for a window called **RPA - Message (HTML)** but cannot find it because the caption was changed by the user. If the **Raise error message** fallback was applied to this step, it is activated and a **Paused** message appears on the user's screen:



This error can be avoided by setting the window caption to **Contains** instead of **Equals**. Read more about editing window properties.

End step

When the **End step** fallback is applied to a step, the step is ended and then continues with the command specified in its **When step ends** event. It is the default action when the wizard successfully performs a step.

End step is useful as a fallback when, after a bubble, the user closes the window that the wizard is about to close. In such cases, this fallback enables the wizard to end the step instead of trying to detect the closed window.

EXAMPLE

You recorded a wizard for specifying a file name and location. In this scenario, users might click **OK** to confirm the action instead of waiting for the wizard to do so. This closes the window in which the wizard is supposed to click **OK** for the user. If the **End step** fallback was applied to this step, it is activated and the wizard ends the step instead of trying to detect the **OK** button.

Go to step

When the **Go to step** fallback is applied to a step, the wizard goes to the specified step in the flow. This is useful for creating wizards that offer two or more alternative paths.

EXAMPLE

You recorded a wizard that moves an email message to another folder. If there is an open message on the screen, the wizard continues by moving this message. If there is no open message and the **Go to step** fallback was applied, it is activated and the wizard goes to the designated step (in this case, maybe a step in which the user is prompted to select a message to move).

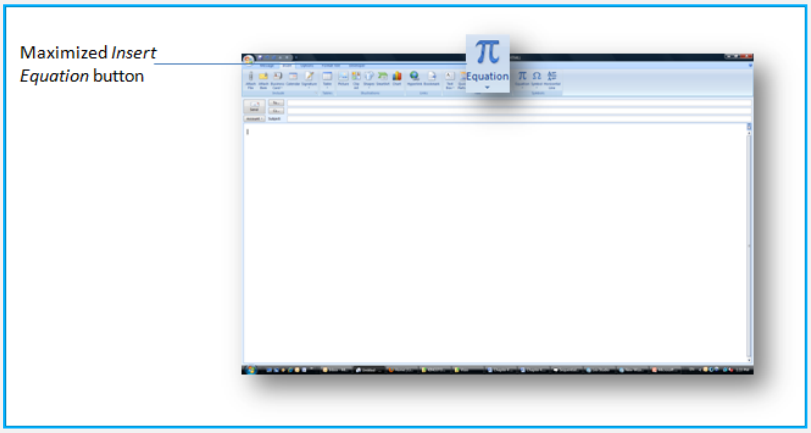
Keyboard shortcut

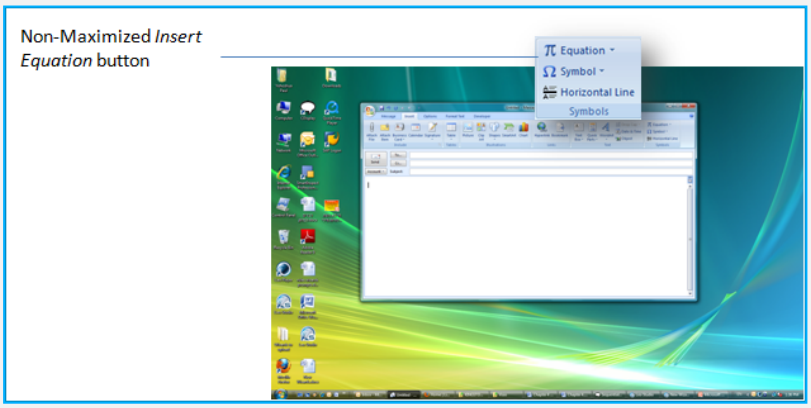
When the **Keyboard shortcut** fallback is applied to a step, the wizard types a keyboard shortcut instead of clicking to perform the core action. This is useful when the action could be performed with either a shortcut key or a mouse click. In this case, if the wizard cannot visually detect the object on the user's screen, it uses the specified keyboard shortcut as a fallback.

- This option is disabled in embedded wizard steps.

EXAMPLE

You recorded a wizard for inserting an equation into an email message in a maximized screen. If the screen is not maximized, the **Equation** button's appearance changes and the wizard cannot detect it.





If the **Keyboard shortcut** fallback was applied to this step, it is activated and the wizard types the specified shortcut keys (e.g., `ALT+N+E`) to click the **Equation** button.

End wizard successfully

When the **End wizard successfully** fallback is applied to a step, the robot exits the wizard. This is useful when you want to end the wizard early in the flow.

Unattended/Hybrid Only In an unattended automation context, the wizard will be reported as ended successfully in Kryon Console dashboard.

NOTE

If the wizard containing the step is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's ***success*** scenario as defined by the RPA developer.

EXAMPLE

You recorded a wizard for saving messages, in which the final step is the Microsoft Outlook message that appears if the user tries to save over an existing message. If the user chooses not to replace the existing message and the **End wizard successfully** fallback was applied to this step, it is activated and the wizard is ended at this point in the flow because there are no more actions to take.

End wizard unsuccessfully

When the **End wizard unsuccessfully** fallback is applied to a step, the robot exits the wizard. This is useful when you want to end the wizard early in the flow.

Unattended/Hybrid Only In an unattended automation context, the wizard will be reported as ended unsuccessfully in the Kryon Console dashboard.

NOTE

If the wizard is an embedded wizard (that is, a wizard inserted as a building block into the flow of the current wizard), the flow continues with the containing wizard's ***failure*** scenario as defined by the RPA developer.

CHAPTER 14: Embedded Wizards

You can think of an embedded wizard as a building block of one or more larger wizards.

- An embedded wizard is recorded and edited once and then reused in multiple locations, without the need to edit it separately in each location.
- Any wizard from any library can be used as a building block within another wizard, regardless of the application on which it was recorded.

NOTES

- **Attended/Hybrid Only** An embedded wizard can be hidden from end users in Kryon Robot (i.e., excluded from search results) by setting its run mode to `None`
- Embedded wizards are not available for sensors

In this chapter:

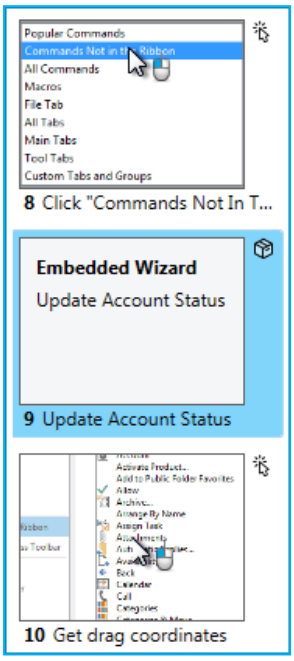
Embedded Wizard Features 201
Embedded Wizard Fallbacks 203

Embedded Wizard Features

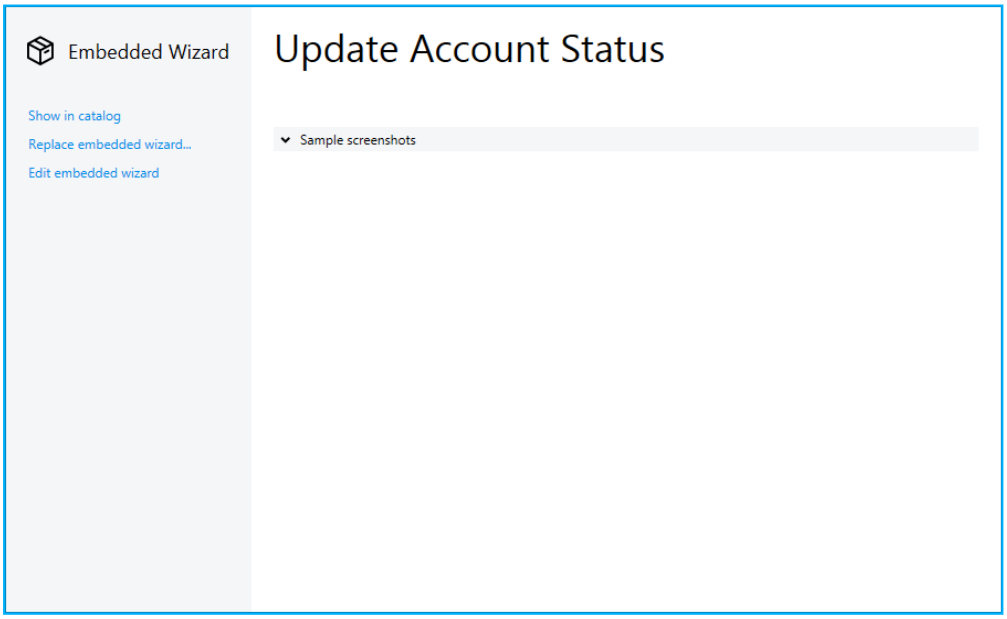
An embedded wizard's features can be managed from its Control pane or the Properties tab.

Control pane

To view an embedded wizard's **control pane**, select its thumbnail from the Navigation Pane:



The **control pane** contains the following features:



- **Show in catalog:** Click this link to show the embedded wizard in its location in the Catalog
- **Replace embedded wizard**: Click this link to open a dialog box that allows you to replace this embedded wizard with a different embedded wizard
- **Edit embedded wizard:** Click this link to open the embedded wizard for editing in a new Wizard Editor window
- **Sample screenshots:** Open this dropdown list to displays preview screenshots of the embedded wizard's steps and step names/bubble text

Properties tab

The features listed above can also be managed from the embedded wizard's Properties tab (with the exception of Sample screenshots).

In addition, the Properties tab displays the embedded wizard's name (as defined in its General Properties).

Embedded Wizard Fallbacks

The fallback events for an embedded wizard are:

- **Wizard not found:** The robot is unable to locate/access the embedded wizard
  - There are a number of reasons this event might occur: the user running the containing wizard does not have permissions to access the embedded wizard, the embedded wizard was not imported, etc.
- **Wizard ended unsuccessfully:** The embedded wizard ended unsuccessfully before it completed its run. The default fallback command for this event is to continue the flow of the containing wizard.

The default fallback command for both of these events is to `Go to next step` of the containing wizard.

To change the default fallback command for one or both of these commands, follow the instructions in Applying Fallbacks.

CHAPTER 15: Advanced Commands

In this chapter:

Advanced Commands: Introduction 205
Advanced Commands Editor 206
Invoking Advanced Commands 207
Variables 208

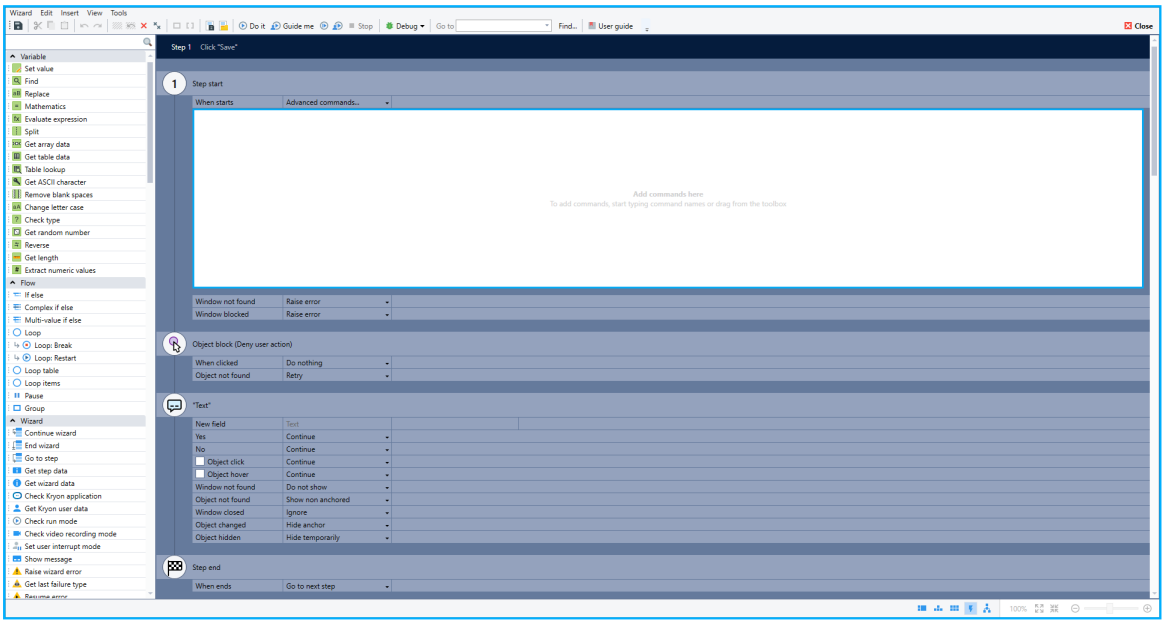
Advanced Commands: Introduction

Advanced commands are a feature of the Kryon RPA Platform that allow business users with little or no programming experience, as well as expert RPA developers, to add sophisticated logic to wizards. Each command provides an instruction to the wizard: to retrieve and utilize data from outside sources, read/write to external applications, execute scripts, direct wizard flow using loops and conditions, and more.

- Advanced commands can be defined directly in the Advanced Commands Editor or invoked from specific locations in a wizard.
- Many advanced commands are comprised of a variable (or multiple variables) and a logical action performed on that variable.
- For detailed information about the wide variety of advanced commands and how to use each of them see the ***Advanced Commands Reference Guide*** available from the Advanced Commands Editor toolbar.

Advanced Commands Editor

Access the **Advanced Commands Editor** by selecting the Advanced command view from the **View Selector** .



Invoking Advanced Commands

Advanced commands can be invoked from the following places in a wizard:

- Step start: Triggers a predefined set of advanced commands when the step begins (even before the robot brings the window to front and activates it)
- Step end: Triggers a predefined set of advanced commands when the step is completed
- Bubbles:
  - Bubble buttons: Triggers a predefined set of advanced commands when the user clicks a bubble button
  - Bubble events: Triggers a predefined set of advanced commands when the user clicks or hovers an anchored bubble object
- Fallbacks: Determines if the expected action failed and triggers advanced commands as a fallback accordingly

Variables

A variable is a named location that stores a textual or numeric value.

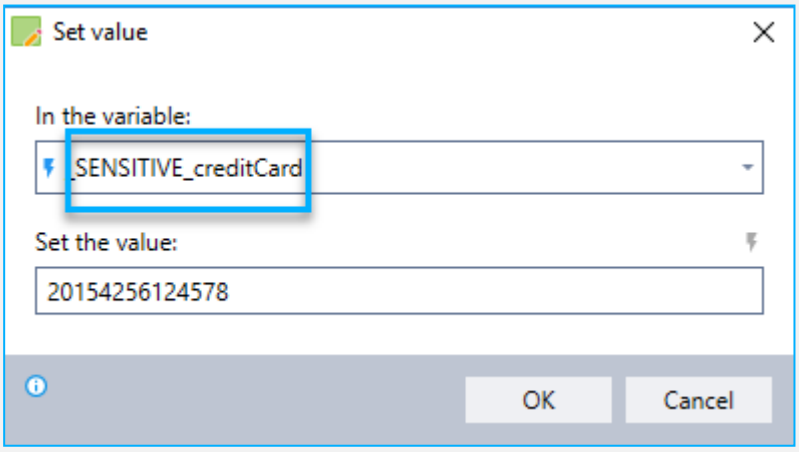
Variable characteristics

Variable characteristics include:

- **Availability:** By default, a variable is valid only during runtime for the wizard for which it was created. The variables of an embedded wizard and its containing wizard become available to each other during runtime.
- **Name:**
  - The variable name is free, alphanumeric text
  - It is not case sensitive, and can include spaces
  - Variable names cannot include any special characters, such as `! @ # $ % ^ & * ( ) / < > - = | \`
  - **Sensitive data:** You can prevent sensitive data from being written to the logs by defining a variable as sensitive. Do this by adding the prefix '`_SENSITIVE_`' to its name. Values of Sensitive variables are not written to logs

FOR EXAMPLE

Suppose you have a variable `creditCard` that represents a credit card number and you want to hide this number from the logs. Simply rename the variable to `_SENSITIVE_creditCard`



- **Value:**
  - Variables do not have to contain a value in order to be used in commands. An undefined variable will simply contain an empty string.

- Variable values can be changed at any time
- The variable value is always treated by the wizard as a string
- Values of *sensitive* variables (identified by the prefix `_SENSITIVE_` added to the variable name) are not copied to the logs

Referencing variables

- In wizard bubbles and commands, variables can be referred to by wrapping them with dollar signs ($), e.g., `$MyVar$`.
- To use a variable as a numeric value, you must wrap the variable name with pound (hash) signs, e.g., `#MyVar#`.
  - A variable used in comparison operators (such as `< > =`) that contain numeric values is automatically defined as numeric. In such a case, you do not need to indicate the variable type.
- A **credential variable** is associated with credential display name(s). Since the credential variables are dynamic, if you export a wizard to another environment, make sure that the credentials associated with the variable are defined in the new environment's credential vault.

Using variables

Variables can be used in advanced commands in order to execute a sequence of actions in a wizard, and also in the following places:

- Bubble text: Insert a variable into the text, so that the text is customized based on the variable's current value
- Bubble button caption: Insert a variable into the text, so that the text is customized based on the variable's current value.
- Bubble Show/Hide properties: Display the bubble only if a specific variable's value is `True`. If the variable's value is `False`, the bubble is not displayed to the user.

CHAPTER 16: Credentials Vault

The Credentials Vault provides secure storage for the usernames and passwords used by robots. The credentials are stored using a 2-phase encryption mechanism, which allows only Kryon clients to retrieve and decrypt the data and enter the credentials when required.

In this chapter:

Accessing the Credentials Vault 211

Types of Credentials 212

Credential Permissions 214

Creating Credentials 215

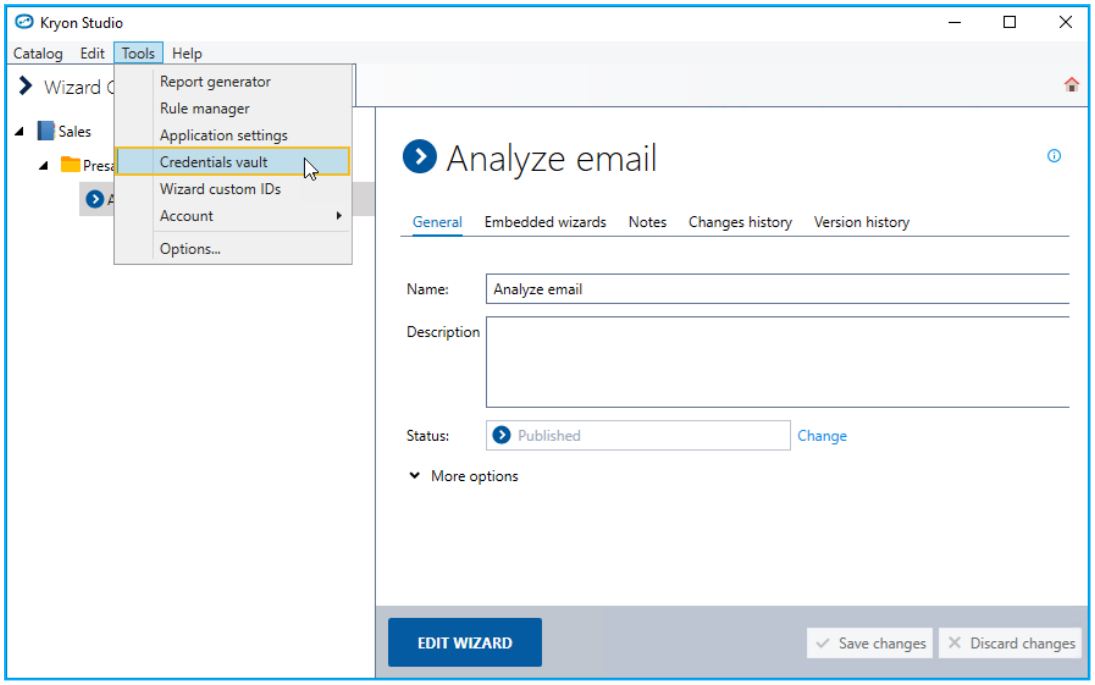
Using Credentials 218

Attended User Access to Credentials 220

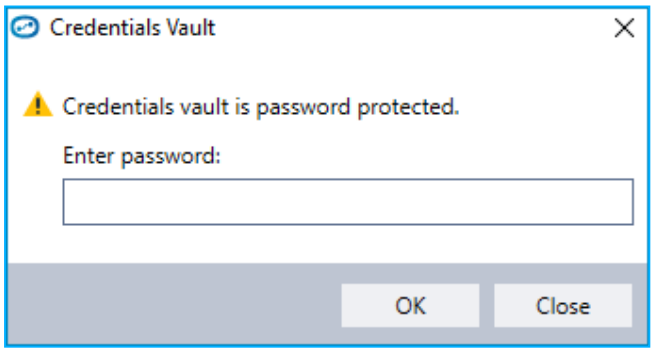
Accessing the Credentials Vault

To access the Credentials Vault:

1. From the menu in the main Kryon Studio window, click **Tools > Credentials Vault**



2. If the vault is password-protected by your system administrator, the following dialog will appear. Enter the password given to you by the system administrator.

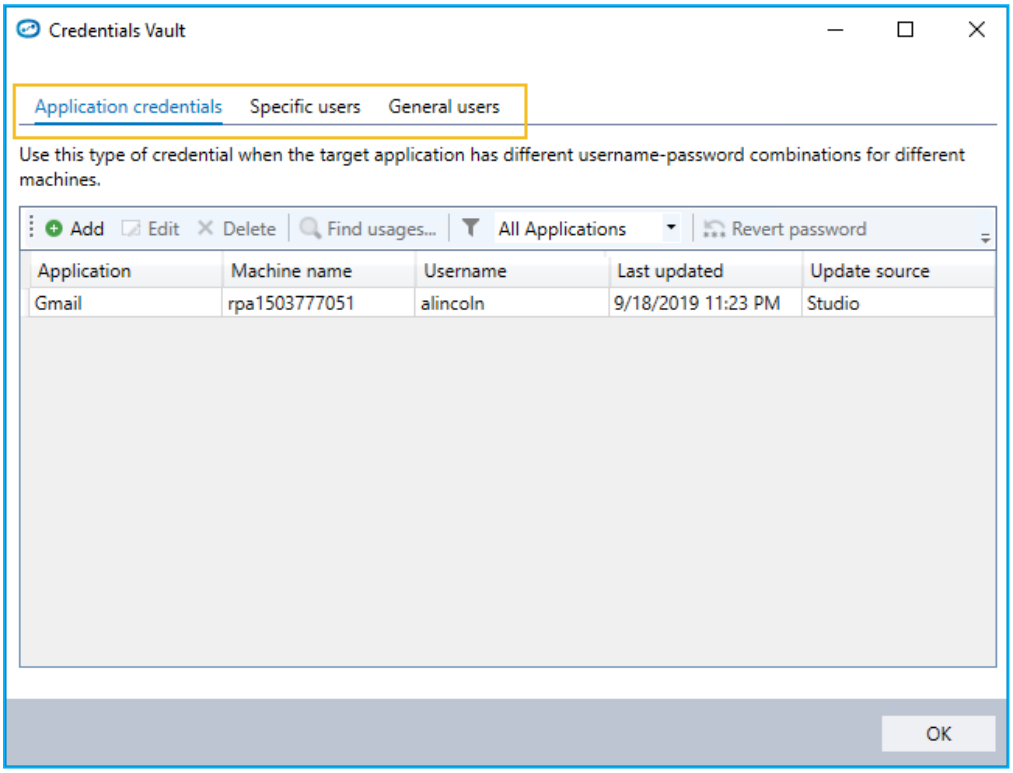


Types of Credentials

Three types of credentials can be created in the **Credentials Vault** from within Kryon Studio:

- Application credential
- Specific users
- General users

Each type is represented by a tab at the top of the **Credentials Vault** window:



Application credential

An **Application credential** is a username-password combination for a specific application ***and for the specific robot using it***. Use this type of credential when the relevant application:

- associates a user with a specific machine; ***or***
- does not permit multiple concurrent uses with the same credentials

When a wizard accesses an **Application credential**, Kryon will fetch the correct username-password combination according to the robot on which the wizard is currently running.

Learn to create an Application credential.

Specific users

A **Specific user** credential is a username-password combination ***that can be used by any robot*** for a specific application. Use this type of credential when the relevant application:

- allows a user to log in from any machine; ***and***
- permits multiple concurrent uses of the same credentials

Learn to create a Specific user credential.

General users

A **General user** credential is a username-password combination that is used to access a general computing resource (such as a mail server or database), as opposed to a specific application.

Learn to create a General user credential.

NOTE

A fourth credential type

An **OS User** credential is a username-password combination that is used by robot to log in to Windows (for purposes of unlocking a locked machine in order to run a wizard).

Since this type of credential is relevant only in an unattended automation context, you can add/edit it only in Kryon Console. For additional information, see the ***Kryon Console User Guide*** – Types of Credentials.

Credential Permissions

When creating a **Specific user** credential or a **General user** credential, you can set permissions for who is authorized to use it: a single user, a list of specific users, a predefined group, or everyone.

When you are developing a wizard in Studio, you can select to include any credential in the wizard (whether or not you are authorized to use it). However, you can successfully run wizards only with credentials for which you are authorized.

At runtime, if the user running the wizard is not authorized to utilize a required credential, the wizard will fail.

CAUTION

By default, a credential can be used by everyone

If a permissions level has not been set for a credential, that credential can be utilized by all Kryon users in your organization.

Permissions level is fixed

Once a permissions level has been set for a credential, it cannot be modified. In order to change an existing credential's permissions, you must delete it and recreate it with the desired level.

For more information about setting a permissions level, see Creating Credentials:

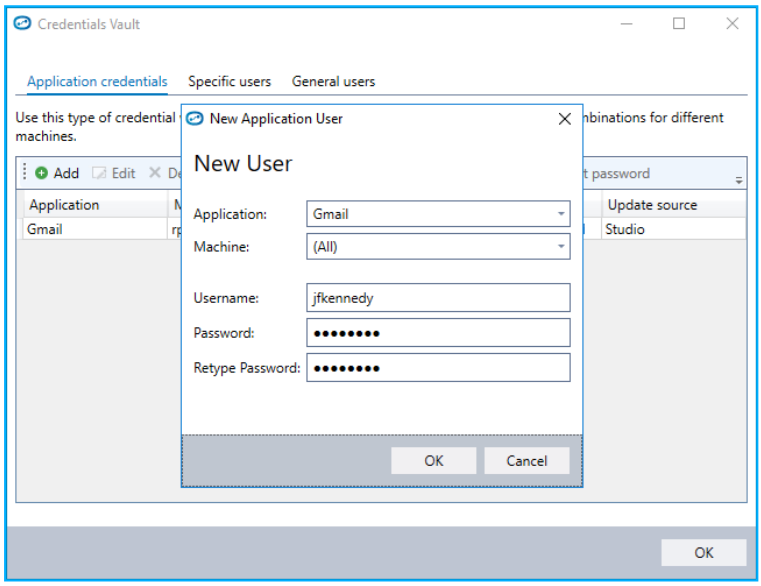
- Specific user credentials
- General user credentials

Creating Credentials

Application credentials

To create an **Application credential**:

1. From **Credentials Vault** main window, select the **Application credentials** tab
2. Click the Add button
3. The following dialog will open:

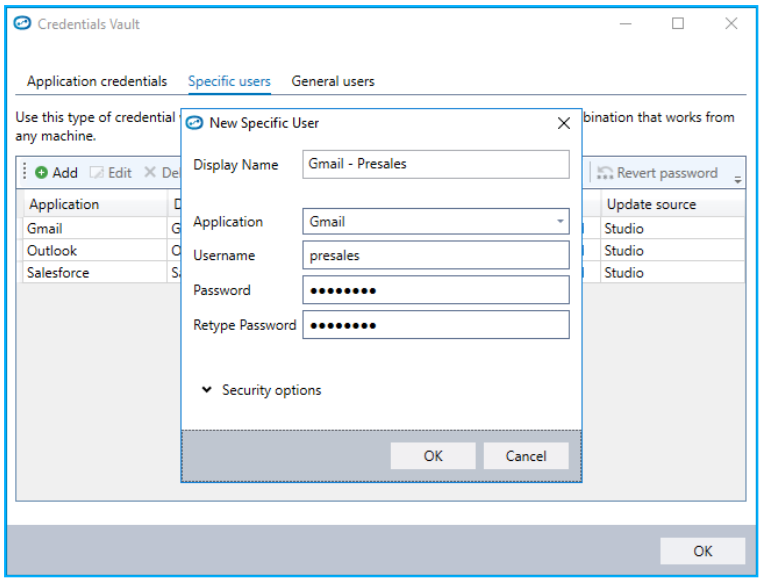


   a. Select the application whose credentials you are entering
   b. Select the machine that will own these credentials, or select **(All)** if these are credentials that every machine can use
   c. Enter the username and password
   d. Click **OK** to save the new credential

Specific user credentials

To create a **Specific user credential**:

1. From **Credentials Vault** main window, select the **Specific users** tab
2. Click the Add button
3. The following dialog will open:

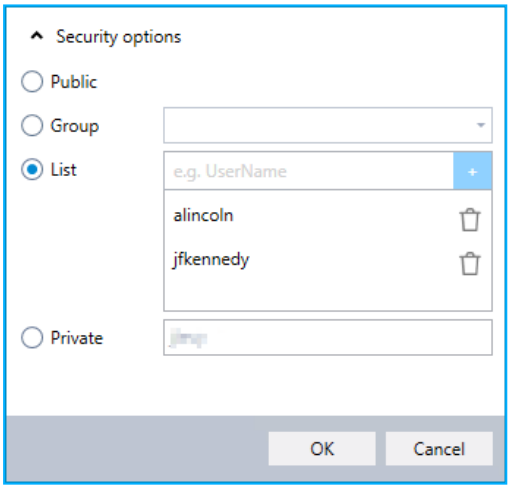


a. Enter a recognizable name that will be used to identify this credential

b. Select the application whose credentials you are entering

c. Enter the username and password

d. Click the arrow next to **Security options** to set the credential permissions:



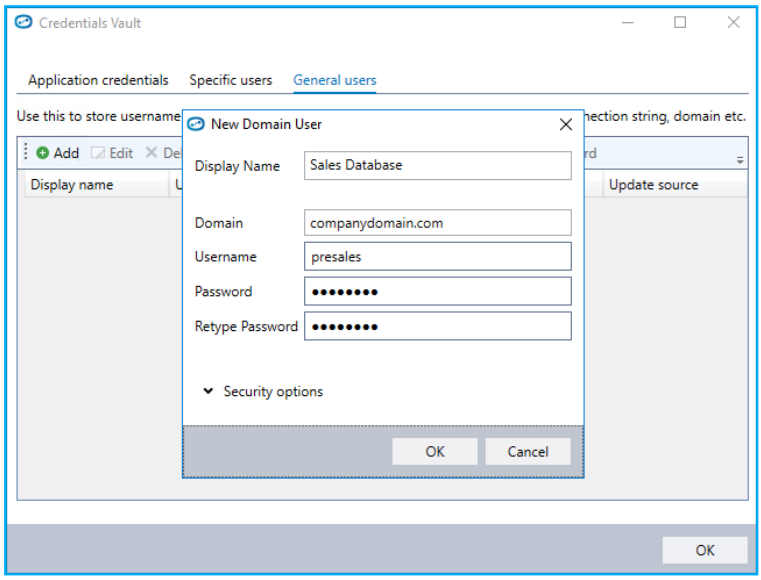
- **Public:** All Kryon users in your organization are authorized to use this credential
- **Group:** Select a group of users who are authorized to use this credential (predefined in Kryon Admin)
- **List:** Enter Kryon usernames of one or more users who are authorized to use this credential
- **Private:** This credential can be used only by you (the user creating it)

e. Click **OK** to save the new credential

General user credentials

To create a **General user credential**:

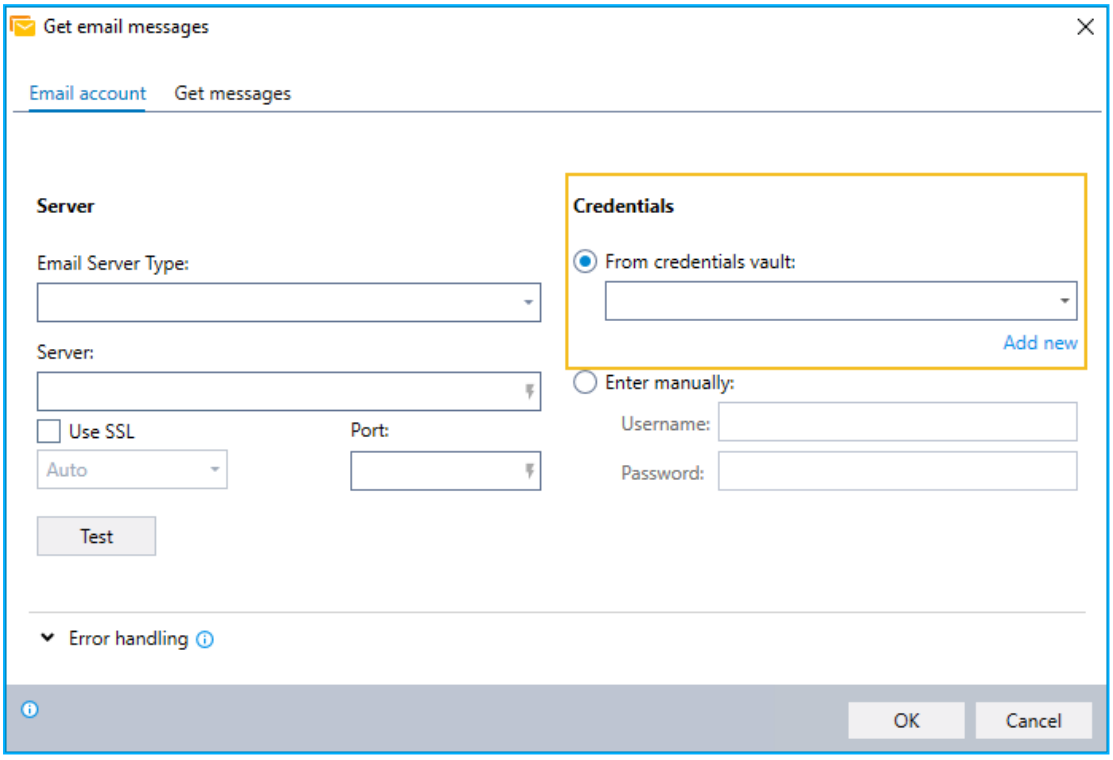
1. From **Credentials Vault** main window, select the **General users** tab
2. Click the Add button
3. The following dialog will open:



   a. Enter a recognizable name that will be used to identify this credential
   b. Select the domain for which the credential will be used
   c. Enter the username and password
   d. Click the arrow next to **Security options** to set the credential permissions, in accordance with the options described in the **Specific user credentials** section, above
   e. Click **OK** to save the new credential

Using Credentials

When using an Advanced Command that requires logging into a resource (application, website, domain, etc.), you have the option to utilize a credential from the **Credentials Vault**. For example, this option appears as follows in the **GET EMAIL MESSAGES** command:

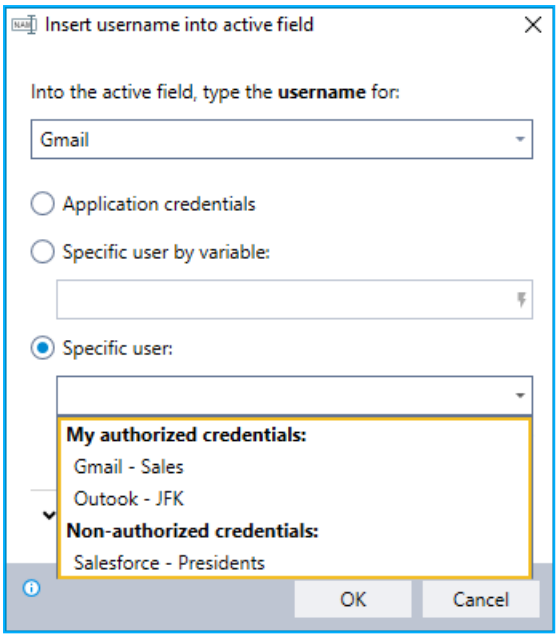


Using non-authorized credentials

As further explained in Credential Permissions, when developing a wizard, you can elect to use a credential for which you do not personally have permissions. What's important is that the user that ***runs the wizard*** is authorized to use it.

When selecting a credential to use, available credentials will be shown in 2 groups:

- those for which you are authorized; and
- those which you are not personally authorized



TIP

When you attempt to run a wizard that utilizes a use a credential for which you are not authorized, the wizard will fail.

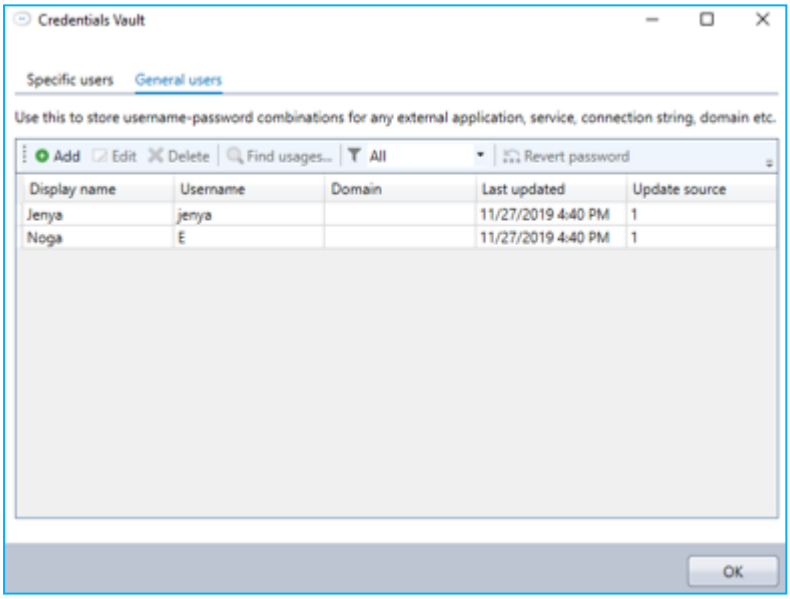
While you are developing/testing a wizard, you may wish to temporarily use a credential for which you are authorized, then change it to the runtime user's credential when the wizard is published.

Attended User Access to Credentials

During an attended robot session, a password that the robot needs may be missing or invalid. You can manage your own passwords straight from the assistant:

TO ADD OR EDIT YOUR CREDENTIALS FROM THE ASSISTANT:

1. In the Assistant, click the **Credential Vault** button . The Credential Vault opens.



2. Select the Credential Type tab

- **Specific users** tab for credentials for login to an application
- **General Users** tab for credentials to access computing resources (like a mail server or database) with no login screen

3. To **add a credential**: Click **Add**. The Add Domain User screen opens.
   a. Enter the **display name**, **user name**, **domain name** and **password**
   b. Click **OK**.
   c. In the Credentials Vault screen, click **OK**.
4. To **edit a credential**: Click **Edit**. The Edit Domain User screen opens.

a. Click **Edit**. Edit credential details as required.

b. Click **OK**.

c. In the Credentials Vault screen, click **OK** .

TIP

When you attempt to run a wizard that utilizes a use a credential for which you are not authorized, the wizard will fail.

While you are developing/testing a wizard, you may wish to temporarily use a credential for which you are authorized, then change it to the runtime user's credential when the wizard is published.

CHAPTER 17: Usage Reports

The **KRYON REPORT GENERATOR** enables you to generate reports according to the type of usage data you want to view. Each report can be filtered by parameters such as dates, output type, and wizard libraries.

Report types

The following are the available reports:

Content

- **Wizard Use:** Displays wizard usage statistics such as the number of answered queries, wizard success rates, failed wizards, and library usage over a specific time range
- **Unanswered Queries:** Lists queries for which the user did not get search results, or got results but did not play any wizard
- **Full Catalog (List):** Displays success and feedback statistics for each wizard, listed by the wizard's path in the catalog
- **User Runtime Actions:** Displays clicks on bubble buttons or other actions which you chose to report on when developing the wizard

Sensors

- **Sensor Activity:** Displays a read receipt for each sensor, listed by username
- **User Runtime Actions:** Displays clicks on bubble buttons or other actions which you chose to report on when developing the wizard

Administration

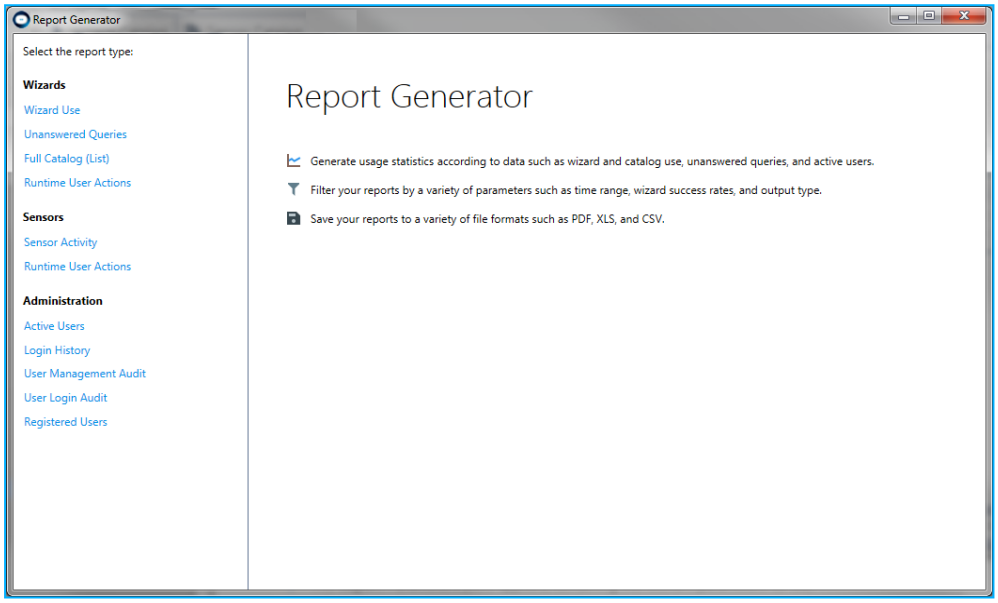
- **Active Users:** Displays the total number of wizards played by each user
- **Login History:** Displays the number of total and first-time logins by Kryon users over a specific time range
- **User Management Audit:** Displays the actions performed on user accounts in Kryon Admin such as user creation, password reset and user deletion
- **User Login Audit:** Displays user actions relating to login such as password changes, failed logins and login history

- **Registered Users:** Displays a brief summary of user registration, login and content development activities

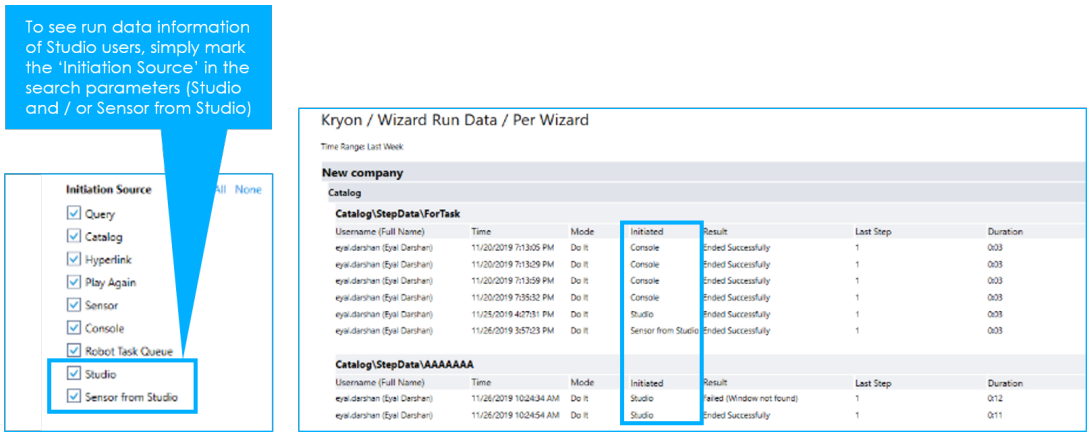
Generating reports

Follow these steps to generate a report:

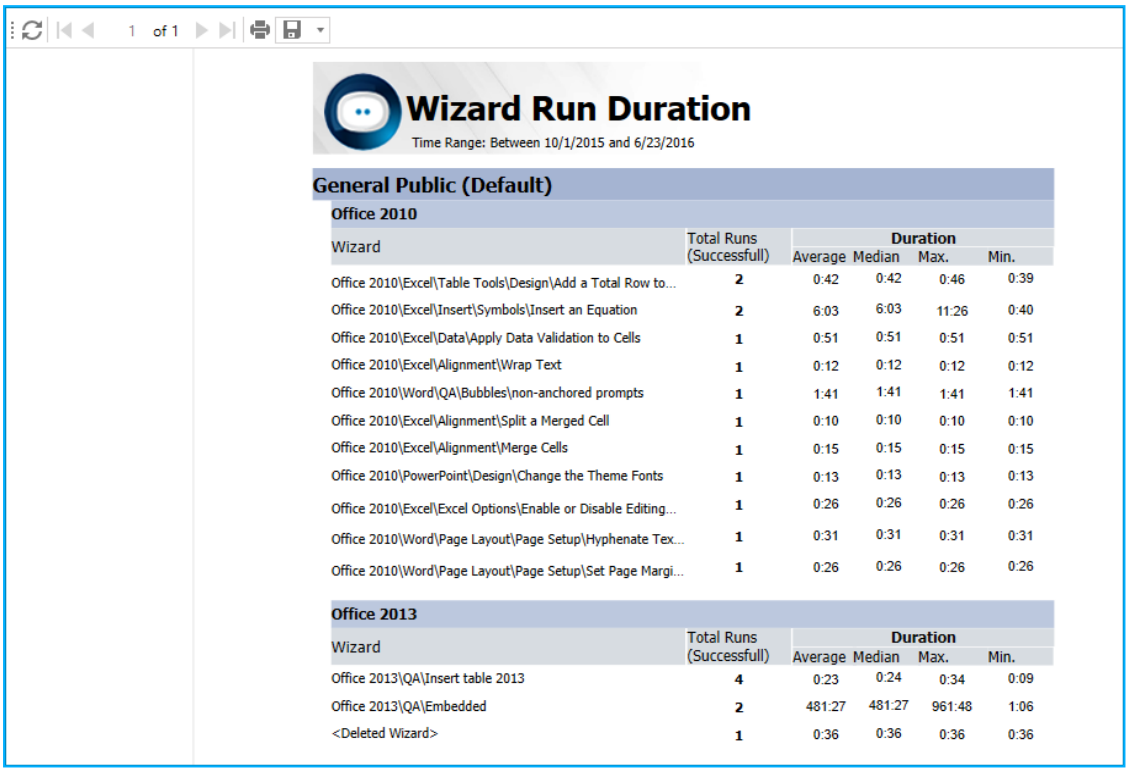
1. From the Kryon Studio menu bar, select **Tools > Report Generator**
2. The **Kryon Report Generator** window appears



3. In the **Report Library** pane, select the type of report to generate
4. The **Report Parameters** pane appears, containing the relevant parameters for the selected report
5. In the **Report Parameters** pane, select the parameters by which to filter your report.
6. To see run data information of Studio users, simply mark the **Initiation Source** in the search parameters (`Studio` and / or `Sensor from Studio`)



7. Click **Get Report**
8. The report you selected appears in the **Report Viewer** pane, filtered by the specified parameters



Wizard Run Duration

Time Range: Between 10/1/2015 and 6/23/2016

General Public (Default)

Office 2010

Wizard	Total Runs (Successfull)	Duration			
		Average	Median	Max.	Min.
Office 2010\Excel\Table Tools\Design\Add a Total Row to...	2	0:42	0:42	0:46	0:39
Office 2010\Excel\Insert\Symbols\Insert an Equation	2	6:03	6:03	11:26	0:40
Office 2010\Excel\Data\Apply Data Validation to Cells	1	0:51	0:51	0:51	0:51
Office 2010\Excel\Alignment\Wrap Text	1	0:12	0:12	0:12	0:12
Office 2010\Word\QA\Bubbles\non-anchored prompts	1	1:41	1:41	1:41	1:41
Office 2010\Excel\Alignment\Split a Merged Cell	1	0:10	0:10	0:10	0:10
Office 2010\Excel\Alignment\Merge Cells	1	0:15	0:15	0:15	0:15
Office 2010\PowerPoint\Design\Change the Theme Fonts	1	0:13	0:13	0:13	0:13
Office 2010\Excel\Excel Options\Enable or Disable Editing...	1	0:26	0:26	0:26	0:26
Office 2010\Word\Page Layout\Page Setup\Hyphenate Tex...	1	0:31	0:31	0:31	0:31
Office 2010\Word\Page Layout\Page Setup\Set Page Margi...	1	0:26	0:26	0:26	0:26

Office 2013

Wizard	Total Runs (Successfull)	Duration			
		Average	Median	Max.	Min.
Office 2013\QA\Insert table 2013	4	0:23	0:24	0:34	0:09
Office 2013\QA\Embedded	2	481:27	481:27	961:48	1:06
<Deleted Wizard>	1	0:36	0:36	0:36	0:36

TIP

Working with reports

To return to the **Report Library** pane and select a different report type, click **Back**.

Use the toolbar at the top of the **Report Viewer** pane to navigate, save, or print your report:

APPENDIX A: Exporting/Importing Catalog Content

In this chapter:

Moving Catalog Items Among Environments 226
Exporting Catalog Items (from DEV) 228
Preparing for Import 231
Importing Catalog Items (to TEST or PROD) 235

Moving Catalog Items Among Environments

The **EXPORT/IMPORT** module is designed to move **Kryon Studio** catalog items (i.e., wizards, sensors, categories) among **Production/Test/Development** environments: **DEV → TEST → PROD**.

As a rule, development is not carried out in the **PROD** environment: creation of wizards/sensors, and changes to existing wizards/sensors should be made only in **DEV** environment and then moved to **TEST** or **PROD** using **Export/Import**.

DEV	TEST (optional)	PROD
• New development • Bug fixes • Updates • Catalog changes (names, location, etc.)	• **No manual changes** • Import from **DEV** only • Any discovered bugs should be fixed in **DEV** and imported into **TEST**	• **No manual changes** • Import from **DEV** only • Any discovered bugs should be fixed in **DEV** and imported into **PROD** once testing is complete

UIDs

For each imported wizard, the **EXPORT/IMPORT** module maintains a reference (known as the **UID**) to the **DEV** environment wizard from which it was originally created. This ensures that future imports will update existing catalog items instead of creating additional items. For example:

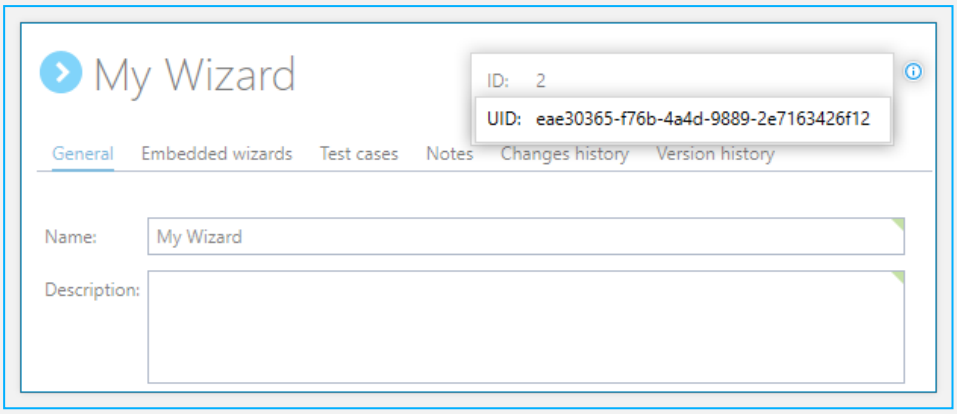
DEV	TEST (optional)	PROD
Category Wizard #1 UID: 1 Wizard #2 UID: 2	Category Wizard #11 UID: 1 Wizard #12 UID: 2	Category Wizard #21 UID: 1 Wizard #22 UID: 2

NOTE

Where can I find the UID?

To locate a wizard's UID:

1. From the Catalog, select the wizard whose **UID** you want to locate
2. Click the info icon in the top-right corner of the screen
3. The **UID** will appear in the popup window (underneath the wizard ID)


My Wizard
ID: 2
UID: eae30365-f76b-4a4d-9889-2e7163426f12
General
Embedded wizards
Test cases
Notes
Changes history
Version history
Name:
My Wizard
Description:

Exporting Catalog Items (from DEV)

Once a wizard/category is selected for export, Kryon Studio prepares a package that includes the following information for each wizard:

- Catalog location and hierarchy
- Metadata (ID, UID, name, description, etc.)
- Wizard steps
  - Recorded data
  - Advanced commands
  - Applications in use
  - Credentials in use
- References to other wizards in the catalog (that are either embedded or launched from the selected wizards)

NOTE

Make sure that all embedded and launched wizards are located in the same library as the one from which you are exporting.

Making changes in the DEV environment

In the **DEV** environment, you can make changes to a wizard that was previously exported to the **PROD** environment (e.g., append new steps, revise logic, add new commands, remove existing commands, use embedded wizards, etc.)

Kryon Studio saves up to 10 history versions of all wizards and sensors. This gives you the ability to roll back to an older version of the wizard (prior to the change that was made).

TIP

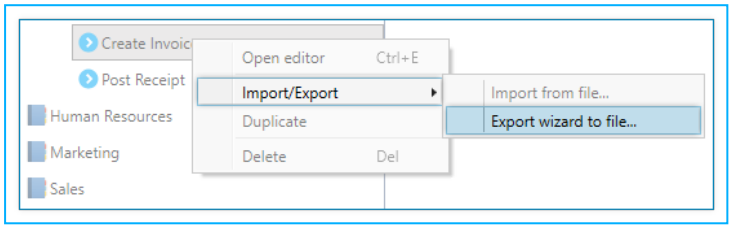
If you want to manually maintain a working version of a wizard in your **DEV** environment, you can use the **Duplicate** option in the catalog: **Select wizard > Right-click > Duplicate**.

If you choose to duplicate a wizard, make sure to make all revisions in the ***original wizard*** and not the duplicate, since the import process utilizes the original wizard's **UID** to update the wizard in **PROD**.

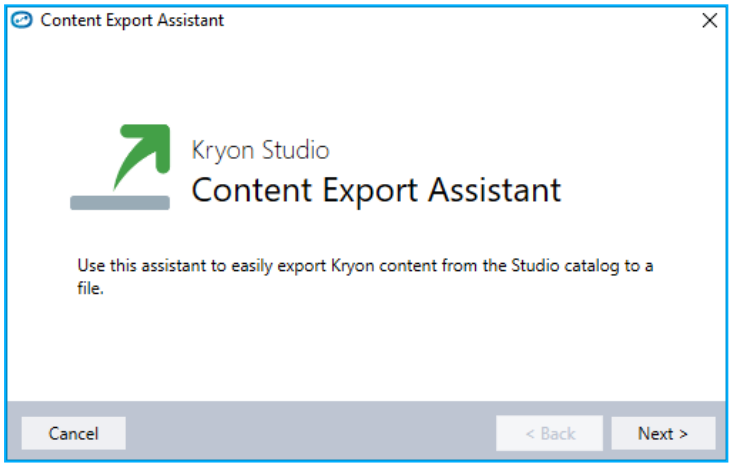
Exporting a wizard/category

To export a wizard or a category from the **DEV** environment:

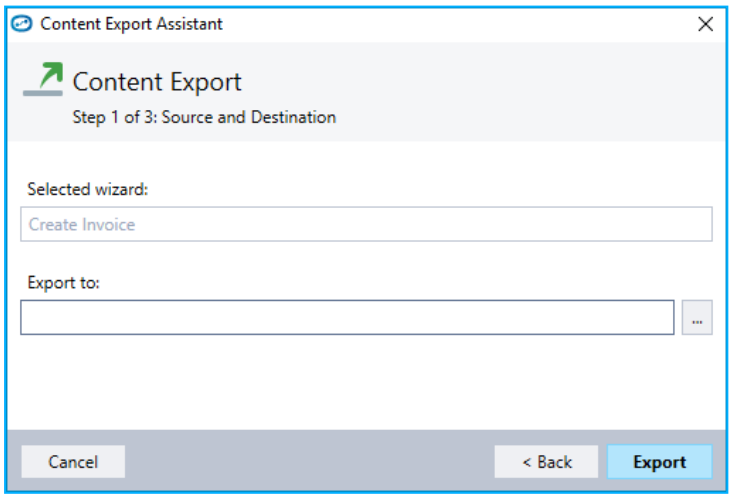
1. From the Catalog, select the wizard or category you want to export
2. Right-click and select **Import/Export**
3. Select **Export wizard to file** or **Export category to file** (as relevant)



4. The **Content Export Assistant** window will open. Click the **Next** button to begin:



5. Select a location in which to save the exported ***.dwiz** file:



6. Click the **Export** button

7. When the export is complete, the file will be saved to the target location you specified
8. Make a note of where the file is located and/or share it with the person that will be importing it into the **PROD** environment

Preparing for Import

Notify stakeholders

If you are updating the **PROD** environment, be sure to notify the relevant stakeholders. Also, ensure all of the planned changes are tested and approved.

Sync application definitions

NOTE

Administrators only!

Application definitions are maintained in **Kryon Admin** on the Kryon RPA Server (not in Studio). Therefore, it is your organization's Kryon RPA Administrator who must sync application definitions.

Each recorded step of a wizard contains a reference to the specific ID of the recorded application as defined in **Kryon Admin**.

Since you will be importing wizards into a different environment (**DEV → TEST/PROD**), it is crucial that the application definitions are the same in both environments. This ensures that the import process will match each step of an imported wizard with the correct application in the target environment.

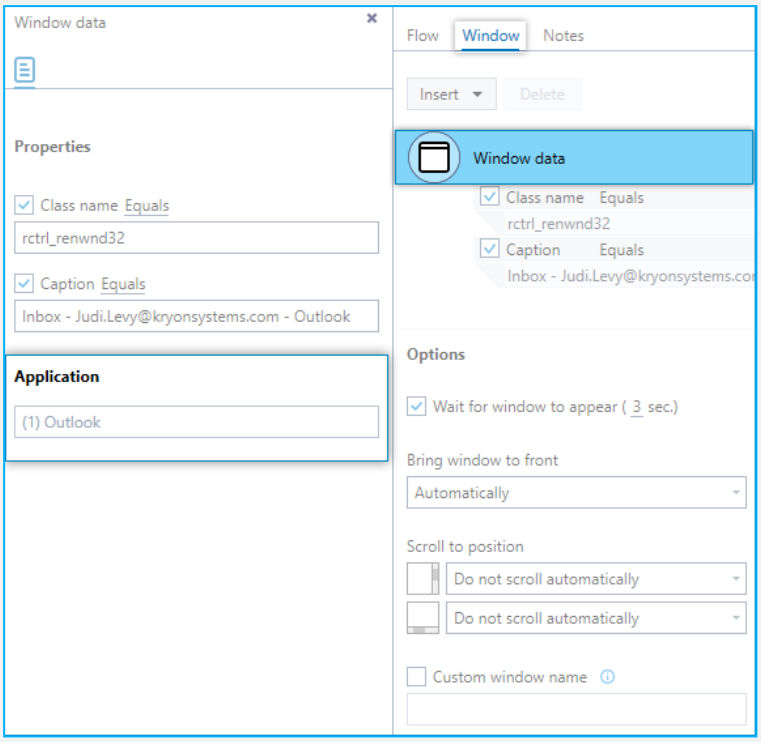
NOTE

How do I identify a step's recorded application?

To identify a step's recorded application:

1. From the Window tab of the Flow Pane, double-click **Window data**

   The Properties Pane will open.
2. The step's recorded application is displayed in the **Application** field

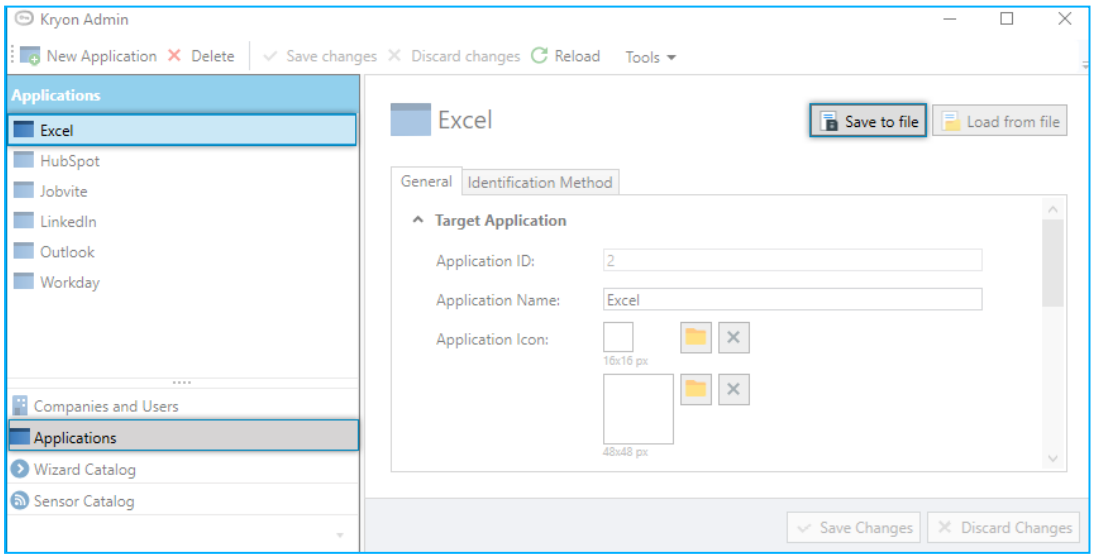


Source environment: save application definitions

Open **Kryon Admin** in your source environment (**DEV**), and click **Applications**.

For each application you wish to save:

1. Select the application from the list
2. Click the Save to file button in the top-right corner of the page



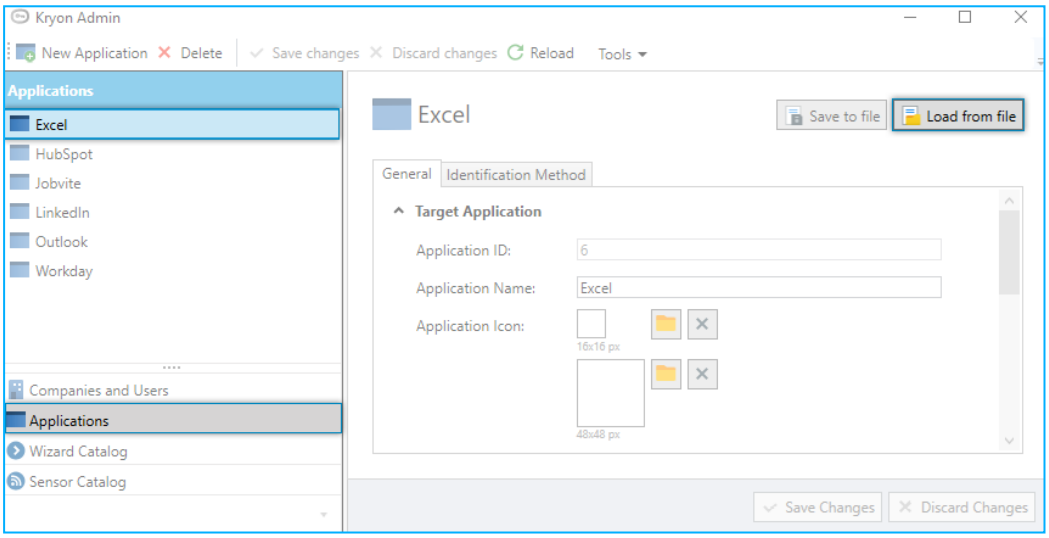
3. Navigate to the location in which to save the **XML** file
4. Click **Save**

Destination environment: load application definitions

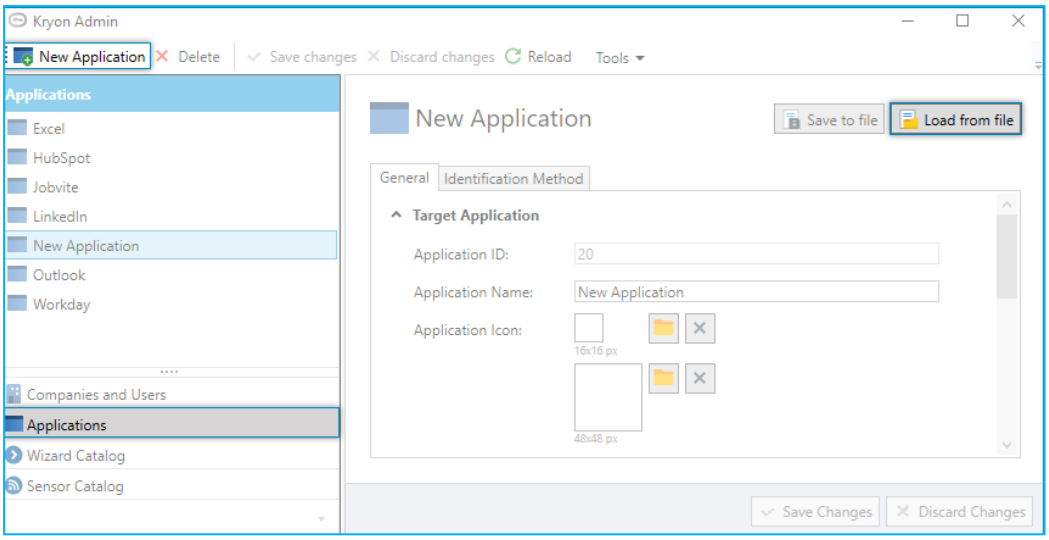
Open **Kryon Admin** in the destination environment (**TEST** or **PROD**), and click **Applications**.

For each application you wish to load:

1. ***If the application already appears in the Applications list***:
   a. Select the application from the list
   b. Click the Load from file button



   c. Navigate to the **XML** file you previously saved for this application and click **Open**
   d. The application definitions will populate the relevant fields
   e. Click the Save Changes button
2. ***If the application does NOT appear in the Applications list***:
   a. From the toolbar, click the New Application button
   b. Click the Load from file button



c. Navigate to the **XML** file you previously saved for this application and click **Open**

d. The application definitions will populate the relevant fields

e. Click the Save Changes button

Importing Catalog Items (to TEST or PROD)

The import process will perform the following actions:

- Validate that all applications in the imported wizards are defined on the destination server

> **NOTE**
>
> Applications are identified by name and detection method. Prior to import, ensure that all application definitions have been synced in Kryon Admin for the **DEV, TEST & PROD** environments. See Sync application definitions.

- Import catalog items
  - Add newly created wizards/update changed wizards
  - Add/update embedded wizards in use
  - Update catalog structure

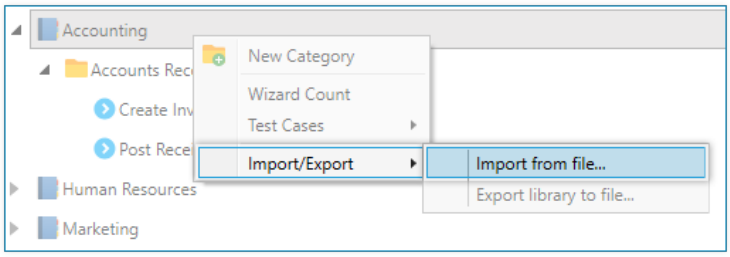
> **NOTE**
>
> If the destination environment already contains older versions of imported wizards, the **Studio** user has the option to update the existing content or create duplicates.

- Add/update credentials in use
- Update references in the imported wizards' steps to:
  - Application IDs in the destination environment
  - Embedded wizard IDs in the destination environment
  - Credentials in the destination environment

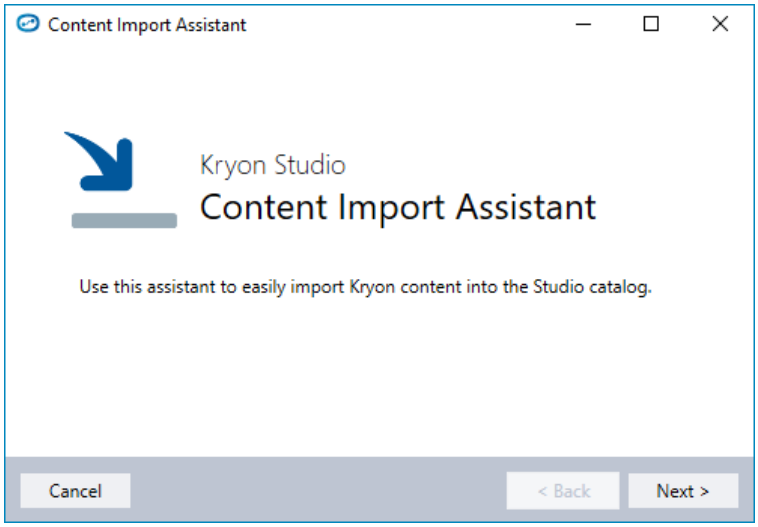
Importing a wizard/category

To import a wizard or a category into the destination environment (**TEST** or **PROD**):

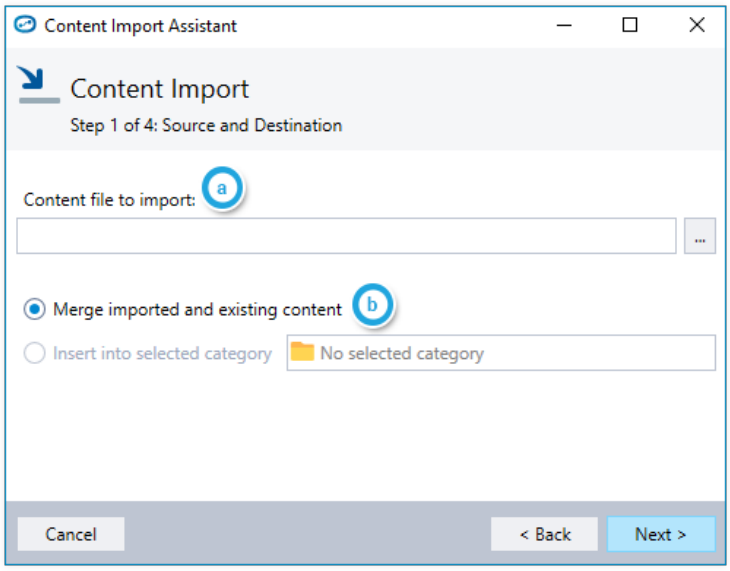
1. From the Catalog, select the parent library or category
2. Right-click and select **Import/Export**
3. Select **Import from file**



4. The **Content Import Assistant** window will open. Click the **Next** button to begin:



5. Specify import source and destination behavior



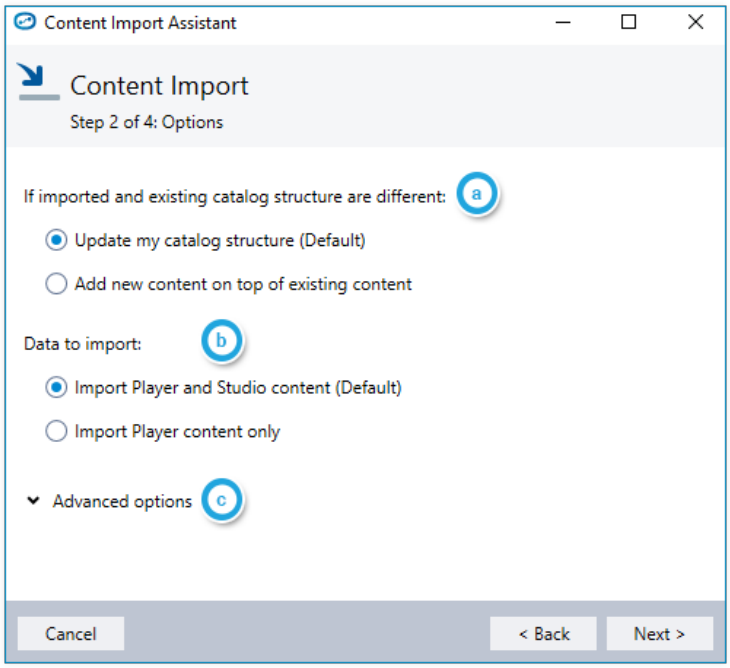
a Import source:

- **Content file to import**
  Navigate to the ***.dwiz** file that you previously exported from **DEV**

b Destination behavior:

- **Merge imported and existing content**
  Recommended. The system will import the content into the same position as it was located in the source environment.
- **Insert into selected category**
  The system will import the wizards into the selected category in the same hierarchy as the source environment. (This option will be available only if you launched the import process from a category in Step 1.)

6. Specify import options



a If imported and existing catalog structure are different:

- **Update my catalog structure**
  Recommended. The system will keep the original UIDs of the wizards so that future imports will update the same content. If those wizards were previously imported into this environment and are located in different categories, the system will update the catalog structure and move them into the new target position.
- **Add new content on top of existing content**
  The system will generate new UIDs for the imported wizards.

> **CAUTION**
> These wizards will not be updated on future imports as they will not have the same UIDs as in the exported package.

b Data to import:

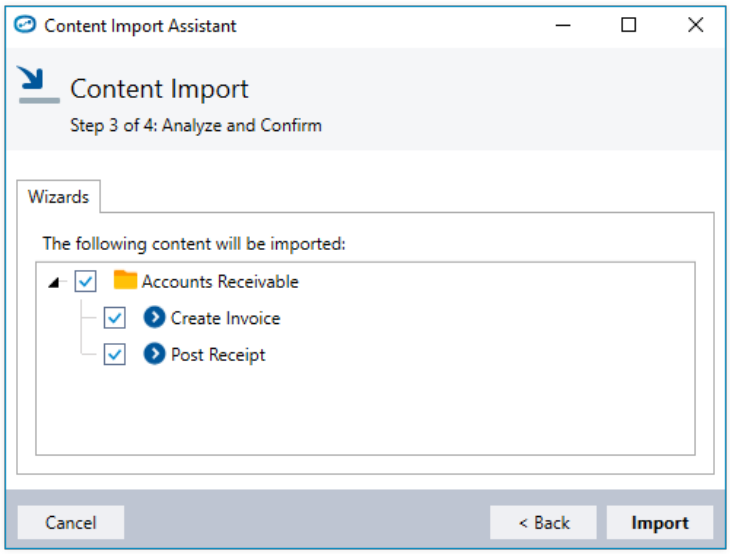
- **Import Player and Studio content**
  Recommended. The system will upload import both Studio wizard information and Player information.
- **Import Player content only**
  Do not use unless explicitly instructed to do so by the Kryon Support team.

Advanced options:

- Do not change these options unless explicitly instructed to do so by the Kryon Support team.

7. Analyze and Confirm

- **Kryon Studio** will analyze the contents of the import package and will display a preview of the imported wizards and sensors
- Click the **Import** button to confirm and start the import



APPENDIX B: Kryon Plugin Development

This topic is intended for computer programmers who need to know how to develop Kryon plugins, for use in the **Run .NET Plugin Method** advanced command in Kryon Studio.

For more information about the **Run .NET Plugin Method** command, see the *Advanced Commands Reference Guide* – **Run .NET Plugin Method**.

Plugin framework

A plugin framework is the most common design pattern used for application extension.

Plugin interfaces enable third-party developers to extend the Kryon RPA Platform and can be used to create an entirely new application.

Most plugin frameworks provide the option of loading plugins at runtime and on demand. However, a plugin must sometimes be unloaded because:

- It consumes excessive resources
- It has not been used for a certain amount of time

Loading an assembly in .NET is generally easy and, for the most part, does not require high plugin design complexity. However, unloading a .NET assembly increases the design complexity: It requires that you implement an interface so that the plugin manager can interact with the loaded assembly.

For the application to be able to unload the plugin, it must first load the plugin assembly in a separate .NET application domain. Loading the assembly in a separate application domain enables the plugin to have its own configuration file and decouples the plugin from the rest of the application. To be able to load and instantiate an object of type **ILeoPlugIn**, the class that implements the **ILeoPlugI**n interface must also inherit from the .NET class **MarshalByRefObject**.

```
public interface ILeoPlugIn
{
string Name { get; }
    void Initialize();
    void Shutdown();
List<LeoPlugInParams> Execute(LeoPlugInMethod pMethod);
}
```

The reason it must inherit from **MarshalByRefObject** is that it is going to be accessed from the main application domain. For the current cross-domain constraint all the same design constraints that exist in an application that uses the Remoting API, also applies in this case. Therefore, to simplify the plugin development, it is suggested to define an abstract class that inherits from **MarshalByRefObject** and implements the **ILeoPlugIn** interface.

```
public abstract class LeoPlugIn : MarshalByRefObject,
ILeoPlugIn
{
    public abstract string Name { get; }
    public abstract void Initialize();
    public abstract void Shutdown();
List<LeoPlugInParams> Execute(LeoPlugInMethod pMethod);
}
```

Once you implement the abstract class **LeoPlugIn**, you must add an assembly attribute in the plugin assembly, the most obvious place being the **AssemblyInfo.cs** file.

```
[Serializable]
[AttributeUsageAttribute(AttributeTargets.Assembly,
Inherited = false, AllowMultiple = false)]
public sealed class LeoPlugInAttribute : Attribute
{
public string PlugInName { get; private set; }
public string EntryType { get; private set; }
public LeoPlugInAttribute (string pluginName, string
entryType)
{
PlugInName = pluginName;
    EntryType = entryType;
}
}
```

Usage:

```
[assembly: LeoPlugInAttribute("PlugInA",
"SamplePlugIns.PlugInA")]
```

This indicates to the **LeoPlugInLoader** that:

- The assembly is a plugin
- The plugin name is **PlugInA**
- The entry class (the class that inherits from the **LeoPlugIn**) is of the type name **SamplePlugIns.PlugInA**

The **LeoPlugInLoader** creates a temporary app domain and load plugin assembly from a provided path. It creates an instance of **LoadAssemblyAttributesProxy** and query for the assembly **LeoPlugInAttribute** attribute. Once the assembly is verified, the **LeoPlugInAttribute** is stored for the instantiation phase. This information is later used by the loader to create an instance of the entry class.

```
public class LoadAssemblyAttributesProxy : MarshalByRefObject
{
public PlugInAttribute[] LoadAssemblyAttributes(string
assFile)
{
Assembly asm = Assembly.LoadFrom(assFile);
var plugInAttributes =
asm.GetCustomAttributes(typeof (PlugInAttribute), false) as
PlugInAttribute[];
return plugInAttributes;
}
}
```

The **LeoPlugInLoader** class creates an **AppDomainSetup** object, sets the name to the plugin name and sets the configuration file to the plugin's ***.config** file (i.e. PlugInName.config.dll), and finally creates the plugin's own application domain (common level of isolation). Once the application domain is created, it remains for future use in expressions. The **LeoPlugInLoader** instantiates the entry class in its corresponding AppDomain by invoking the CreateInstanceAndUnwrap method. This method returns a proxy object to the entry class and is cast to the **ILeoPlugIn**.

Plugin methods

The execution of plugin methods is performed by **PlugInManager** which in turn resides in the external executable **PlugInLauncher**. In order to execute specific methods, you must implement the abstract class **LeoPlugIn** and decorate the particular method with the **LeoPlugInMethodAttribute** attribute. This attribute indicates that the current method should be published and used within the Kryon expression. The attribute also applies the Kryon policy regarding method signature, validates return type and input arguments permitted by Kryon expression .NET types:

- Blittable
- Some non-blittable types: string, object, Boolean

Plugin usage

Kryon plugins can be defined and used in advanced commands in two ways:

- By location (defining a path)
- Embedding

By location

The plugin is located by its path on the client's machine. This is a solution for IT managers that distribute, install, update and uninstall software applications remotely. The **LeoPlugInLoader** will try to load the potential plugin into a dedicated application domain, inspect all published methods for existence of a specific expression's method, and validate its signature. In case of any error during the initialization or invocation phases, the expression ends silently, but a verbal message is written to a log. This solution simplifies updating of a particular plugin without involving Kryon Studio, unless there are changes in the method signature.

NOTE

Distribution of the external plugin done by the organization’s IT department. When executing a wizard that uses an external plugin, the Kryon platform will assume that the required DLL files are already located on the client machine and will try to run them.

Embedding

The plugin is embedded into the particular script's assets. This solution (installation free) distributes the plugin and is used by the expression on demand. As the plugin developer, you will be responsible for the content and its dependencies. There is no reference validation on the Kryon platform’s behalf. The **LeoPlugInLoader** will try to load the plugin into a dedicated application domain, so there is no need for published methods inspection or its signature validation. If there are any errors during the initialization or invocation phases, the expression ends silently, but a

verbal message is written to a log. The updating of a particular plugin can only be done through the Kryon Studio interface and by redefining its in/out parameters.

NOTE

Distribution of the external plugin files is handled automatically by the Kryon platform. When running a wizard that uses the external plugin, the wizard itself contains the plugin files, so that IT does not need to distribute the plugin files to the client machines.

Developing a Kryon plugin

To create a Kryon plugin:

1. Add a reference to Leo.Plugin.Library.dll
2. Mark assembly as Kryon plugin
3. Derive from LeoPlugIn
4. Implement LeoPlugIn
5. Add plugin to the plugin library
6. Apply plugin to individual methods

Add a reference to Leo.Plugin.Library.dll

Use the **Add Reference** dialog box to make **LeoPlugIn** a reference in your project.

Mark assembly as Kryon plugin

Add an assembly attribute in the plugin assembly, the most obvious place being the **AssemblyInfo.cs** file.

```
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;

[assembly: LeoPlugInAttribute("PlugInA",
"SamplePlugIns.PlugInA")]
```

This line of code indicates that the assembly is a plugin. The plugin name is **PlugInA** and that the qualified entry type (the class that inherits from the **LeoPlugIn**) is **SamplePlugIns.PlugInA** (includes namespace).

Therefore, make sure there is a full match between your class name, namespace and **LeoPlugInAttribute** definition.

Derive from LeoPlugIn

To make this class a Kryon plugin, make sure that it derives from a **LeoPlugIn** parent class in the namespace **Kryon.Leo.Infrastructure.Plugin.Library**.

```
using Kryon.Leo.Infrastructure.Plugin.Library;
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;

namespace SamplePlugIns
{
    public class PlugInA : LeoPlugIn
    {
        public PlugInA
        {
        }
    }
}
```

Implement LeoPlugIn

Basically, there is an interface that must be implemented by the plugin implementer and a plugin manager will be able to interact with loaded assembly. The parent **LeoPlugIn** parent implements the interface and marks all signatures of methods, delegates or events as abstract in sake of concrete implementation.

When creating your own logic:

- To initialize your own objects, use the initialize method, which is invoked before the execution process
- To clean up/dispose your own objects, use the shutdown method, which is invoked prior to the Kryon plugin disposal

```
public abstract class LeoPlugIn : MarshalByRefObject,
ILeoPlugIn
{
    public abstract string Name { get; }
    public abstract void Initialize();
    public abstract void Shutdown();
List<LeoPlugInParams> Execute(LeoPlugInMethod pMethod);
}
```

- **Name:** The runtime name of plugin as you defined it
- **Initialize (can be overridden):** Method is called after plugin instantiation
- **Shutdown (can be overridden):** Called before disposing the plugin
- **Execute:** Encapsulates the invocation of a concrete method and is responsible for input arguments and return values

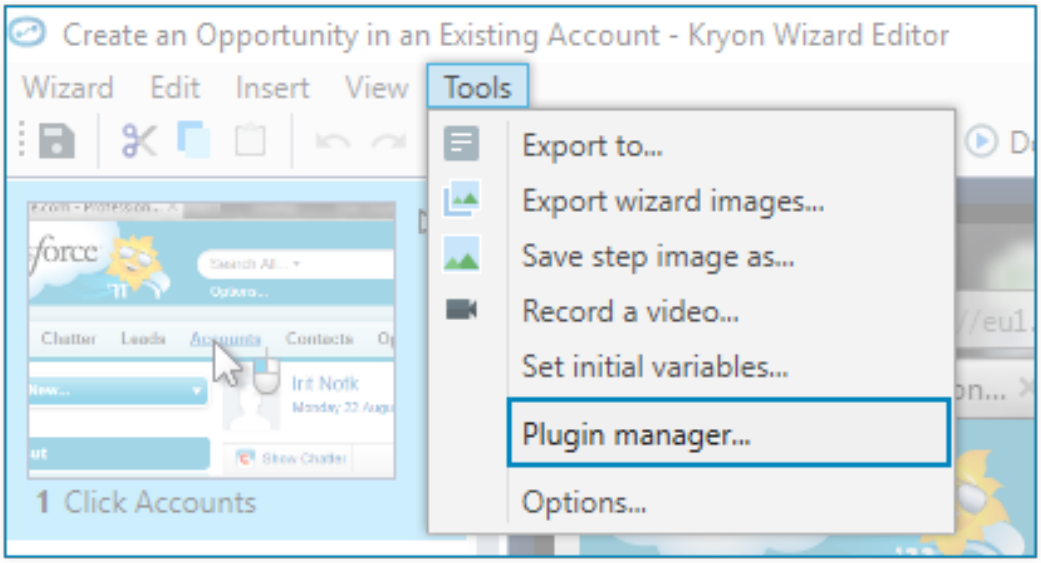
```
using Kryon.Leo.Infrastructure.Plugin.Library;
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;

namespace SamplePlugIns
{
    public class PlugInA : LeoPlugIn
    {
        public override void Initialize()
        {
            Console.WriteLine(Name + "
```

```
                Initializing...");
            }
        }
}
```

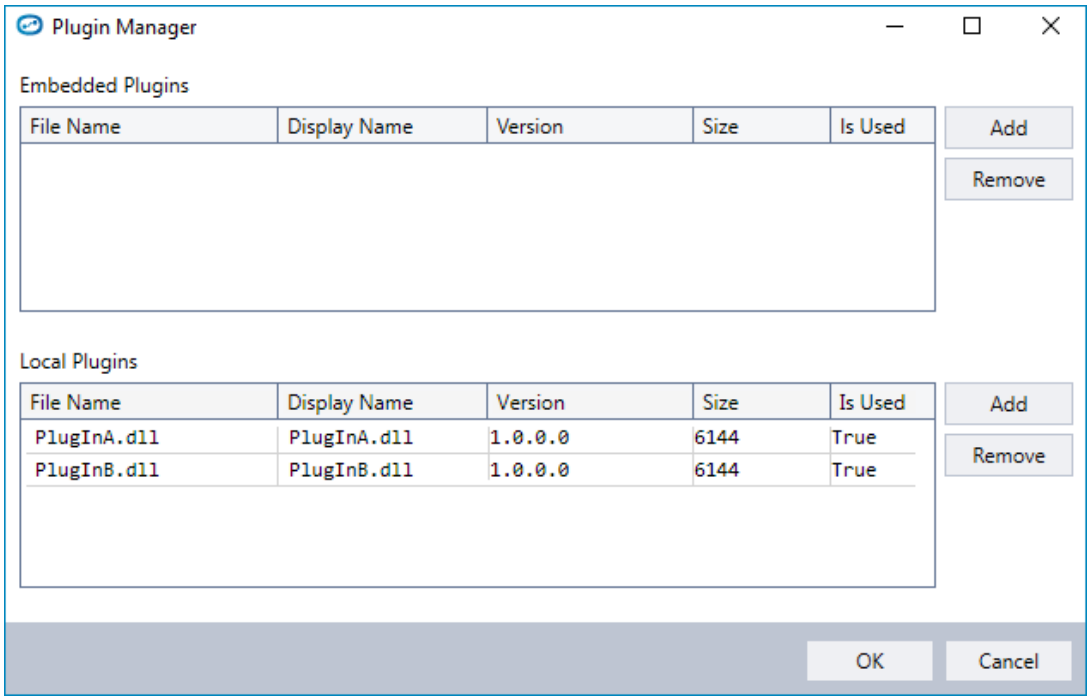
Add plugin to the plugin library

Follow these steps to add a plugin to the plugin library:

1. From the Kryon Studio Editor window, go to **Tools > Plugin Manager**



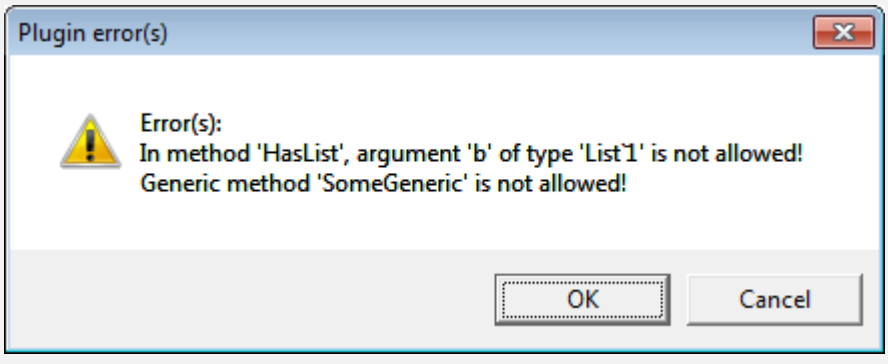
The **Plugin Manager** window appears.



2. Depending on whether your plugin is embedded or local, click the corresponding pane's **Add** button
3. Navigate to the folder in which your DLL plugin is saved and select it
4. Your plugin is added and can be used from the **Run .NET Plugin Method** advanced command

CAUTION

If the following error appears, it indicates that the plugin you are trying to add uses methods that are unsupported by the Kryon Plugin Library framework. To resolve this error, use a different plugin or make sure that the plugin is updated to use methods supported by the Kryon framework.

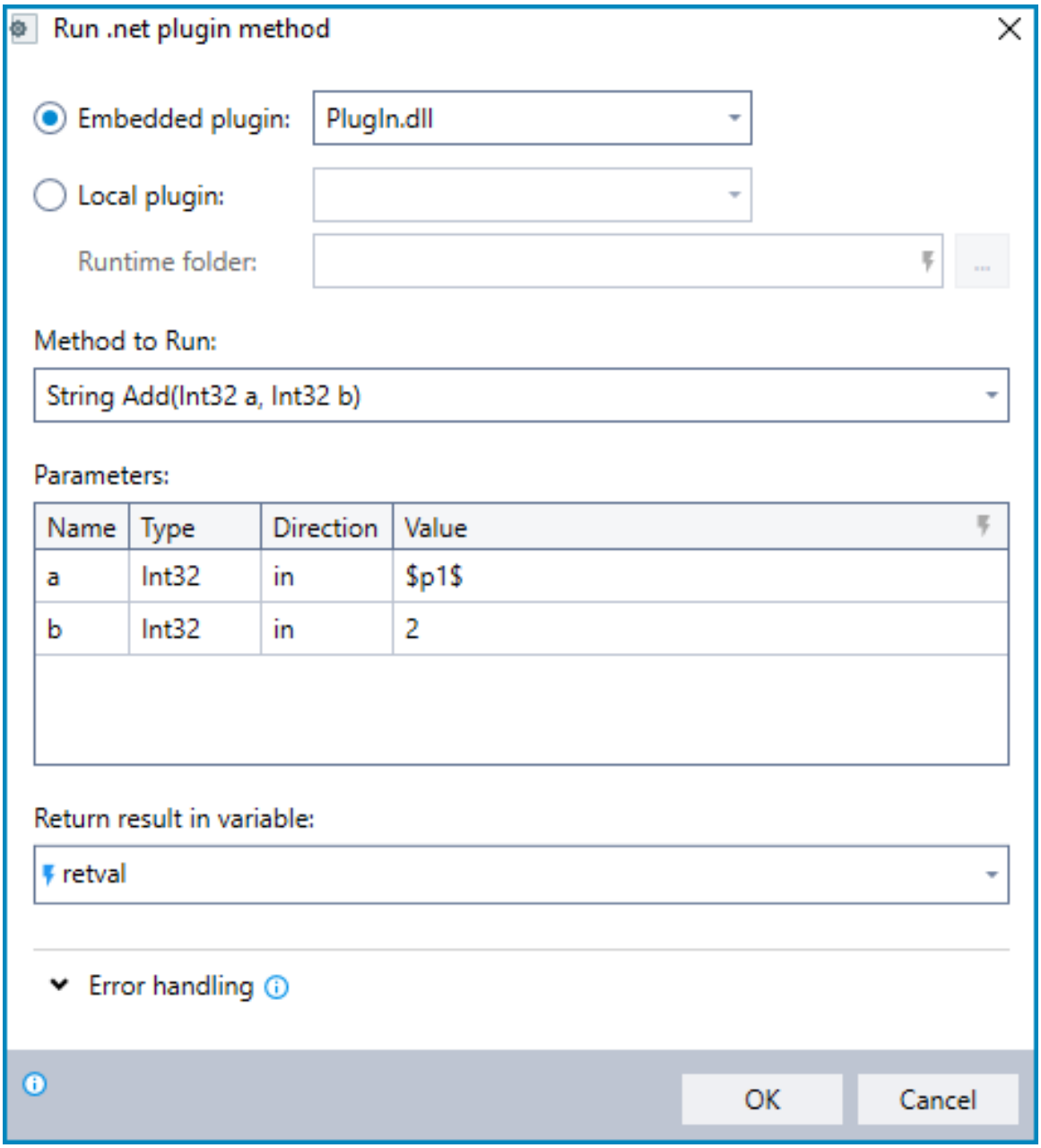


Apply plugin to individual methods

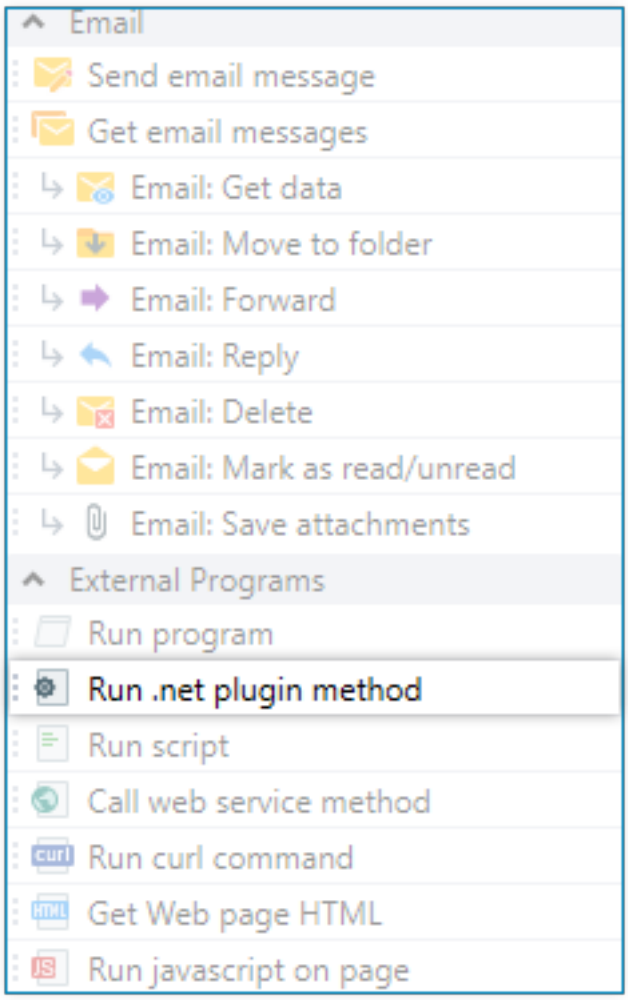
A .NET **LeoPlugInMethod** attribute declaration is all it takes to wire your logic into one or more existing class methods that should be invoked by the Kryon RPA Platform. Simply apply it to individual methods.

```
using Kryon.Leo.Infrastructure.Plugin.Library;
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;

namespace SamplePlugIns
{
    public class PlugInA : LeoPlugIn
    {
        [LeoPlugInMethod]
        public string Add(int a, int b)
        {
            return string.Format("{0} + {1} =
            {2}", a, b, a + b);
        }
    }
}
```

As a result of applying the plugin method attributes, the method appears as an advanced command action in the **Advanced Commands** toolbox:



To launch the method, Kryon Studio users must click the **Run .NET Plugin Method** command in Studio's Advanced Commands Editor:


Email
Send email message
Get email messages
Email: Get data
Email: Move to folder
Email: Forward
Email: Reply
Email: Delete
Email: Mark as read/unread
Email: Save attachments
External Programs
Run program
Run .net plugin method
Run script
Call web service method
Run curl command
Get Web page HTML
Run javascript on page