User Guide

Leo Studio v5

INTELLIGENT ROBOTIC PROCESS AUTOMATION

KRYON™ SYSTEMS

Table of Contents

Preface

Target Audience

This information is intended for automation developers who need to know how to create and maintain content for Leo Studio, and who are familiar with the Leo platform.

About Kryon Systems

Kryon Systems delivers innovative, intelligent Robotic Process Automation (RPA) solutions enabling digital transformation for enterprises. Using patented visual and deep learning technologies, our flagship platform, Leo, allows companies to automate business processes quickly and easily, for immediate productivity gains, near zero error rates, reduced costs and significant ROI results.

The Leo platform supports both virtual and human workforces alike, facilitating the efficient and accurate execution of business processes on any enterprise application. The Leo RPA platform can be leveraged for both unattended (virtual machine) and attended (desktop) automation as well as Hybrid Automation where there is interaction between the virtual and human workforce providing a greater ROI on automation investments and delivering corporate-wide business process improvement.

For more information, visit www.kryonsystems.com.

Copyright Information

1 Introduction to Leo Studio

Leo Studio is the tool with which you can create and manage automation content for Leo robots. Leo automation wizards are recorded by the automation developer in the 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)

1.1 Glossary

Table 1 describes the terms that are essential to understanding the Leo functionality, and which are used throughout this user guide.

Table 1: Glossary

Term	Description
Business Intelligence (BI)	The handling of organizational data. BI systems are capable of retrieving data from various database, such as the Leo database.
Content Author	A Leo Studio user who creates, edits and manages Leo content.
Leo Robotic Process Automation (RPA)	Leo Robotic Process Automation is an emerging form of clerical process automation technology based on the notion of software robots. A software 'robot' (Leo 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 Leo robot. The Leo robot operates on the user interface (UI) in the same way that a human would. In addition Leo's technology can run background activities by a web services, SQL query, DLL and integration technology (such as SAP, based in HTML, .NET, Excel, PDF, Mails, etc.)
Index	A number representing the position of an item in a series or array of items. In Leo, indexes are used in some advanced commands.
Leo Server	Leo is a client-server solution. Leo Players/Robots , running on client desktops/VMs, are connected to a central server (repository) to obtain Leo Wizards in a response to an end-user query. The Leo Server is a central repository that stores all Leo Wizards, collects end-user usage statistics, and manages licenses and permissions.
Leo Catalog	A hierarchical tree of libraries, categories and Leo content for predefined applications.
Leo Category	A Leo library subfolder that contains Leo content.
Leo Content	Leo Wizards and Sensors that are listed in Leo Catalogs.
Leo Library	The top node in the Leo Catalog, containing the categories that contain Leo content.

Term	Description
Leo Sensor	Leo sensors are a powerful guard that intervenes only when a predetermined criteria is met, as if there is a trainer watching over the user's shoulder. Sensitive to the context on the screen and to the user behavior, Leo Sensors are used to push notifications and relevant information, as well as to validate user input and block user errors.
Leo Wizard	Leo Wizards are intelligent predeveloped scripts (macros). On the PSS solution end-users run Leo Wizards on the target application via the Leo Player, whereas on the RPA solution Leo Wizards are run by Leo Robots automatically. Leo Wizards are created using Leo Studio, stored on the Leo Server and run by Leo Player/Leo Robot.
Leo Robot (Player)	A lightweight desktop client that runs Leo's sequence of actions on target applications • Leo Unattended Robot - installed on a virtual machine and runs processes with no human intervention. • Leo Attended Robot - installed on an end-user desktop that runs Leo's sequence of actions on the user's applications

2 Leo Catalog Management

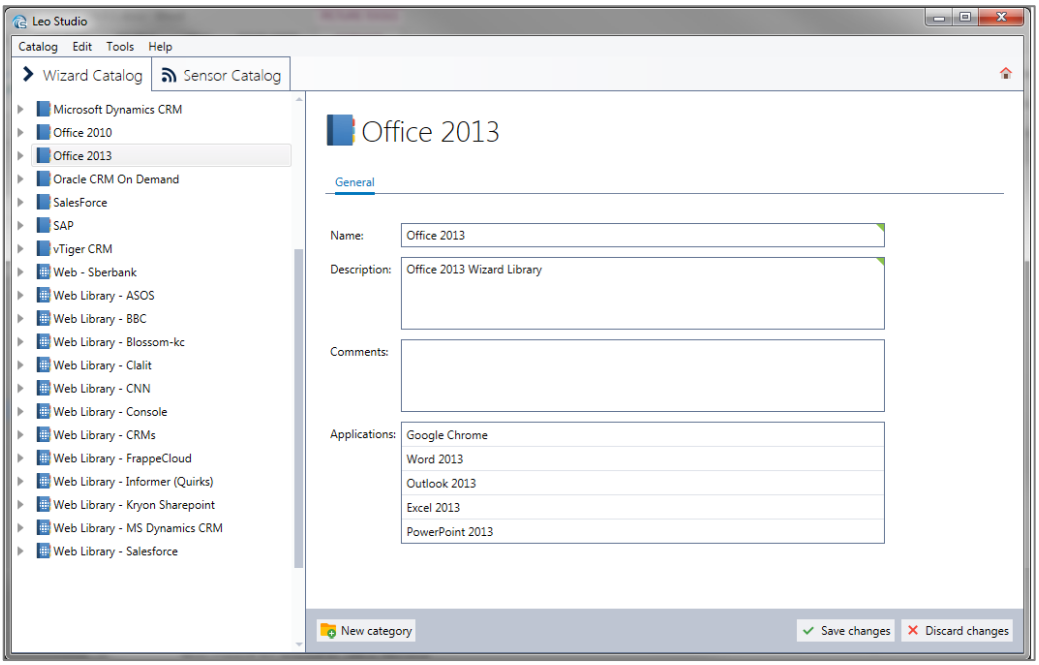
A Leo Catalog is a hierarchical tree made up of libraries, categories and wizards. These catalog levels, their properties, and catalog management procedures are described in the following sections.

2.1 Library Properties

Libraries are the top level in the Leo Catalog hierarchy tree. Each library contains categories and subcategories, which contain Leo Wizards. Each library is associated with predefined desktop or Web applications for which content can be recorded. The supported applications associated with a library are listed in the library's **Applications** list (Figure 5).

You can assign properties to each library such as a name and description, to provide information about the Leo Wizards that the library contains.

Figure 1: Applications in the Library Properties



By default, the Wizard and the Sensor catalogs share an identical library structure.

Libraries cannot be created in Leo Studio. Libraries are created in Leo Admin, where they are associated with their supported applications.

Leo Wizards cannot be created directly under a library, but must be created in a category under that library. Leo Wizards can be moved between categories or within a library, but they cannot be moved to other libraries.

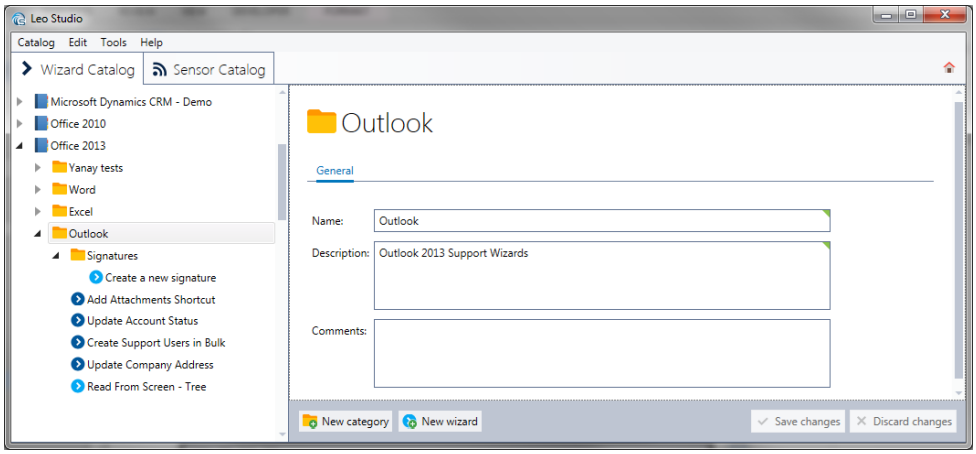
In active libraries, all Leo Wizards in **Published** status appear in the Leo search results and can be accessed by Leo users. In inactive libraries, Leo Wizards of any status do not appear in the Leo search results and cannot be accessed Leo users. Leo Wizards can still be edited in inactive libraries.

2.2 Category Properties

Categories and subcategories are subfolders created in a library in order to organize Leo Wizards within that library.

You can assign properties to each category such as a name and description, to provide information about the Leo Wizards that the category contains (Figure 6).

Figure 2: Category Properties



You can organize categories by:

- **Application Attributes**: For example, an Office 2010 library that supports all the organization's employees, might be organized by the following subcategories:
  - Word
  - Excel
  - PowerPoint
  - Outlook
- **Organizational Attributes**: For example, an Office 2010 library that supports the organization's Training department, might be organized by the following categories:
  - Instructional Designers
  - Trainers
  - Training Team Leaders

Categories in the Wizard and Sensor catalogs are managed separately.

2.3 Wizard Properties

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

- General Properties: The main wizard properties, entered in the **General** tab.
- Embedded Wizards: Information about wizards and sensors that are linked to the selected wizard.
- Notes: A free-form text field where you can type comments. You can use this tab to note the wizard status, testing details, required review, and more.
- Test Cases: This feature is not yet available.
- Changes 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.

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

This applies to Wizards only. All editable Sensor fields are invisible to end users in Leo Player.

Figure 7 shows the editable **Name**, **Description**, and **Keywords** fields in Leo Studio.

Figure 3: Yellow Fields in Leo Studio

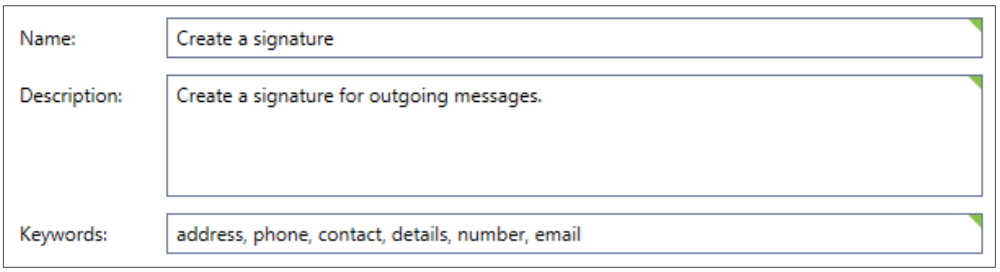


Figure 8 shows how the **Name**, **Description**, and **Keywords** fields appear to users in the Leo Player catalog.

Figure 4: Wizard Properties in the Leo Catalog

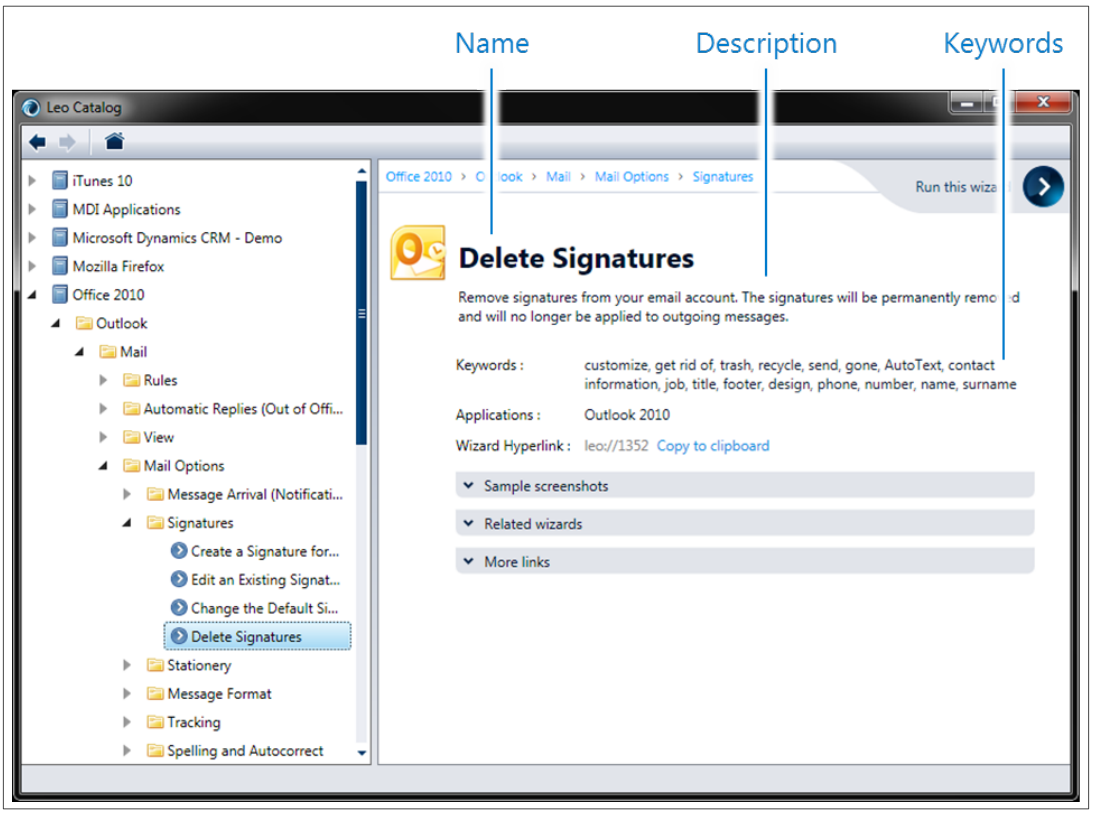
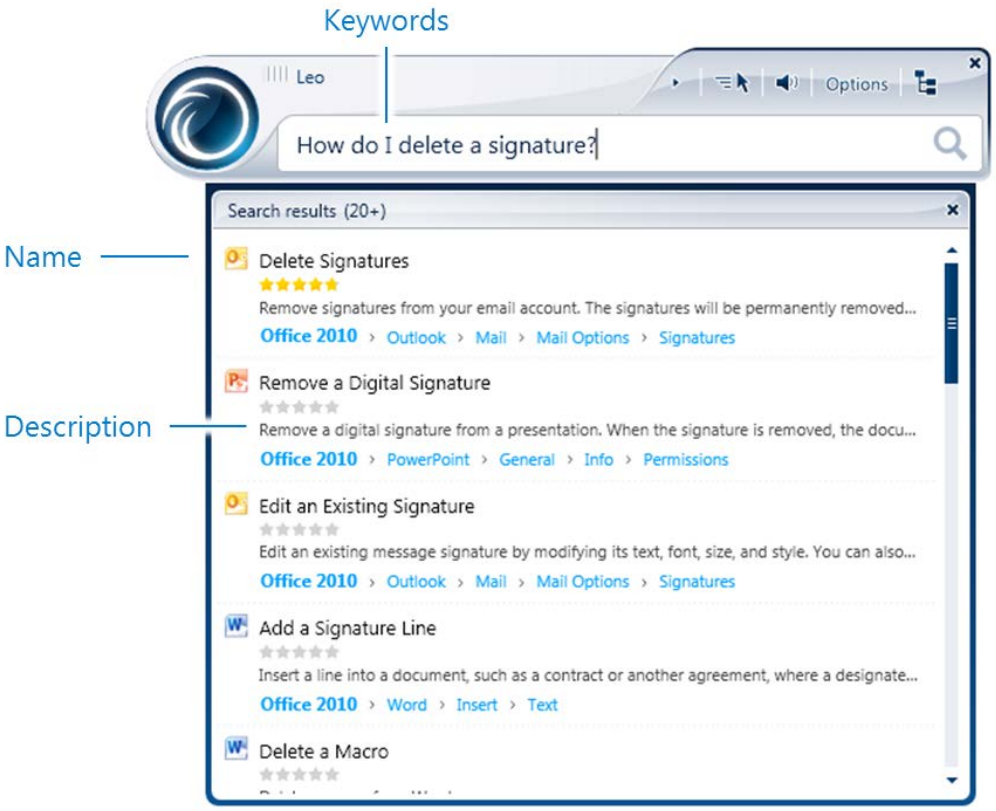


Figure 9 shows how the **Name**, **Description**, and **Keywords** fields appear to users in the Leo Player search results.

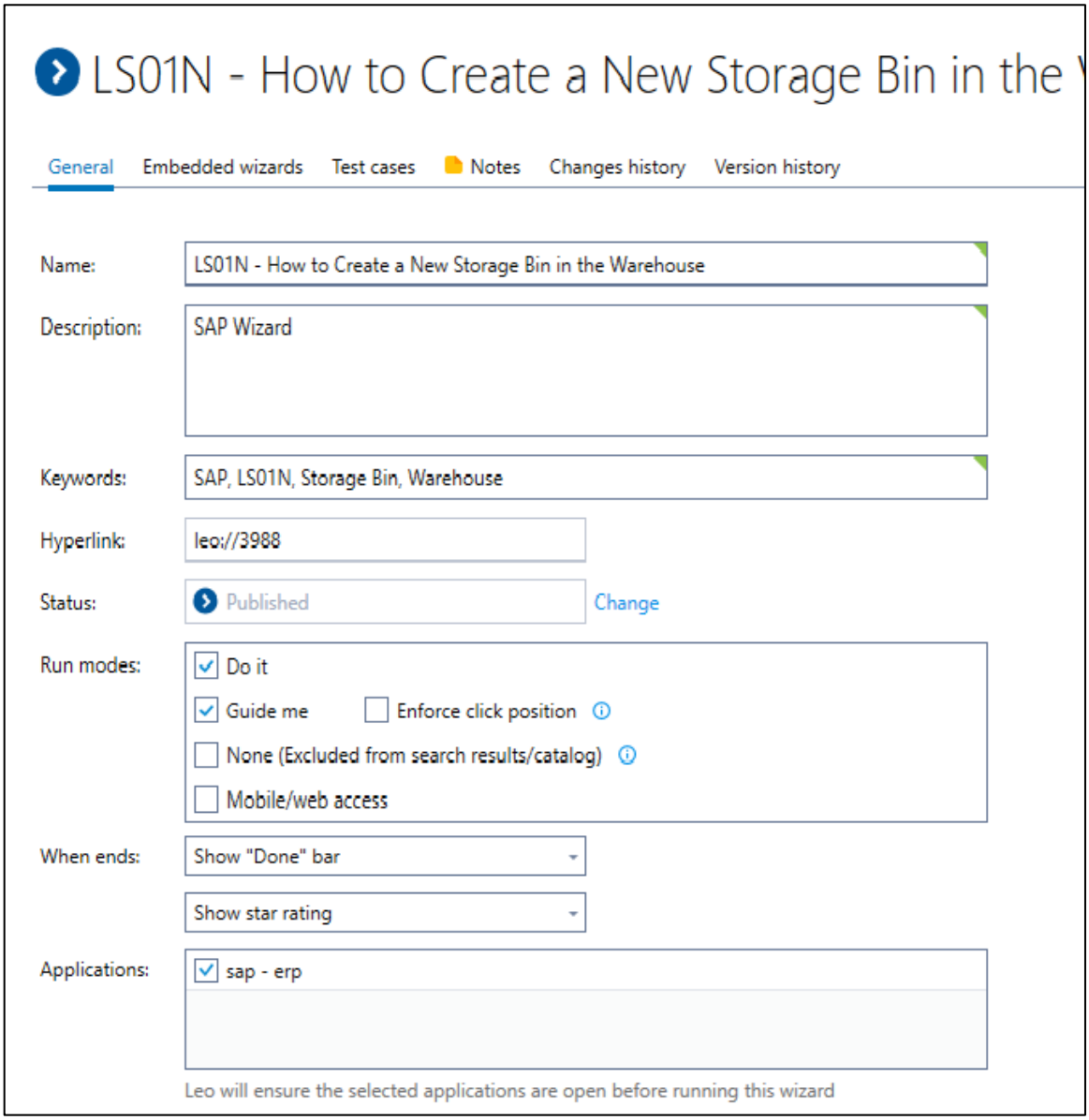
Figure 5: Wizard Properties in the Leo Player Search Results



2.3.1 General Properties

The **General** tab contains the following properties (Figure 10):

Figure 6: General Properties



2.3.1.1 Wizard Name, Description and Keywords

- **Name**: A title that contains the main search words used by the Leo search engine. The wizard name is visible to end users in Leo Player and the Leo Catalog.
- **Description**: A description of what the wizard does. Words in the description are not used by the Leo search engine. The description is visible to end users in Leo Player and the Leo Catalog.
- **Keywords**: Comma-separated search words, used by the Leo search engine to bring up relevant results when users type a search query in the Leo search bar. 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

2.3.1.2 Wizard Hyperlink

A unique Wizard link is automatically generated by Leo for each new wizard (Figure 10). When the link is clicked, it launches a wizard directly without requiring the Leo search bar.

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

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

The Wizard hyperlink is made up of a unique Leo prefix, the wizard ID, and additional parameters that can be added manually (see Table 2).

2.3.1.2.1 Wizard Hyperlink Format

The Wizard hyperlink format is as follows:
Leo://<wizardID>/<runMode>/<stepNumber>/?<varname1>=<value1>&<varname2>=<value2>
Where:

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

For example:

Leo://123

Leo://123/doit

Leo://123/doit/3

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

Leo://123/doit/?firstname= jane&lastname=doe

Leo://123/3/?firstname= jane&lastname=doe

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

2.3.1.2.2 Wizard Hyperlink Components

Table 2 describes each component of a Wizard hyperlink.

Table 2: Wizard Hyperlink Components

Component	Description	Example	Comments
Prefix	The unique Leo hyperlink prefix.	Leo://	Mandatory. Generated automatically.
Wizard ID	A number automatically generated when the wizard is created.	Leo://123	Mandatory. Generated automatically.
Run Mode	Determines whether the wizard will be run directly in Do It or Guide Me mode.	Leo://123/doit Leo://123/guideme	Optional. Added manually. The relevant wizard's Run mode must be enabled accordingly. For information about how to control the wizard Run modes, see Section 2.3.1.4.
Step Number	Starts the Wizard from a specific step.	Leo://123/doit/3	Optional. Added manually.
Variables and Values	Variables that exist in the wizard but whose values are initialized manually by the person who writes or embeds the link.	Leo://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.

2.3.1.3 Wizard Status

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

Leo 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 Leo search results and cannot be accessed by Leo 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 Leo search results and cannot be accessed by Leo users.
- **Published**: The wizard is completed and approved, and can be accessed by users from Leo. The wizard appears in the Leo search results and can be accessed by Leo users.

Only Wizards in Published status are visible to end users in Leo Player.

- **Inactive**: The wizard should not be published. It does not appear in the Leo search results and cannot be accessed by Leo users.
- **Faulty**: The wizard does not work properly and needs to be fixed. It does not appear in the Leo search results and cannot be accessed by Leo users.

When **Faulty** is selected, the **Faulty description** field is enabled for typing comments about wizard issues.

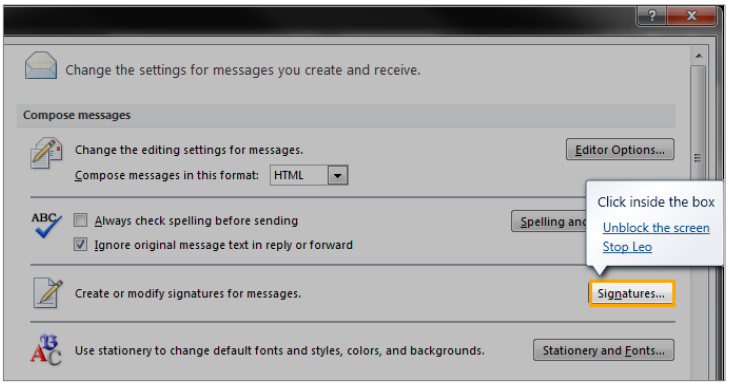
For information about publishing a wizard and changing its wizard status, see Section 3.8.1.

2.3.1.4 Run Modes

Indicates the modes in which users can run the wizard. The run mode options are:

- **Do It**: Indicates whether the wizard can be run in Do It mode.
- **Guide Me**: Indicates whether the wizard can be run in Guide Me mode. 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, Leo provides the user with the option of unblocking the screen (see Figure 11).

Figure 7: **Enforce Click Position** on the End-User's Screen



- **None**: All run modes are disabled and the wizard is excluded from the search results. The wizard is only available to the user as an embedded wizard triggered from within other wizards.

2.3.1.5 When Wizard Ends

- When Wizard Ends

Determines the appearance of certain features when Leo Player is done running a wizard and the Player's **Done** bar appears. The feature options are:

- **Done Bar Display**:
  - Show **Done** Bar: Display the **Done** bar at the end of the wizard run (Figure 12).

    Figure 8: Done Bar

  - Hide **Done** Bar if launched by sensor: Provides a more unobtrusive user experience by disabling the **Done** bar at the end of the wizard run.

For more information about launching wizards from sensors, see Section 4.2.3.

  - Hide **Done** Bar
- **Star Rating Display**:
  - Show Star Rating
  - Hide Star Rating

2.3.1.6 Applications

Supported applications that the Wizard runs on. When a wizard is recorded, Leo identifies the applications used during the recording and adds them to the Applications list.

Leo ensures that all selected applications are open before the wizard is played. Application checkboxes in the list are selected by default.

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.

Supported applications are defined on the Leo Server by the Kryon Systems team or Leo Administrator in your organization. For information about the applications currently supported in Leo, contact the Leo project manager in your organization.

2.3.1.7 Related Wizards

Wizards suggested to the user at the end of a Leo Wizard, which related to the task supported by the completed wizard.

For example, to the Create a Signature wizard you can relate the Edit a Signature and Delete Signatures wizards.

Figure 13 shows related wizards in the Leo Player search bar after the user runs a wizard.

Figure 9: Related Wizards in Leo Player

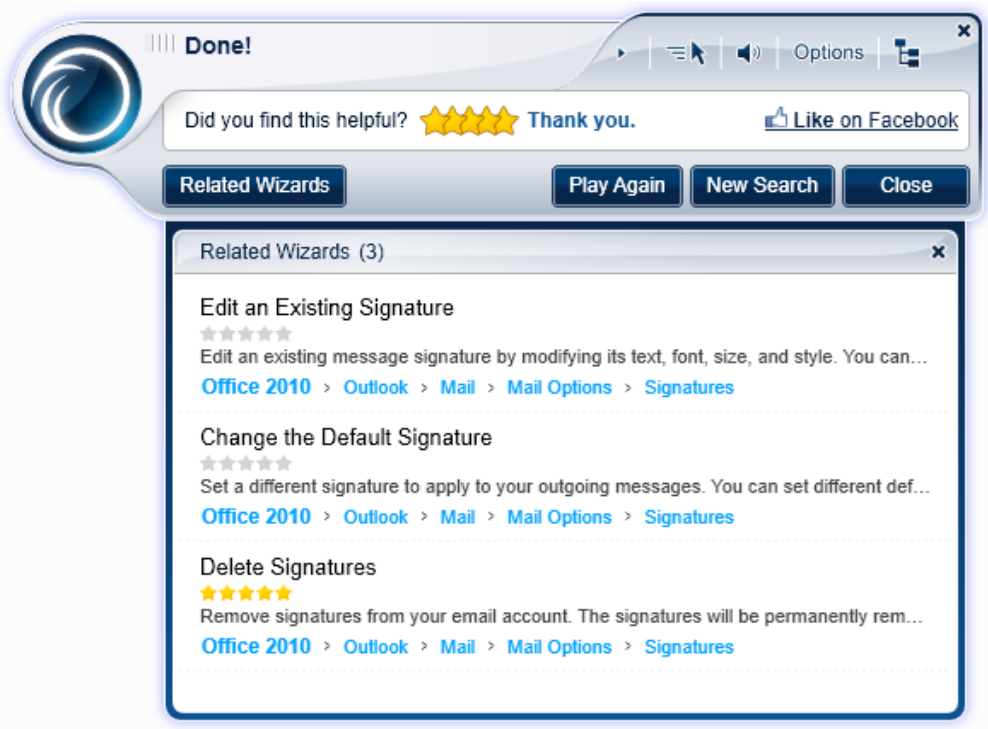
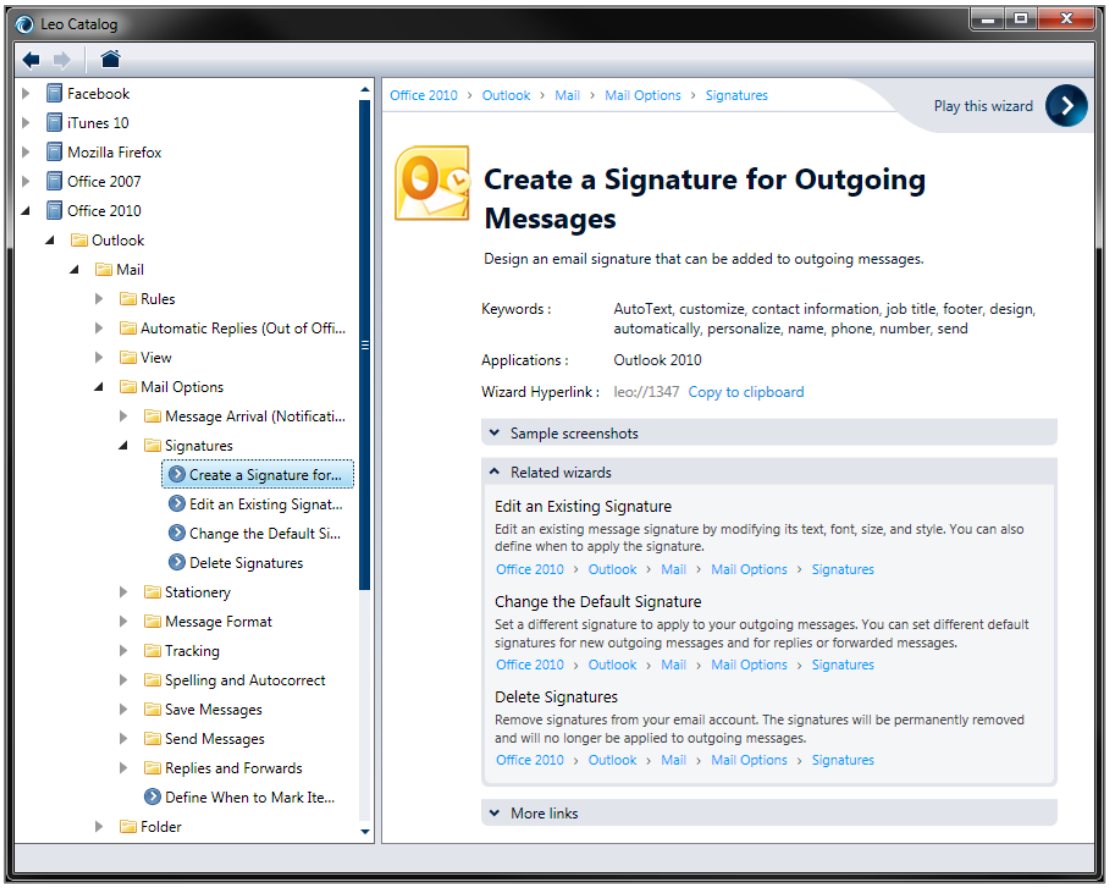


Figure 14 shows related wizards in Leo Player after the user runs a wizard.

Figure 10: Related Wizards in the Leo Catalog



2.3.1.8 External Links

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

For information about adding external links to wizards, see Section 2.6.2.5.

The **External Links** field does not display links added to the wizard flow from the Editor window. For information about inserting links into the wizard flow, see Section 3.6.2.1.3.15.2.

Figure 15 shows external links in the **More links** dropdown list in the Leo Player search bar, after the user selects a wizard.

Figure 11: External Links in Leo Player

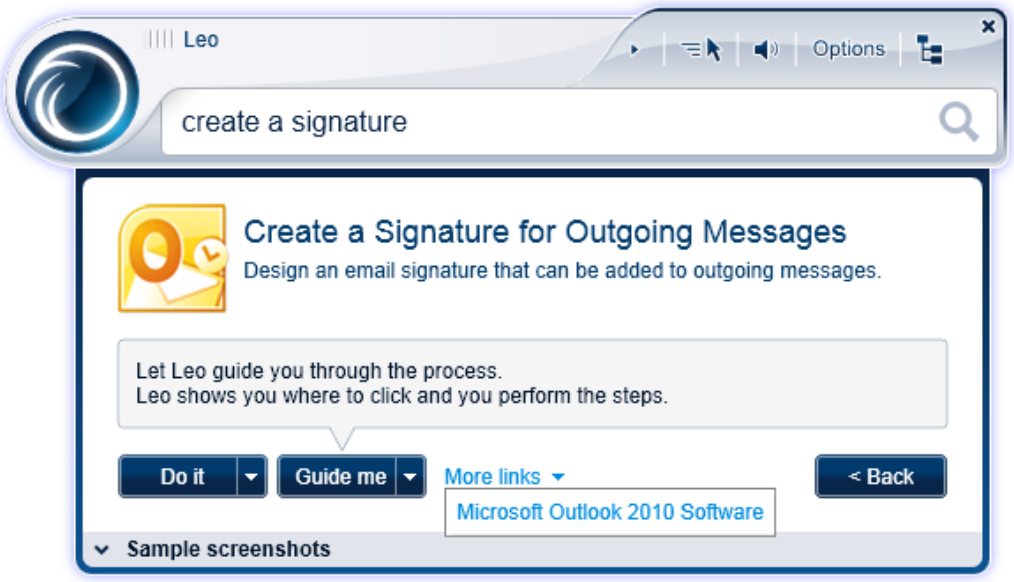
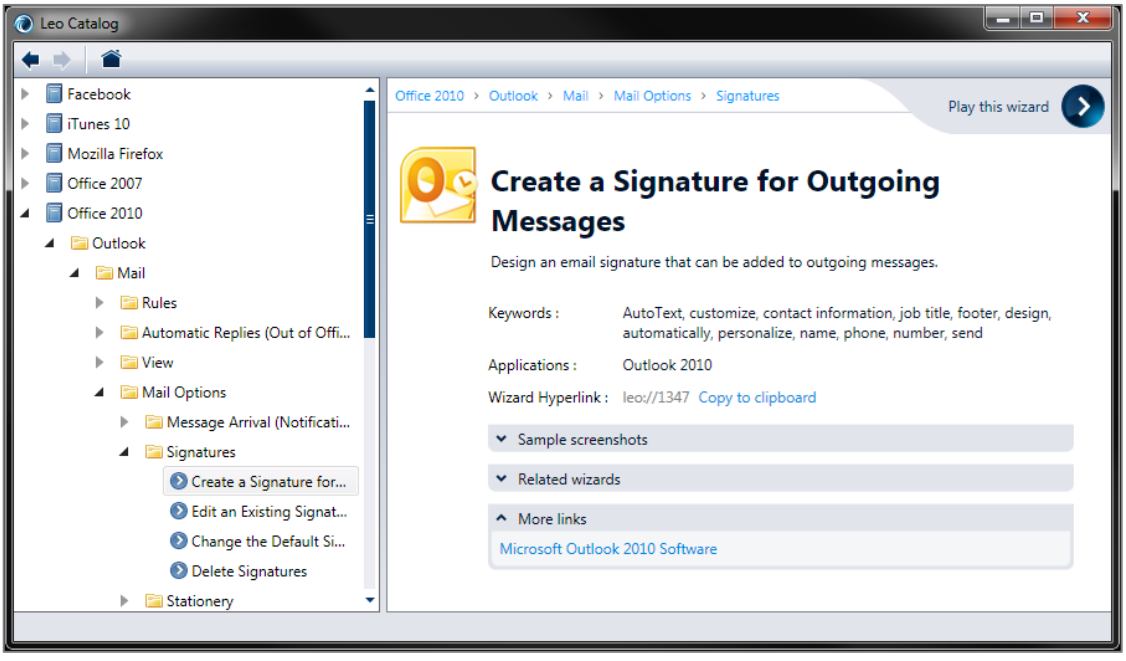


Figure 16 shows external links in the **More links** dropdown list in the Leo Catalog, after the user selects a wizard.

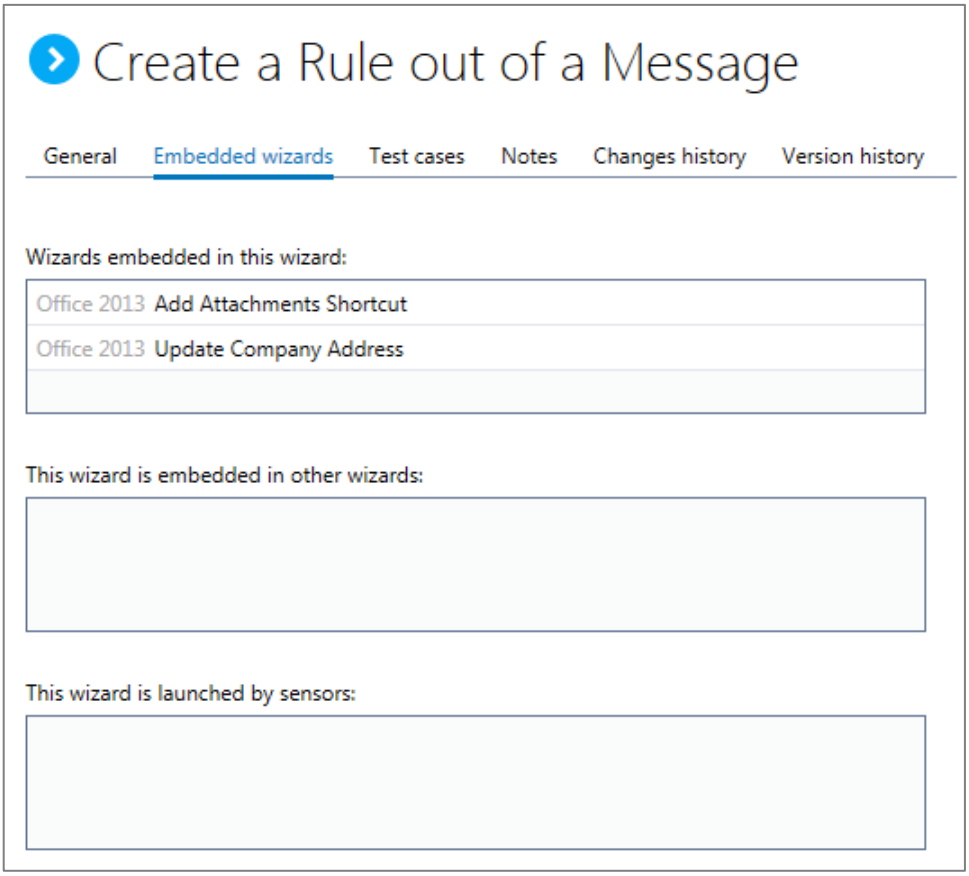
Figure 12: External Links in the Leo Catalog



2.3.2 Embedded Wizards

The **Embedded Wizards** tab displays information about the selected wizard's use within other wizards and sensors, and whether any wizards are embedded in it (Figure 17).

Figure 13: Embedded Wizards



The **Embedded Wizards** tab contains the following information:

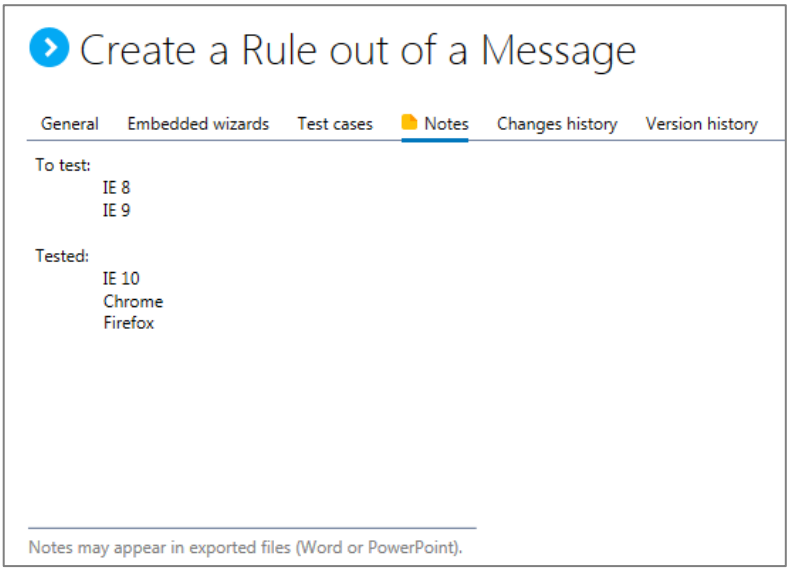
- **Wizards embedded in this wizard**: Lists wizards that are embedded in the selected wizard as building blocks.
- **This wizard is embedded in other wizards**: Lists wizards in which the selected wizard is embedded as a building block.
- **This wizard is launched by sensors**: Lists sensors that launch the selected wizard.

The non-editable Embedded Wizards fields are invisible to end users in Leo Player.

2.3.3 Notes

The **Notes** tab (Figure 18) can be used to type free text and add comments for the selected wizard. The editable **Notes** field is invisible to end users in Leo Player.

Figure 14: Notes



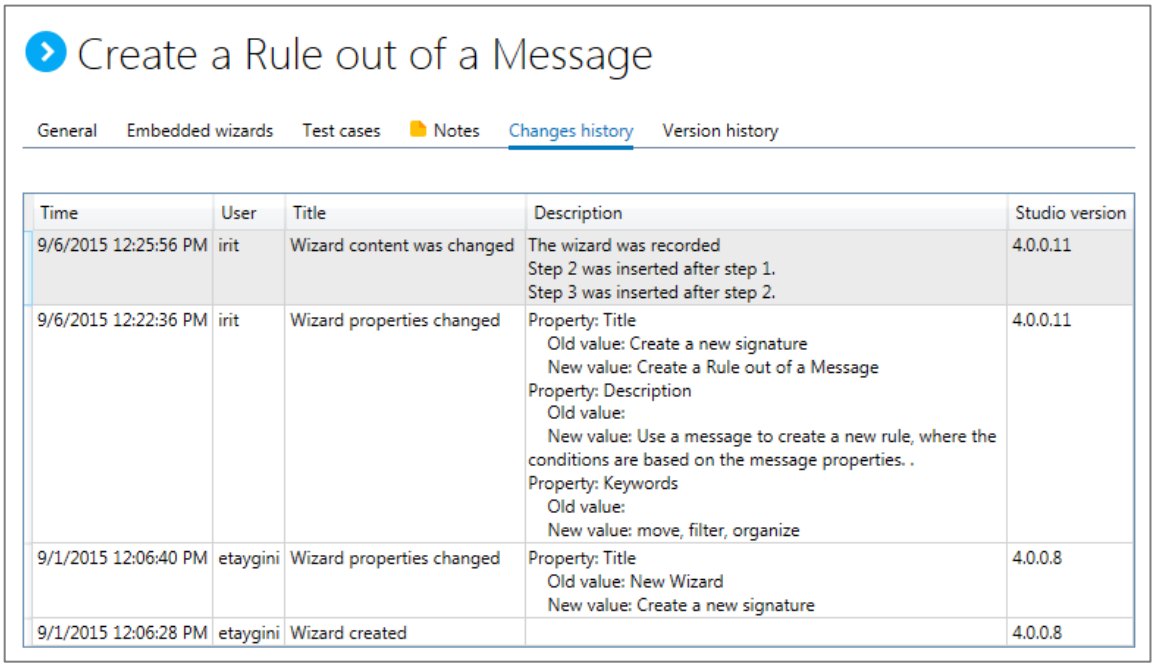
2.3.4 Changes History

The **Changes History** tab contains a table that lists all changes done to the selected wizard's content and properties since it was created (Figure 19). Changes are logged in the table according to the following parameters:

- **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 Leo Studio in which the change was performed

The non-editable Changes History table is invisible to end users in Leo Player.

Figure 15: Changes History



The Changes History table lists the following changes:

- **Wizard Creation**: Creation of a new wizard by the user

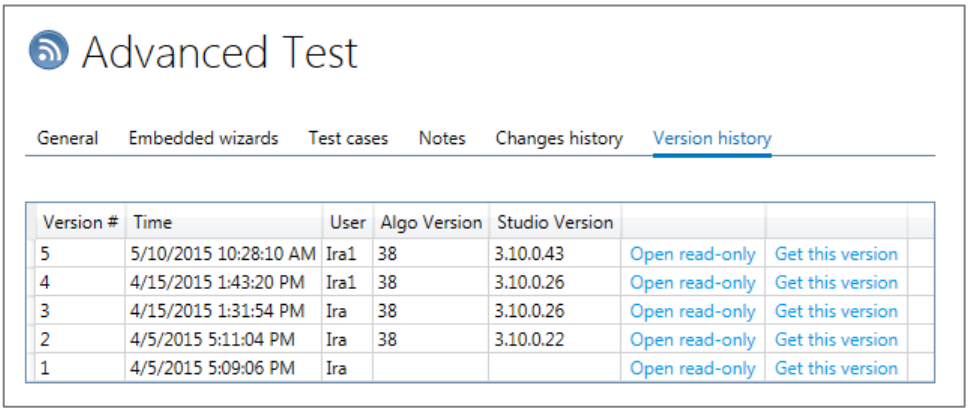
- **Wizard Content**: Any changes to the wizard content such as steps and bubbles, or if the wizard is imported from a local drive to the database.
- **Wizard Properties**: Any changes to wizard properties such as the name, description, and keywords.
- **Wizard Migration**: When a new version of Leo Studio is released, all wizards are migrated to the new version. This migration is logged in each wizard's changes history as a change performed by the Migration tool.
- **Wizard Category**: The change performed when the wizard is moved from one category to another in the Leo Catalog.

The Changes History table can be sorted by clicking the header of a specific column.

2.3.5 Version History

The **Version History** tab contains a table that lists the previous versions of a wizard (Figure 20).

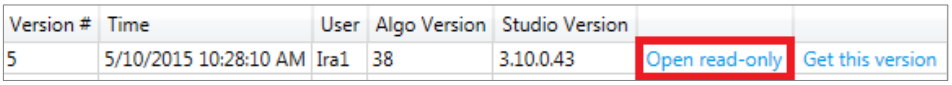
Figure 16: Version History Tab



This table enables you to:

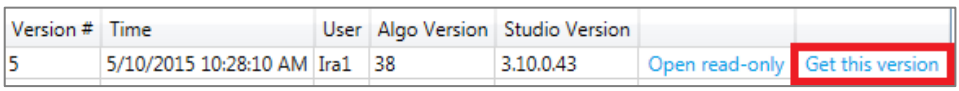
- Open a read-only copy of a previous version by clicking the **Open read-only** link (Figure 21).

  Figure 17: Open Read-Only Link

- Open a previous version for editing by clicking the **Get this version** link (Figure 22).

  Figure 18: Get this Version Link

The Version History table lists the following version details:

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

The Version History table can be sorted by clicking the header of a specific column.

By default, Leo saves and displays the last 10 versions in the **Version History** tab. The maximum number of versions to save can be configured in the Leo Server.

Opening a previous version does not automatically override the current wizard version. Override only occurs if you save the previous version to the database.

When you save a previous version to the database, the following occurs:

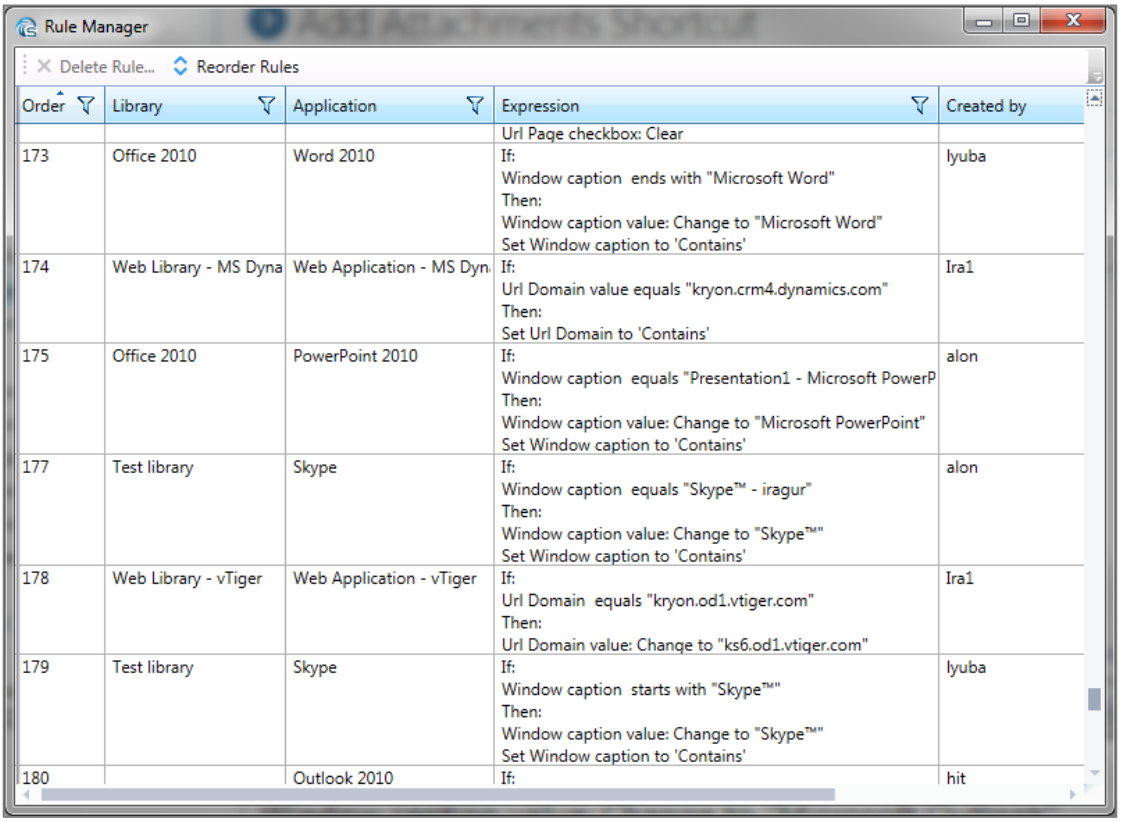
- The previous version overrides the current wizard version.
- The current wizard version being overridden is saved in the version history as a previous version.
- The previous version becomes the current version.

2.4 Wizard Rules

Leo enables you to apply rules and conditions to the window settings in a wizard. The **Rule Manager** dialog box enables you to view all the rules created for a specific library. It also enables you to reorder and delete rules, and to sort and filter the rule table.

Figure 23 shows the **Rule Manager** dialog box.

Figure 19: Rule Manager Dialog Box



The rule table in the Rule Manager dialog box contains the following columns:

- **Order**: The order by which the rules are applied
- **Library Name**: The library that the rule applies to
- **Application Name**: 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 at which the rule was created

For information about how to manage wizard rules, see Section 2.6.3.

For more information about wizard rules and how to add them, see Section 3.6.2.2.2.4.

2.5 Leo Usage Reports

The Leo 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.

For a breakdown of all available reports, go to Section 5.3.

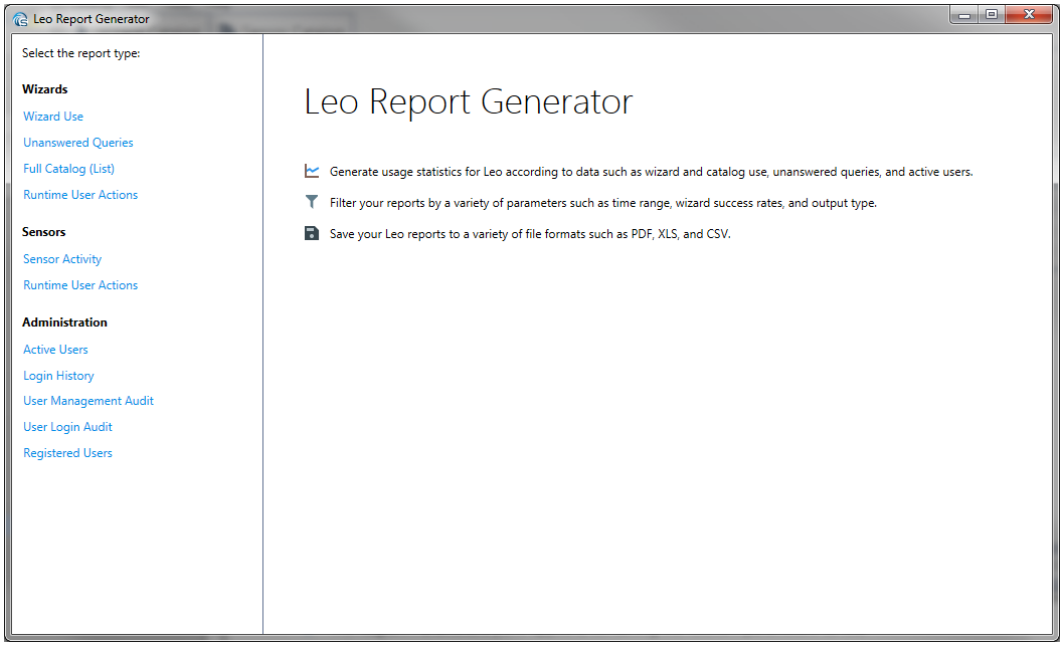
The available Leo reports are:

- **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 Leo 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 Leo users over a specific time range
  - User Management Audit: Displays the actions performed on user accounts in Leo 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.

Generate a report by doing the following:

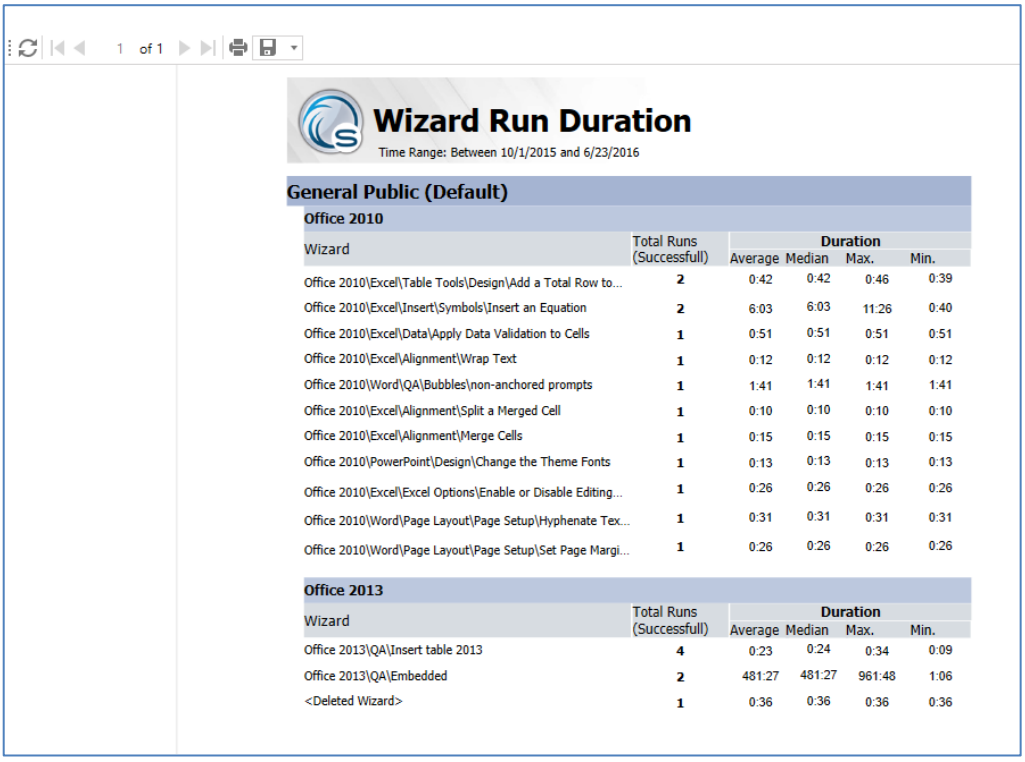
1 From the Leo Studio menu bar, select **Tools > Report Generator**.
2 The **Leo Report Generator** window appears (Figure 24).

Figure 20: Leo Report Generator Window



1 In the **Report Library** pane, select the type of report to generate.
2 The **Report Parameters** pane appears, containing the relevant parameters for the selected report.
3 In the **Report Parameters** pane, select the parameters by which to filter your report.
4 Click **Get Report**.
5 The report you selected appears in the **Report Viewer** pane, filtered by the specified parameters.

Figure 21 - Report Generation Results



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

To return to the **Report Library** pane and select a different report type, click **Back**.

Use the **Report Viewer** toolbar at the top of the **Report Viewer** pane to navigate, save, or print your report (Figure 26).

Figure 22: Report Viewer Toolbar

2.6 Catalog Management Procedures

The following sections describe how to manage a Leo Catalog by handling its content, structure and properties.

2.6.1 Creating a Category

Create a new category or subcategory by doing the following:

1. From Leo Studio, in the **Catalog** pane, select a library or, if it contains existing categories, select the category in which you want to create a new category or subcategory.
2. Do one of the following:
   - From the Leo Studio 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 (Figure 6), do the following:
   - To change the category name, type a name in the **Name** field.
   - To change the category description, type the description in the **Description** field.
   - In the **Comments** field, type any relevant comments you have about the category.
4. Save your changes by clicking **Apply**.

2.6.2 Creating a Wizard

The creation of a new Leo Wizard consists of the following procedures:

1. Adding a New Wizard to the Leo Catalog: Leo Wizards are created in categories in Leo Studio and can then be assigned properties and recorded.
2. Editing Wizard Properties: Properties such as the Leo Wizard name and description are assigned in the Studio window.
3. Opening a Wizard: Wizards are recorded and edited in the Wizard Editor window.
4. Adding Related Wizards: Related wizards are a list of wizards suggested to the user that are related to the wizard the user has just finished playing.
5. Adding External Links: External links are hyperlinks that direct the user to Web pages, knowledge management systems, or files that are external to the Leo platform.

2.6.2.1 Adding a New Wizard to the Leo Catalog

Add a new Leo Wizard to the catalog by doing the following:

1. From Leo Studio, in the **Wizard Catalog** tab, select the category in which to create the wizard.
2. Do one of the following:
   - From the Leo Studio menu bar, select **Catalog > New Wizard**.
   - Right-click the category and select **New Wizard**.

   A new Leo Wizard is added to the Leo Catalog.

2.6.2.2 Editing Wizard Properties

Edit the general Leo Wizard properties by doing the following:

1. From Leo Studio, in the **Wizard Catalog** tab, select the wizard whose properties you want to edit.
2. In the **General** tab, type the wizard name.

Users who access wizards from Leo Player 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 properties. To cancel your changes, click **Discard**.

2.6.2.3 Opening a Wizard

Wizards are opened, recorded and edited in the Wizard Editor. Access the Wizard Editor by doing the following:

1. From Leo Studio, in the Wizard Catalog, select the wizard that you want to record or edit.
2. Do one of the following:
   - At the bottom of the **General** tab, click **Edit Wizard**.
   - Press **CTRL+E**.
   - From Leo Studio, in the **Catalog** pane, right-click the wizard and select **Edit Wizard**.

   The **Wizard Editor** window opens (see Figure 4).

If the wizard is currently open on another user's machine, the wizard is checked out and locked for editing. To read more about Leo's Check In mechanism and how to use it, see Section 3.7.5.3.

To access and work on several Leo Wizards at the same time, open multiple Editor windows.
To cancel the wizard opening, click the X on the wizard progress bar.

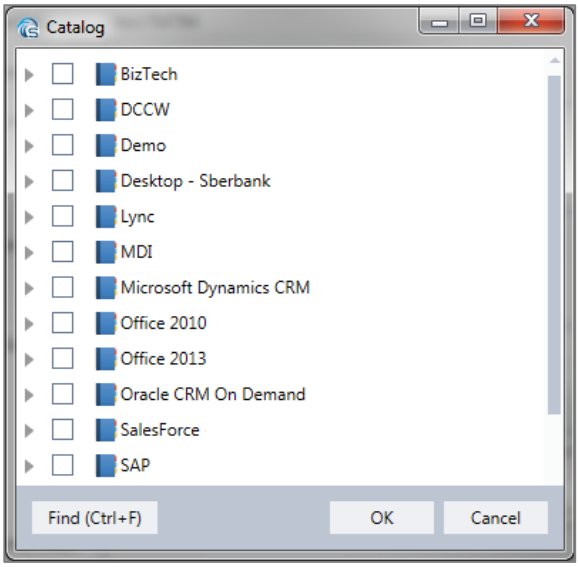
2.6.2.4 Adding Related Wizards

Add related wizards to a wizard by doing the following:

1 From Leo Studio, in the **Catalog** pane, select the Leo Wizard to which you want to add related wizards.
2 In the **General** tab, under the **Related Wizards** field, click **Add**.

The **Catalog** dialog box appears (Figure 27).

Figure 23: Catalog Dialog Box



3 From the **Catalog** dialog box, select the checkboxes for wizards that you want to add. You can select multiple wizards.

To search for a specific Leo Wizard, click **Find** or press **CTRL+F** and then type your search query.

4 Click **OK**.

The wizards you selected are added to the **Related Wizards** field in Leo Studio (Figure 10). If the wizard is in Published status, the related wizards also appear in the Leo Catalog and Player.

When a wizard is related to another wizard, both wizards are automatically added to each other's **Related Wizards** field.

5 Save your changes by clicking **Apply**.

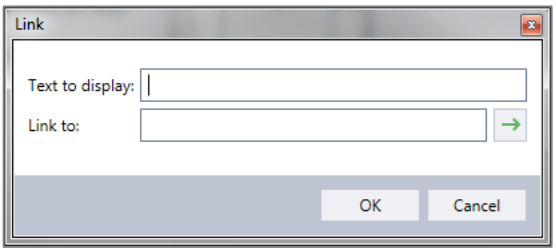
2.6.2.5 Adding External Links

Add external links to a wizard by doing the following:

1 From Leo Studio, in the **Catalog** pane, select the Leo Wizard to which you want to add external links.
2 In the **General** tab, under the **External Links** field, click **Add**.

The **Link** dialog box appears (Figure 28).

Figure 24: Link Dialog Box



3 In the **Link** dialog box's **Text to display** field, type the text that you want to display for the link.
Example:
Microsoft Outlook 2010 Software

4 In the **Link to** field, type or copy and paste the link address.
Example:
http://office.microsoft.com/en-us/outlook/

To go to the address you inserted, click the **Go** button.

5 Click **OK**.
The link you typed is added to the **External Links** field in Leo Studio (Figure 10). If the wizard is in Published status, the link also appears in the Leo search bar (Figure 15) and Catalog (Figure 16) in Leo Player.

To edit or remove a link from the **External Links** field, select it and click the **Edit** or **Remove** button, respectively.

6 Save your changes by clicking **Save Changes**. To cancel your changes, click **Discard Changes**.

2.6.3 Managing Wizard Rules

The following sections describe how to manage the Wizard Rules table:

- Accessing the Rule Manager
- Filtering the Rule Manager Table
- Reordering Rules
- Deleting a Rule

2.6.3.1 Accessing the Rule Manager

Access the Rule Manager by doing the following:

1 In the Leo Studio window, click **Tools > Rule Manager**.
2 The **Rule Manager** dialog box appears (Figure 23).

2.6.3.2 Filtering the Rule Manager Table

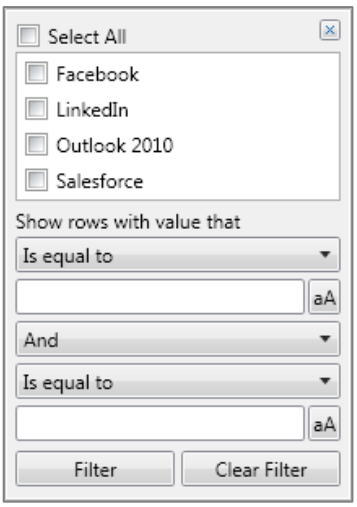
The Rule Manager lists a table of all rules that exist for your company. The Rule Manager table can be filtered by column, so that only rows containing specific types of information are shown in the manager, while rows containing other types of information are hidden.

Filter the Rule Manager table by doing the following:

1 Access the **Rule Manager** dialog box as described in Section 2.6.3.1.

2 From the column heading by which you want to filter the table, click the **Filter** icon.
The **Filter** dialog box appears (Figure 29).

Figure 25: Filter Dialog Box



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.

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

A preview of the filtered table appears in the manager dialog box.
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 manager dialog box.
8 Click **Filter** or **Clear Filter** to apply or undo the filter that you selected.
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.

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

2.6.3.3 Reordering Rules

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

Reorder rules in the Rule Manager by doing the following:

1 Access the **Rule Manager** dialog box, as described in Section 2.6.3.1.
2 From the **Rule Manager** dialog box, click **Reorder Rules**.
3 At the top of the **Rule Manager** dialog box, the **Reorder Rules** toolbar appears.
4 Select the rule that you want to reorder, and then click the **Move Up** or **Move Down** icon to move the rule.
5 Repeat step 4 for each rule that you want to reorder.
6 On the **Rule Manager** toolbar, save your changes by clicking **Save Changes**.

To visually sort the rule table in the Rule Manager by a specific column, click the relevant column header.

2.6.3.4 Deleting a Rule

When a rule is deleted, it is deleted for **all** wizards in the library.

Delete a rule by doing the following:

1. Access the **Rule Manager** dialog box, as described in Section 2.6.3.1.
2. From the **Rule Manager** dialog box, select the rule that you want to delete, and then click **Delete Rule**.
3. From the confirmation dialog box that appears, click **OK**.
4. The rule is deleted from the rule table.

Deleting or adding rules does not affect existing wizards.

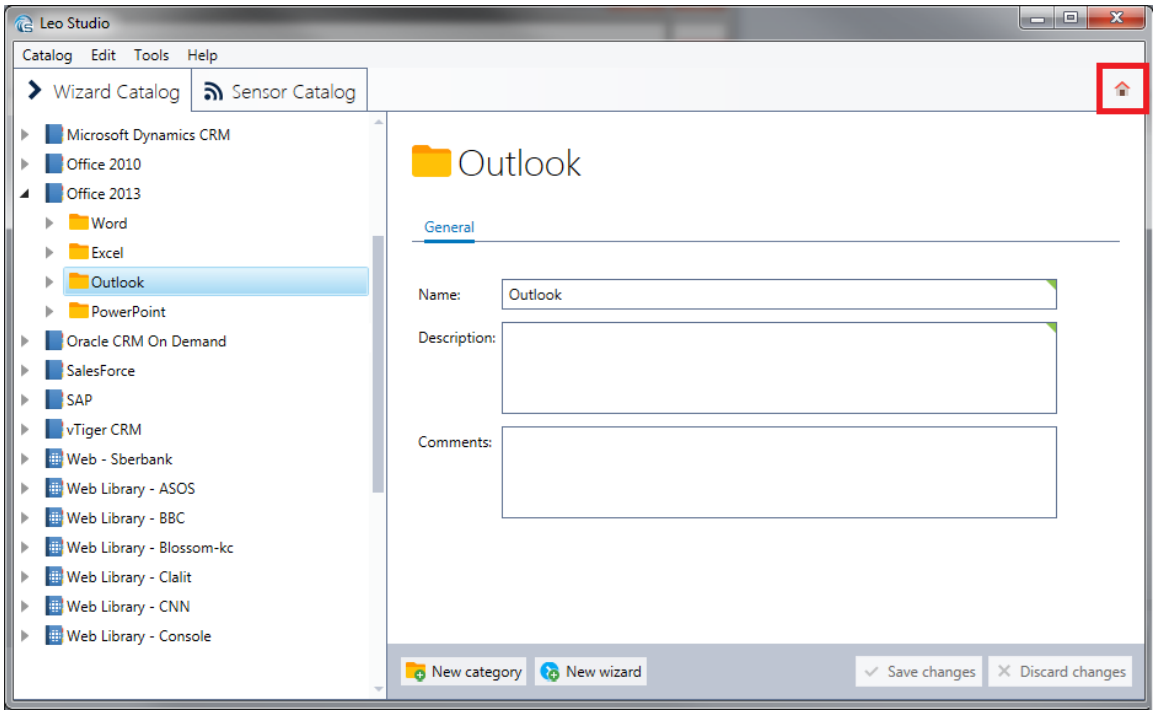
2.7 Restoring the Catalog Home View

Every time you launch Leo Studio, the default **Home** catalog view appears. When you select a catalog item from the Leo Catalog, the **Studio** window displays the selected catalog item's properties in the **Properties** pane.

Restore the Catalog **Home** view by doing the following:

1. From the **Studio** window, click **Home** (Figure 30).

Figure 26: Catalog Home Button



The Catalog **Home** view is restored (Figure 2).

3 Wizard Development

The following sections contain the procedures required to develop new Leo wizards, edit them, test them for quality assurance, and manage them after publication. The Leo Wizards you develop by performing these procedures can then be played on users' environments, applications and data via Leo.

While Leo supports all applications, **some** Leo capabilities may not be available for certain applications.

3.1 What Leo Detects

Leo wizards are made up of the steps and functionality required to perform a process on an application.

Leo's patent-pending technology is based on image and optical character recognition, enabling it to "see" the screen just like the user does. Since Leo relies on screen captures to "see" the user's screen, content is first recorded on the target application, and then edited for functionality so that Leo can correctly execute the process.

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

When recording or running a wizard, Leo detects the following:

- Window Detection: Leo detects properties of the window in which the action is performed, in order to later use this information when the wizard is run on end user environments.
- Object Detection: Visual data within a window such as objects, text, and object position in the window, which indicates to Leo the object it needs to click or detect on the screen.

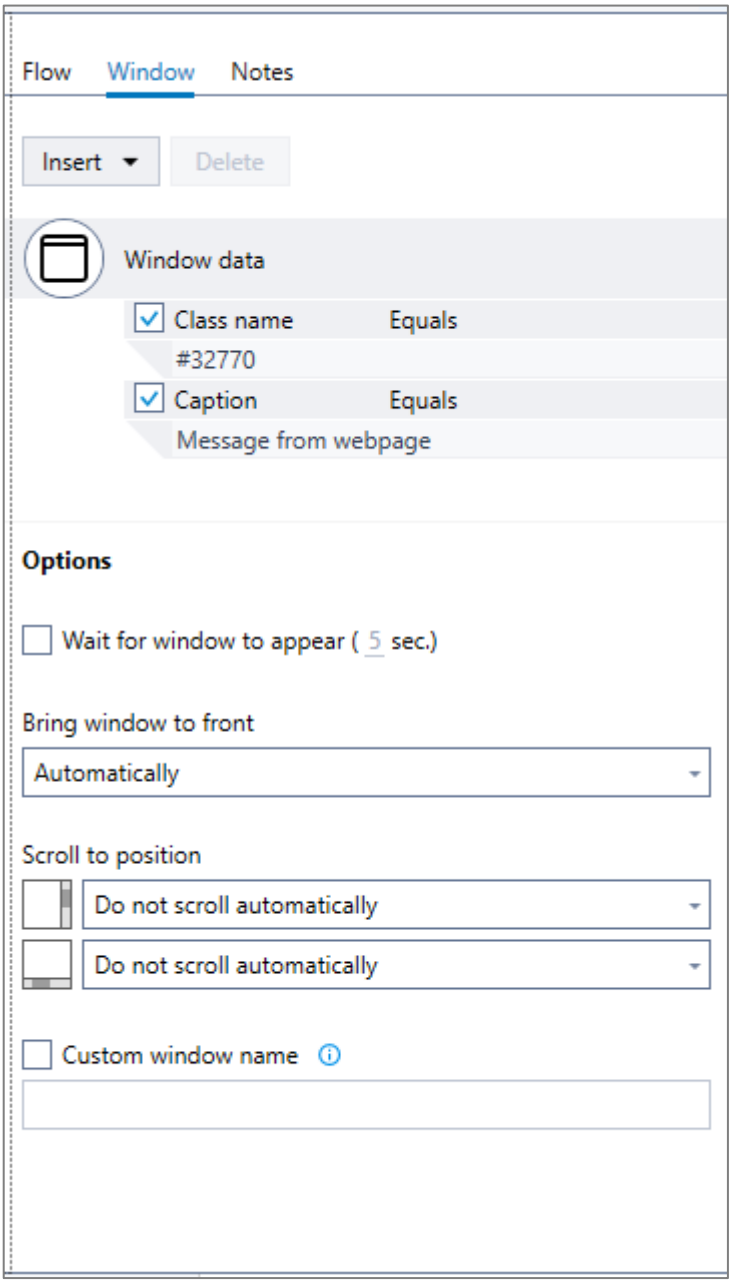
3.1.1 Window Detection

Leo uses window detection to identify the window in which the action is performed. During the recording, Leo stores properties of the window on which you record the core action, and uses these properties to detect the relevant window when the Leo Wizard is played. Window detection occurs for both desktop and Web applications.

3.1.1.1 Window Detection for Desktop Applications

Figure 31 shows the **Window** tab for desktop applications.

Figure 27: Window Tab for Desktop Applications



Each window event for desktop applications 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 into the step's window properties: The **Class name** checkbox is selected and the field contains the class of the window in which the step is performed.

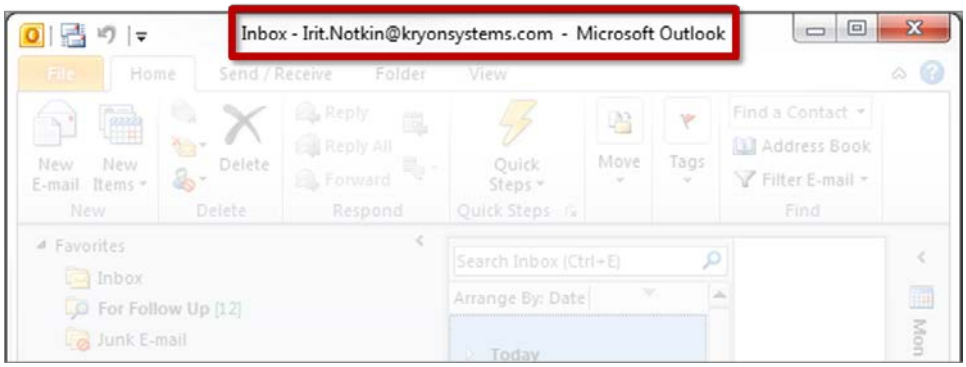
For some windows, the window class differs when you open that window in different operating systems or Web browsers.

For some applications, the window class changes every time you launch that window.

- **Caption**: The text that appears in the window title bar.

Window captions might vary from one user to another, and some captions are user specific (Figure 32).

Figure 28: User-Specific Window Caption

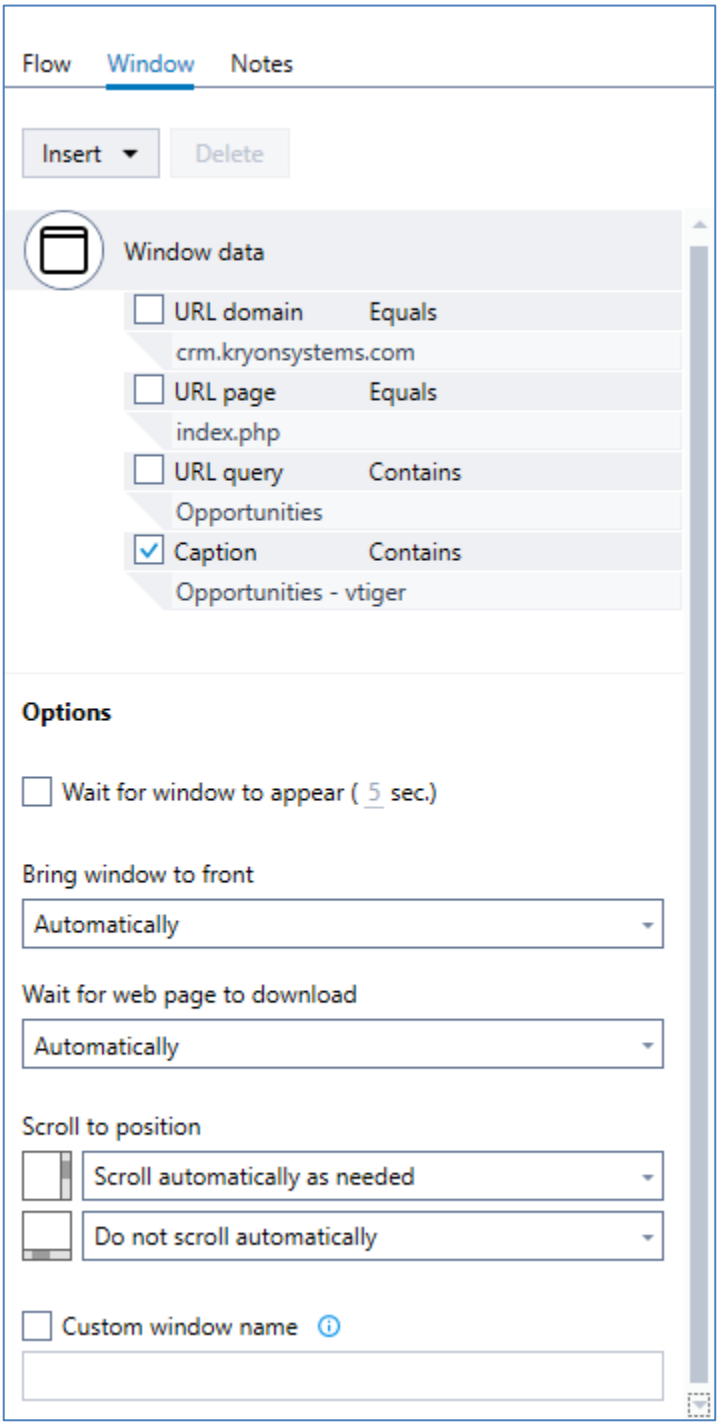


The window caption is captured automatically into the window properties:
The **Caption** checkbox is selected, the dropdown list is set to the **Equals** option, and the field contains the full caption of the window in which the step is performed.

3.1.1.2 Window Detection for Web Applications

Figure 33 shows the **Window** tab for Web browser windows in Web applications.

Figure 29: Window Tab for Web Applications



Each window event for Web applications 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 field contains the website domain name.

- **URL Page**: The name of a page on the website.
  By default, the **URL page** checkbox is selected and the field contains 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 field contains the query string, if available.
- **Caption**: The text that appears in the window title bar. For more information, see Caption.

3.1.1.3 Window Visual Object

An image of an object in the recorded window. If the application has multiple windows with the same window class/caption (i.e. window properties are too generic for Leo to detect the relevant window), Leo searches for an image on the user's screen that is compatible with the captured image in order to identify the window it needs to work in.

Window objects must be selected manually by the content author.

3.1.1.4 Wait for Window to Appear

Allows Leo to wait longer for the window to be displayed on the screen. Specify the maximum amount of time that Leo will wait. If the window loads before the maximum time is up, Leo will continue the flow and will not wait the remaining amount of time.

3.1.1.5 Custom Window Name

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

3.1.1.6 Automatic Scrolling

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

The automatic scrolling options are (Figure 34):

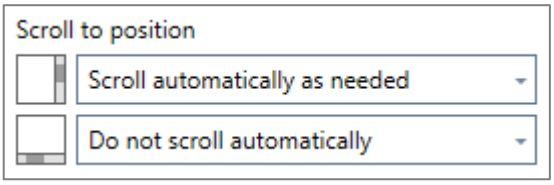
- **Scroll automatically as needed**: Leo detects whether the click position is located outside the area shown in the user's application window, and automatically scrolls up or down as needed to reach the click position.

This is the default setting for recorded Web applications.

- **Do not scroll automatically:** Leo 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.

Figure 30: Automatic Scrolling in the Window Tab



3.1.1.7 Window Detection Rules

Window detection rules enable you to apply the same property conditions and text to all Leo Wizards that share the same window detection properties, such as a window caption in Microsoft Outlook that contains a specific username. You can add a rule from within a Leo Wizard you are currently working on in the Leo Studio Wizard Editor, and then view and manage all rules in the Rule Manager.

For information about how to add window detection rules, see Section 3.6.2.2.2.4.

3.1.2 Object Detection

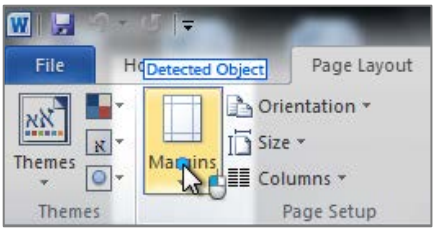
A detected object is an object that was clicked during the recording, or an object inserted or customized by the content author.

The following sections describe detected objects and how to use them:

- Detected Object Types
- Detected Object Display
- Object Detection Criteria
- Object Detection Procedures

Figure 35 shows a detected object in the Wizard Editor window.

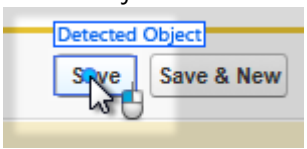
Figure 31: Detected Object in the Wizard Editor



3.1.2.1 Detected Object Types

Leo Studio functionality offers several types of detected objects. In the Leo Studio Editor, 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.

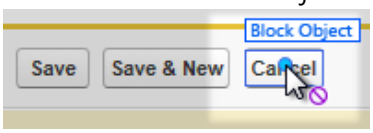
- **Core Action**: The core action is recorded, and captured by default as a click, during the recording. There can be only one recorded click per step, i.e. a single core action.
  Core actions that use a detected object are:
  - Click
  - Hover
  - Detect Object

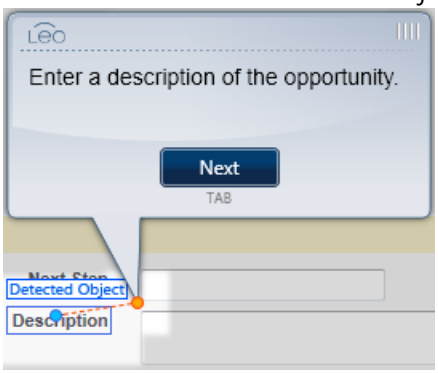
  - Read from Screen

- **Window Visual Object**:
  For information about window visual objects, see Section 3.1.1.3.

- **Object block**:
  For information about object blocks, see Section 3.6.2.1.2.3.1.1.



- **Anchored Object**:
  For information about anchored objects, see Section 3.6.2.1.3.12.



If you are unsure what each detected object in a step does, click the Detected Object box and Leo will highlight the relevant action on the right. If the detected object belongs to an anchored bubble, for example, Leo will highlight that bubble action.

3.1.2.2 Detected Object Display

- Detection Box: A blue bounding box that surrounds the detected object.
- Click: A blue dot that, if no offset is used, represents the cursor click position
- Offset: An orange dot created by offset, that, if used, represents the cursor click position
  For information about click offset, see Section 3.6.2.1.4.9.6.
- Highlight Box: An orange or purple bounding box that represents the highlighted or blocked object.
  For information about the highlight box, see Section 3.6.2.1.4.9.7.

3.1.2.3 Object Detection Criteria

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

Any such object that you expect Leo to detect must meet three criteria to ensure proper detection. All detected object types must meet the detection criteria.

The detection criteria are:

- **Unique**: The object doesn't have any identical or near-identical siblings anywhere in the recorded window (Figure 36 and Figure 37).

Figure 32: Non-Unique Objects

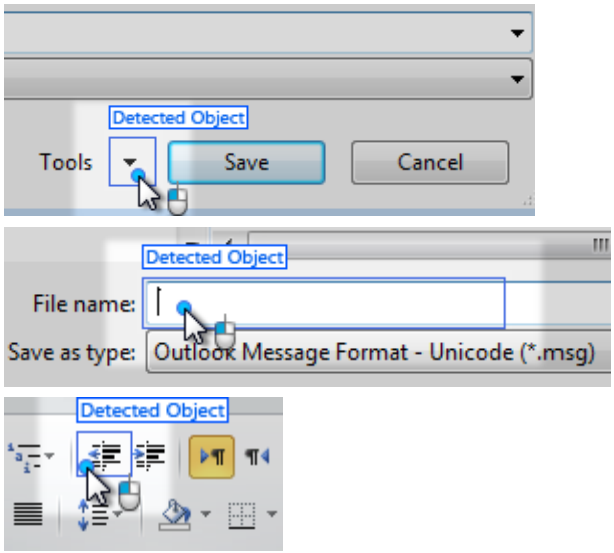
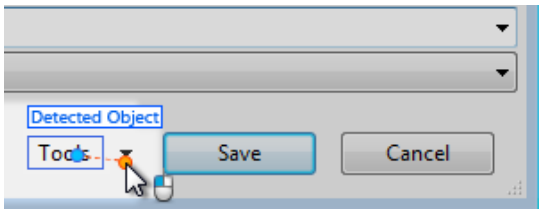


Figure 33: Unique Object



- **Static**: The object appearance never, or rarely changes (Figure 38 and Figure 39).

Figure 34: Non-Static Objects

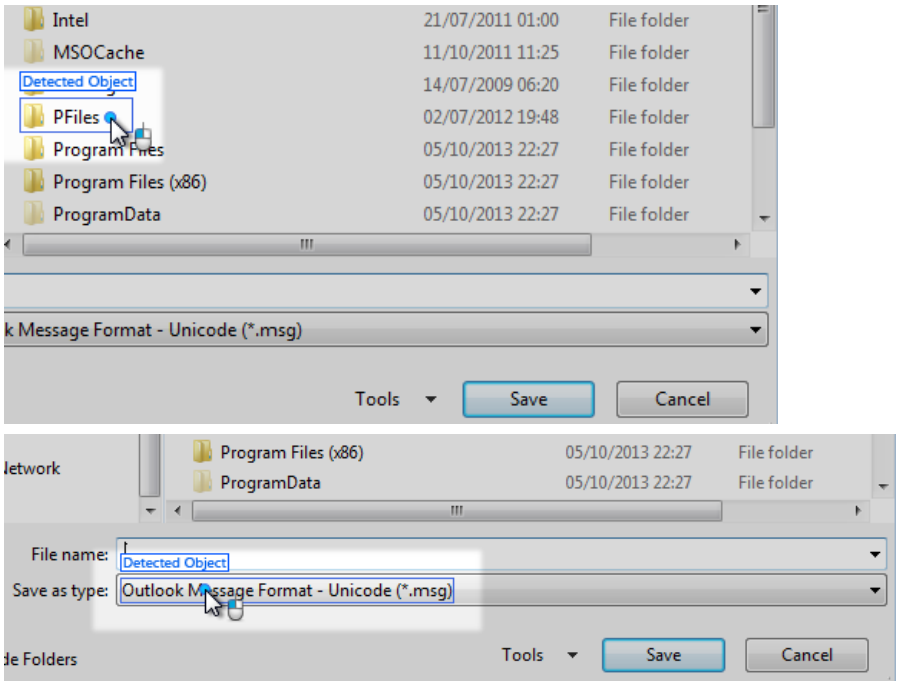


Figure 35: Static Object

- **Clean**: The object isn't chopped, doesn't include any blank areas, and doesn't include unneeded text, blank areas, lines or other shapes (Figure 40 and Figure 41).

Figure 36: Non-Clean Object

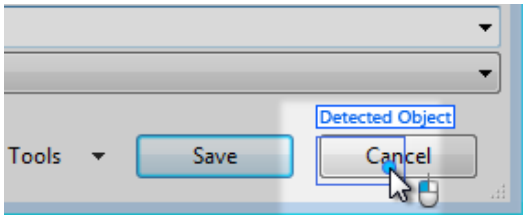
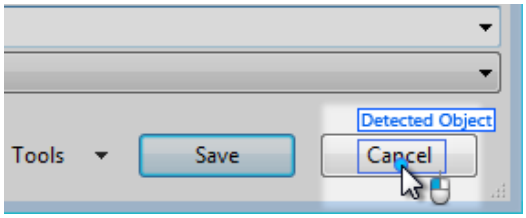


Figure 37: Clean Object



If an object does not meet the object detection criteria, use click offset as described in Section 3.6.2.1.4.9.6.

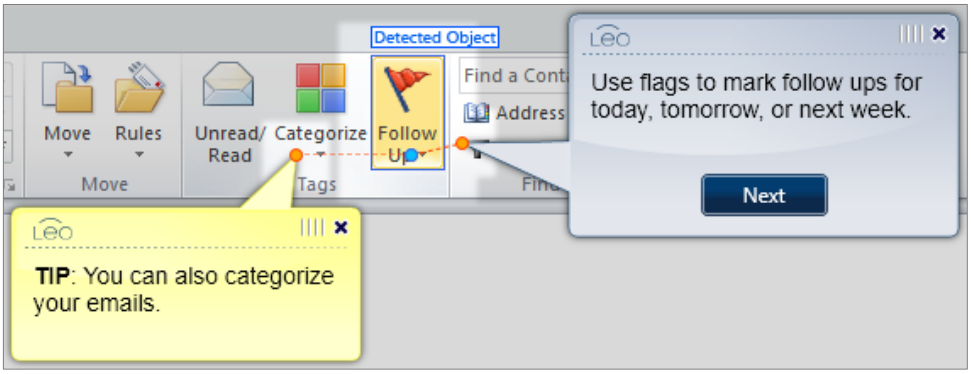
3.1.2.4 Object Detection Procedures

3.1.2.4.1 Reusing Detected Objects

Reusing detected objects ensures optimal Leo performance. Reusing an object can be done by either copying it or snapping one object over another. Leo 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 Leo to detect the object over and over.

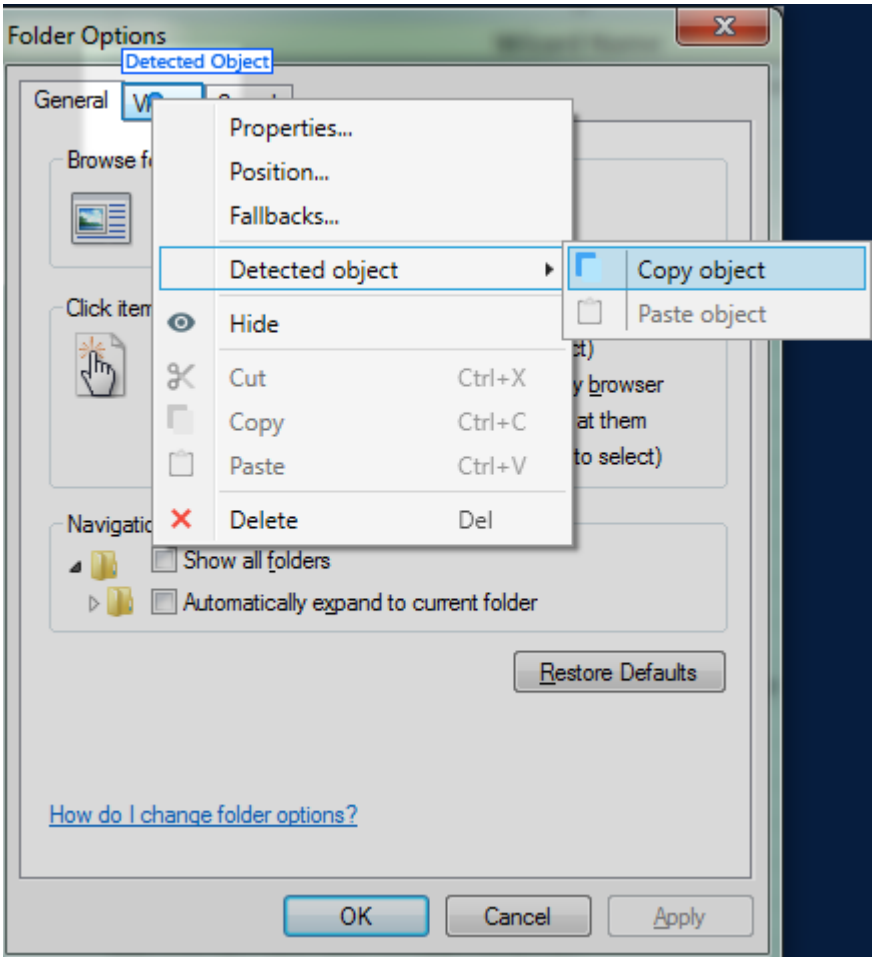
- Reusing within the same step: Reuse in the same step by snapping one object's blue dot over another object's blue dot (Figure 42).

Figure 38: Reusing Detected Object in the Same Step (Snap to Object)



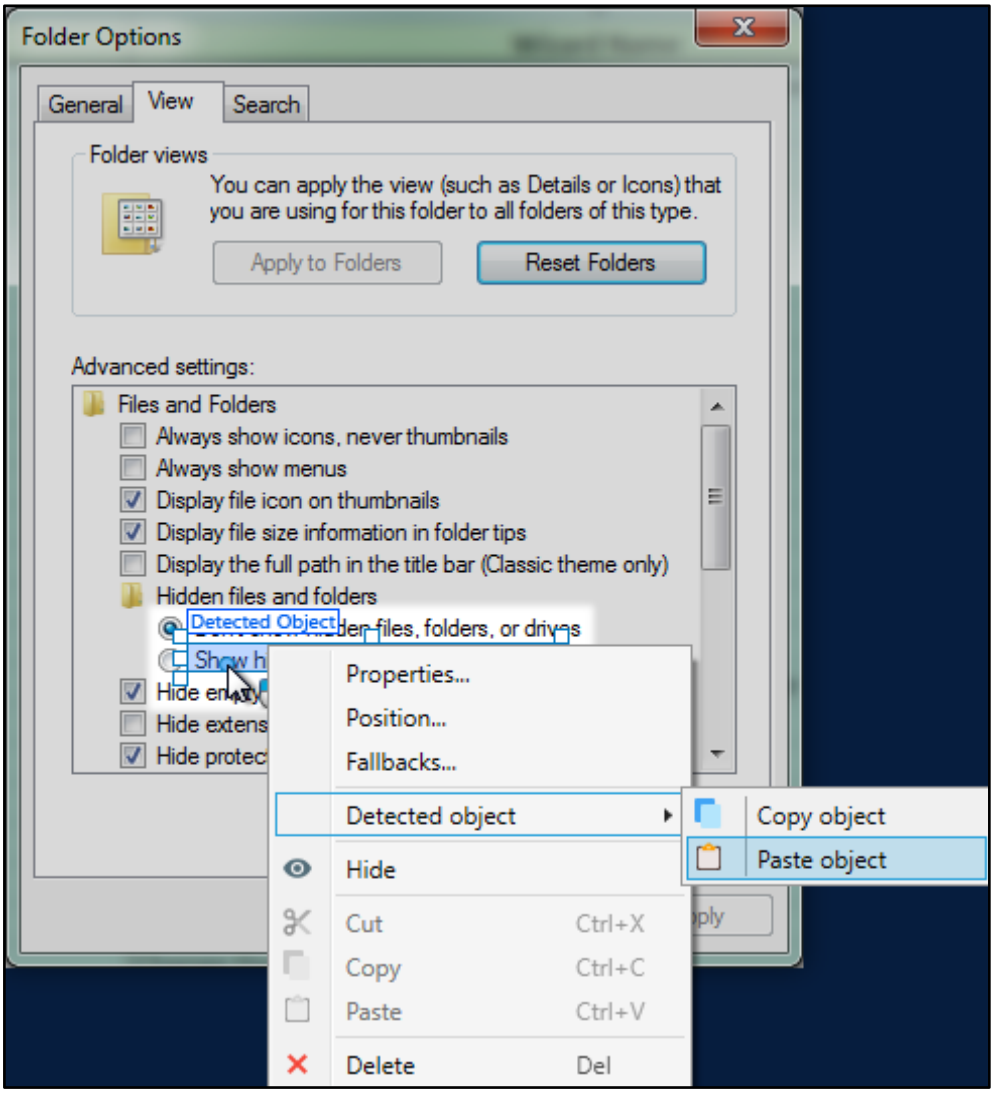
- Reusing across different steps:
  a Copy the object from one step (Figure 43).

Figure 39: Copying a Detected Object



b In another step, paste over the object (Figure 44).

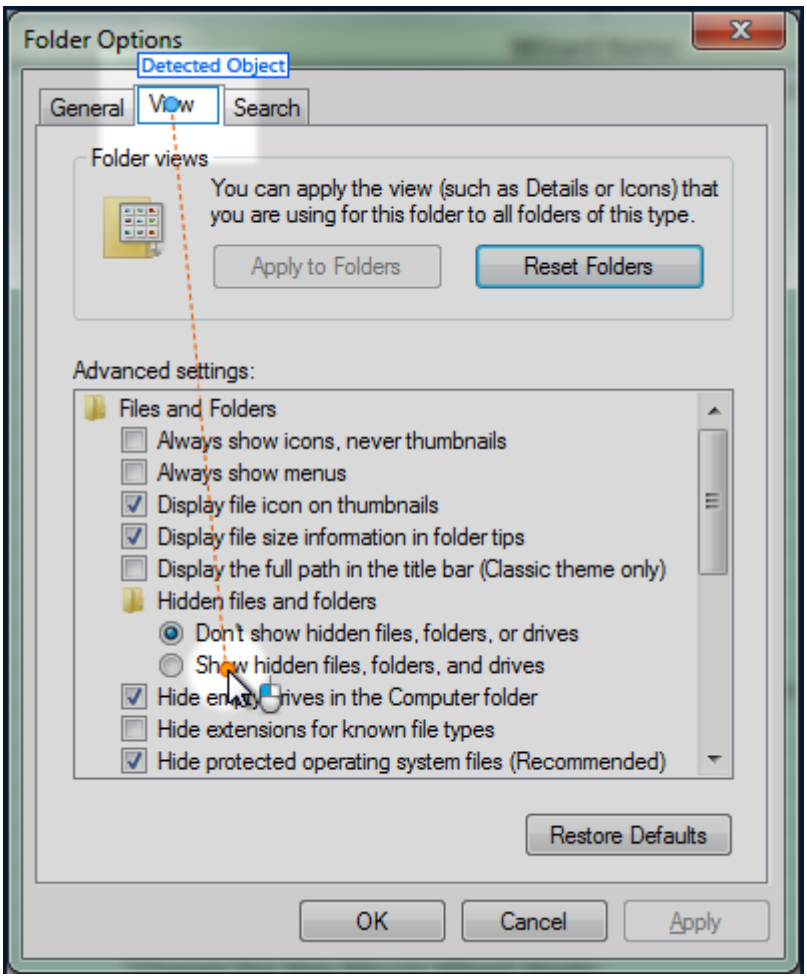
Figure 40: Pasting a Detected Object



⚠ Make sure both steps use the exact same screen, with the exact same window size.

The detected object is now the same in both steps (Figure 45).

Figure 41: Pasted Detected Object

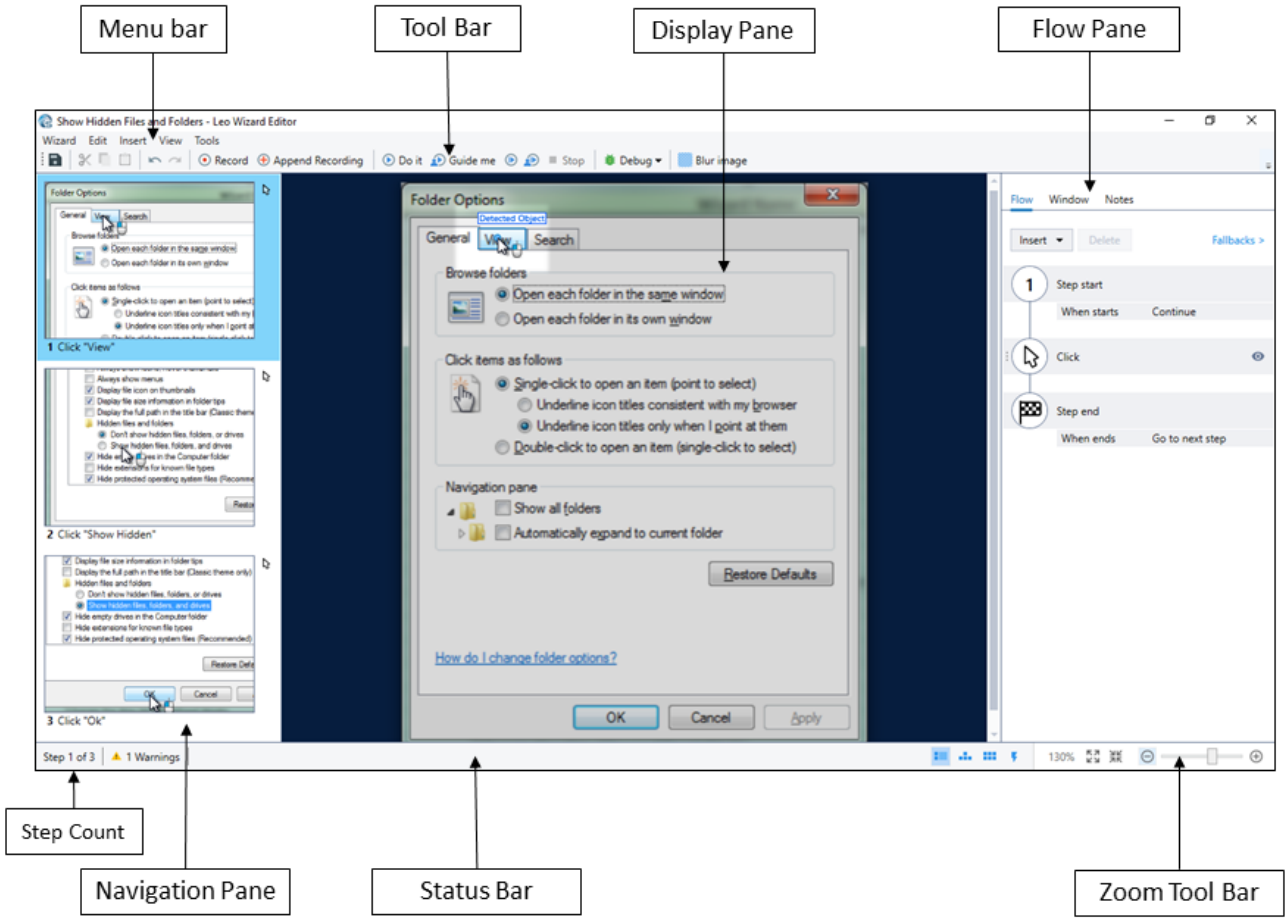


If you are changing the settings of a reused detected object, they are not automatically updated for the reused object in other steps. To update the settings of the detected object in other steps, you must modify them manually.

3.2 Wizard Editor Overview

The Wizard Editor is used for viewing, creating and maintaining Leo Wizards in Leo Studio. It enables you to record and edit new Leo Wizards and to update older wizards. shows the work areas of the Wizard Editor window.

Figure 42 – Wizard Editor Window



The Wizard Editor work areas are:

- **Menu Bar**: A 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**: A collection of functions that affect either the entire wizard or the selected step.
- **Display Pane**: The step's main display, including layers of functionality
- **Flow Pane**: A collection of tabs that provide access to functions on the selected step.
- **Navigation Pane**: The sequence of steps, where each step is represented by a thumbnail the recorded image and action, and displays indicators for the layers of functionality added to it.
- **Status Bar**: Wizard/Sensor errors and warnings.
- **Step Count**: The selected step number and total number of steps.
- **Zoom Toolbar**: The buttons that allow you to zoom in or out of the selected step display.

- **Errors / Warnings indicator**: a label in the status bar indicates whether there are errors or warning in the process.

For information about how to access the Wizard Editor, see Section 2.6.2.3.

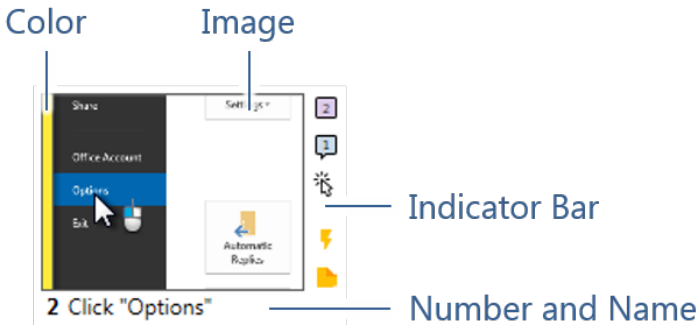
3.3 Wizard Step Overview

Each step is structured in a way that represents the flow of actions within the step. Each step contains settings that allows you to edit its functionality.

Each step thumbnail is made up of the following (Figure 47):

- Step Number
- Step Name
- Image
- Color
- Indicator Bar: indicates whether the step contains any of the following:
  - Core action type: The **Core Action** icon, which changes based on the type of action performed by the step.
  - Blocks: The **Blocks** icon, which appears if the step contains blocks, including the number of bubbles.
  - Bubbles: The **Bubbles** icon, which appears if the step contains bubbles, including the number of bubbles.
  - Advanced Commands: The **Advanced Commands** icon, which appears if the step includes advanced commands.
  - Notes: The **Notes** icon, which appears if the step includes notes.

Figure 43: Step Thumbnail Orientation



3.3.1 Flow Pane

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

- **Flow**: The main pane display. Displays the flow of actions in the step
- **Window**: Contains all the window properties available for the recorded step
- **Notes**: Enables developers to add internal notes and comments to a step

3.3.1.1 Flow Tab

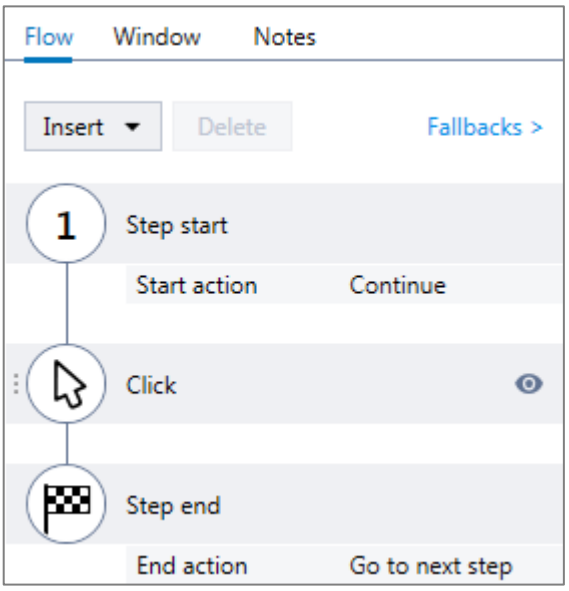
The **Flow** tab is the main pane that appears in the right side of the Wizard Editor window. This pane displays the flow of actions within the selected step, and enables you to perform actions relating to the flow actions. When you record a step, the **Flow** tab automatically displays the three default step actions (Figure 48):

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

- Insert step actions
- Change action types

- Delete actions
- Change action events
- View action fallbacks
- Access the **Properties** pane

Figure 44: Flow Tab with Default Actions

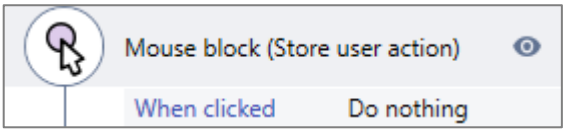


3.3.1.1.1 Step Actions

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

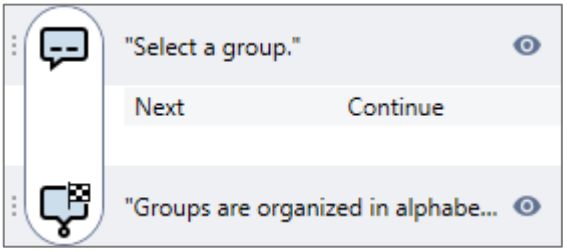
Some actions are built-in by default, while others can be inserted by the content developer according to the process logic. The settings of all actions can be customized to fit the process needs.

Figure 45: Single Step Action



Actions are listed in the **Flow** tab in the order in which they occur. Actions can occur either one at a time or start at the same time as other actions (Figure 50).

Figure 46: Grouped Step Actions



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 settings.

The available step actions are described in Section 3.6.2.1.

By default, all actions are set per step. Some actions can be set to work across more than one step, as described in Section 3.3.1.1.2.

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

- **Visual Indication and Label**: An icon and text label, representing the type of action. For the Step Start action, the icon displays the step number.
- **Properties**: Each action contains additional customizable properties. To access the action properties, either double-click the action or right-click it and select **Properties**.
- **Show/Hide Icon**: Multiple actions can be added to a single step, in which case they might be layered on top of each other in the **Display** pane. To hide an action that prevents you from accessing actions beneath it, click the action's **Eye** icon.
- **Delete**: For removable actions, press **DELETE** or right-click and select **Delete**.
- **Drag Handle**: Movable actions are indicated by a drag handle (Figure 51). Such actions can be dragged and dropped onto a different position in the flow.

  Figure 47: Action with a Drag Handle

- **Assign Actions**: A dropdown list of additional actions, available for some actions.

3.3.1.1.2 Cross-Step Actions

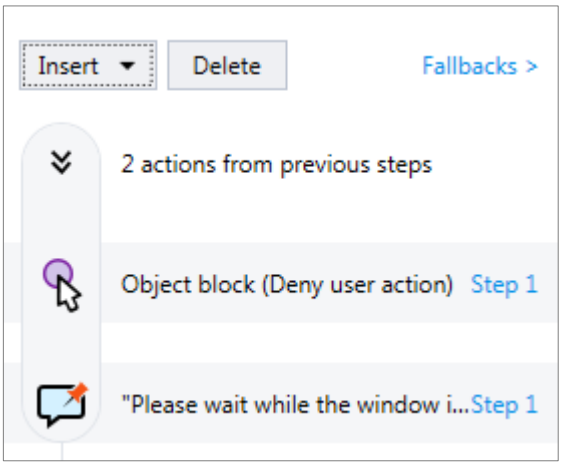
Cross-step actions are actions set to end at the end of a subsequent step in the wizard. In this way, these actions function across more than one step.

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 (Figure 52), and are collapsed by default. They can be edited by clicking the action's step link.

Figure 48: Cross-Step Actions in the Flow Tab



3.3.1.1.3 Fallbacks Display

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

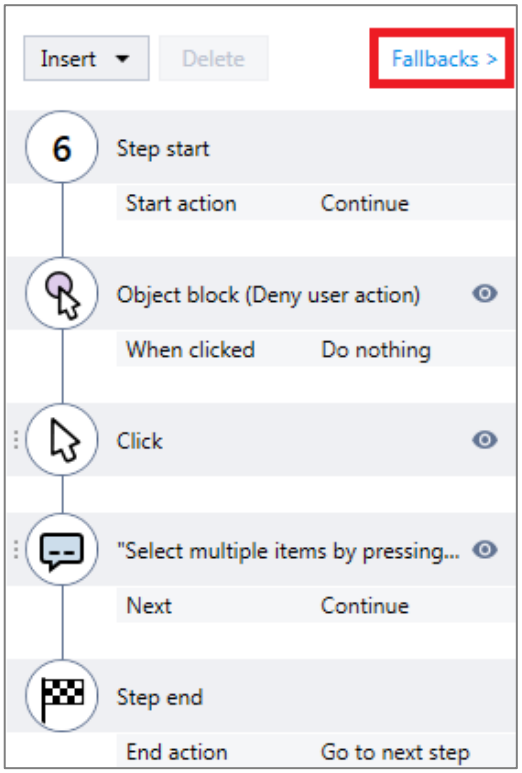
The Step End action does not have fallbacks.

For information about fallbacks, see Section 3.3.2.23.3.2.3.

Access the Fallbacks display by doing the following:

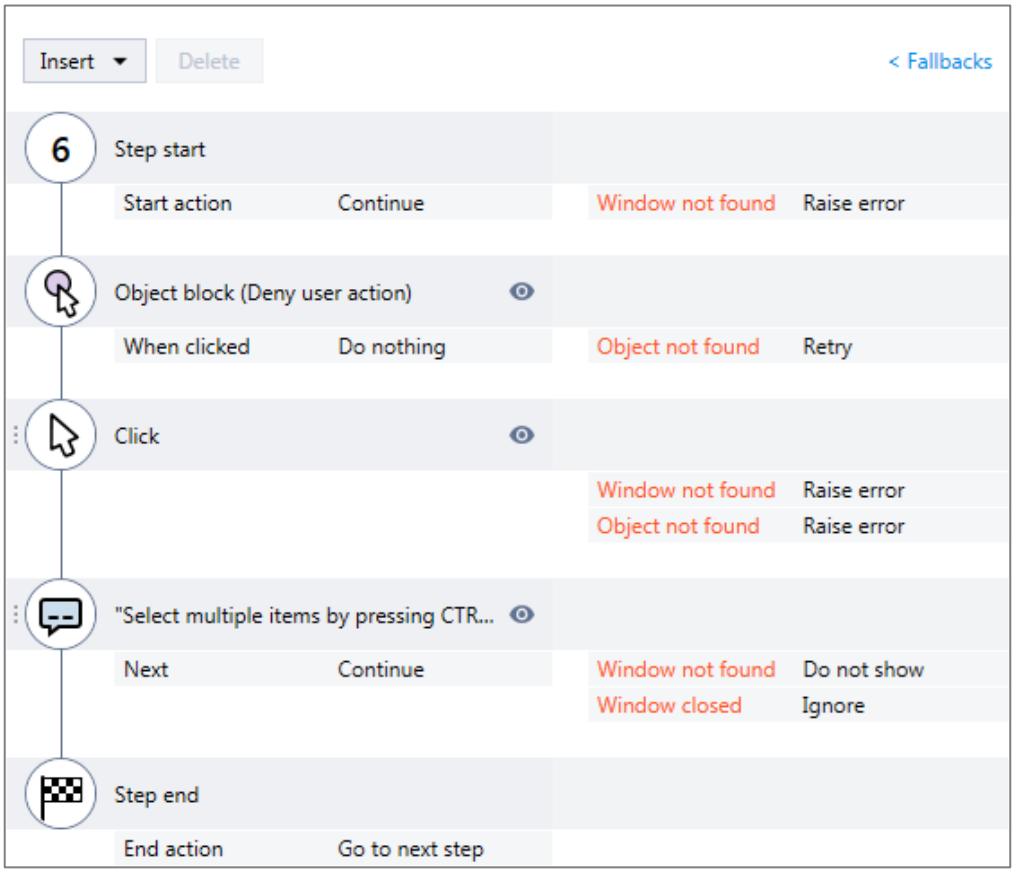
1 From the **Flow** tab, click **Fallbacks** (Figure 53).

Figure 49: Fallbacks Button in the Flow Tab



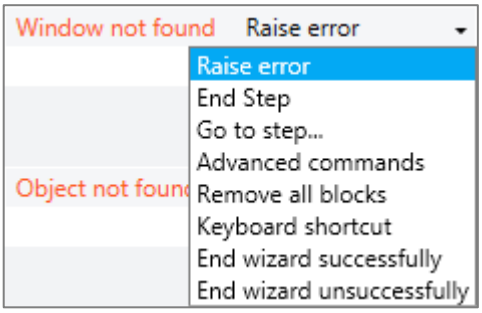
The Fallbacks display appears, listing the fallbacks for each action (Figure 54).

Figure 50: Fallbacks Display in the Flow Pane



You can update a fallback's actions by clicking on the fallback dropdown list (Figure 55).

Figure 51: Fallback Events



For more information about fallbacks events and commands, see Section 3.3.2.3.1.

3.3.1.2 Window Tab

Leo uses window detection to identify the right window in which to work. When you record a wizard, Leo stores properties of each window that you click, and uses these properties to detect the relevant window when the wizard is run.

The **Window** tab contains all the window properties available for the recorded step

For a description of the window properties, see Section 3.1.1

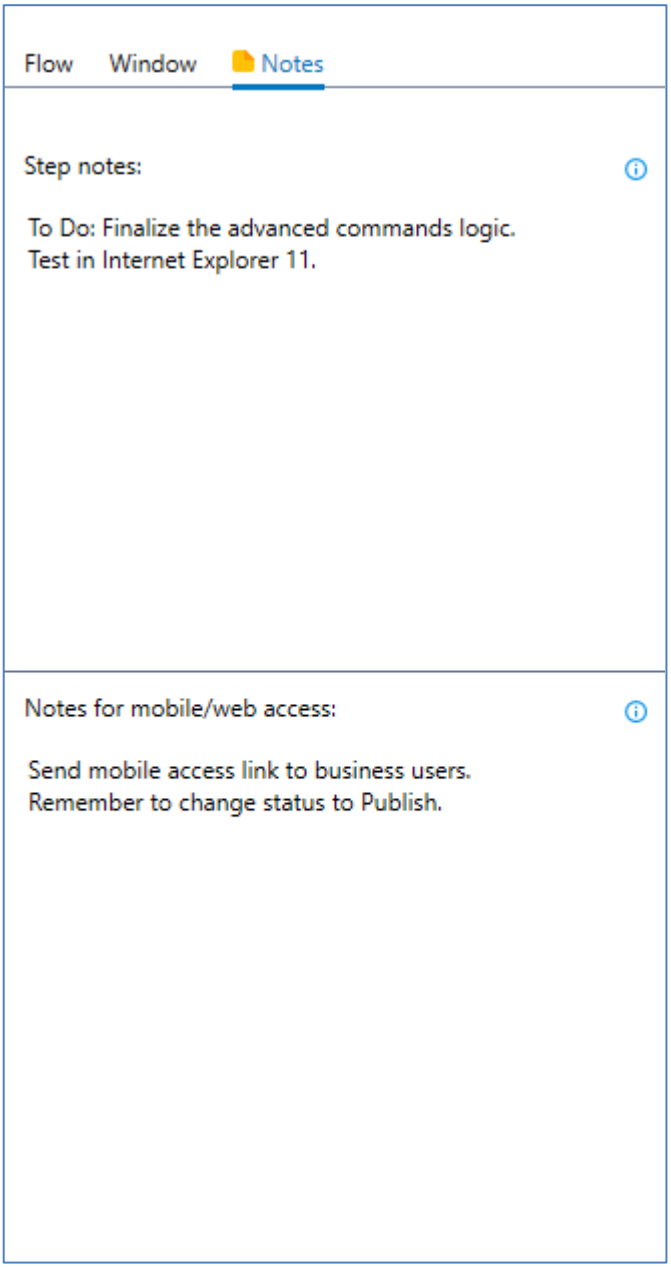
In Leo Studio, window detection is managed from the Flow pane's **Window** tab (Figure 31).

3.3.1.3 Notes Tab

Content 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 Leo Wizard is played in Leo Player. Notes can only be viewed in the Wizard Editor or when the Leo Wizard is exported to a Word or PowerPoint file whose template contains the **Notes** field.

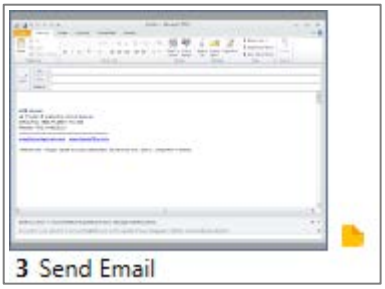
Figure 56 shows the **Notes** tab in the Wizard Editor's **Flow** pane.

Figure 52: Notes Tab



When notes are added to a step, the **Note** icon appears in the Tab's heading, and in the step thumbnail (Figure 57).

Figure 53: Note Indication in Step Thumbnail



3.3.2 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.

For each action or window, the **Properties** pane displays that action's unique properties. The tabs in the **Properties** pane vary depending on the selected action or window.

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

3.3.2.1 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.

3.3.2.2 Position Tab

Some actions require detection of an object on the screen. Object detection is managed by detection settings, which are listed in the action's **Position** tab in the **Properties** pane.

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

3.3.2.3 Fallbacks Tab

All actions, except the Step End action, can use fallbacks to ensure that if an expected action does not take place (e.g. the expected window or object are not found), Leo triggers a backup action that continues the wizard flow uninterrupted.

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

3.3.2.3.1 Fallback Events

Each action in a step depends on either window or object detection. For each action, Leo provides fallbacks that handle failed detection. The goal of a fallback is to transition users successfully through deviations from the process and to minimize the possibility of errors or user confusion.

Actions that do not have fallbacks are:

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

The available fallback events are:

- **Window not Found**:
  - Leo attempts to detect the window in which the action is performed. If Leo fails to detect the window, it triggers this fallback. The default fallback is an error message.
  - For Step Start actions, Leo attempts to detect the window in which the step occurs before the step starts. If Leo fails to detect the window, it performs a fallback. The default fallback is an error message.
  - For Bubble actions, Leo 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 Leo 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, which block access to the recorded window.
- **Window Closed**: This fallback event handles scenarios where the recorded window was initially detected successfully, but was closed in the middle of the step.
- **Object not Found**: Leo attempts to detect the click position before it performs the action. If Leo 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. For information, see Section 3.6.2.1.3.14.
- **Object Hidden**: This fallback event is available for anchored bubbles only. For information, see Section 3.6.2.1.3.14.
- **Wizard not Found**: This fallback event is available for embedded or launched wizards only. For information, see Section 3.6.2.1.5.2.
- **Wizard Ended Unsuccessfully**: This fallback event is available for embedded or launched wizards only. For information, see Section 3.6.2.1.5.2.

For information about the fallback commands used to handle fallback events, see Section 3.3.2.3.2.

3.3.2.3.2 Fallback Commands

Leo provides a number of commands to handle each fallback event.

The fallback commands Leo can perform for each event are:

- Raise Error Message: A Leo error message appears on the **Wizard** bar.

Not applicable to Leo Sensors or embedded wizard steps.

- Fail Sensor: Leo ends the sensor and the sensor is 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 Leo Wizards.

- End Step: Leo ends the step and continues with its **When step ends** event.
- Go to Step: Leo goes to a specific step in the flow.

Not applicable to embedded wizard steps.

- **Advanced Commands**: Leo triggers a predefined set of advanced commands. When the advanced commands are completed, Leo continues the wizard flow in the order determined by the content author.
  For more information about advanced commands, see Section 5.1.
- **Remove all Blocks**: Dismisses all blocks and cancels their functionality for that run of the wizard or sensor.
  For more information about blocks, see Section 0.
- Use Keyboard Shortcut: Leo types a keyboard combination entered by the content author (e.g. **ALT+P**).
- End Current Wizard Successfully: Leo exits the Leo 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 content author.
For more information about embedded wizards, see Sections 3.6.1.4-3.6.1.6.

- End Current Wizard Unsuccessfully: Leo exits the Leo 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 content author.
  For more information about embedded wizards, see Sections 3.6.1.4-3.6.1.6.

In addition to backing up failed scenarios, fallbacks can be used to create more complex Leo 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.

The following sections describe the available fallbacks and usage examples:

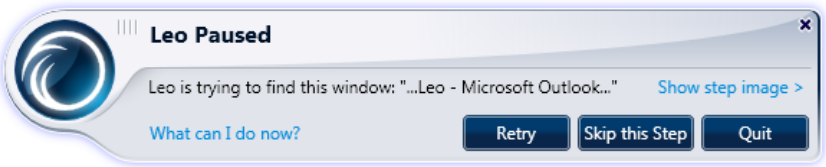
- Raise Error Message
- End Step
- Go to Step
- Use Keyboard Shortcut
- End Current Wizard Successfully
- End Current Wizard Unsuccessfully

3.3.2.3.2.1 Raise Error Message

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

Example: You recorded a Leo Wizard for creating a new mail message and typing its content. If while playing the Leo Wizard the user changes the subject of the message before Leo reaches this step in the flow, the message's window caption is changed. In this scenario, Leo looks for a window called **Leo – Microsoft Outlook** 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 Leo Paused message appears on the user's screen (Figure 58).

Figure 54: Leo Paused Message



This error can be avoided by setting the window caption to **Contains** instead of **Equals**. For more information, see Section 3.1.1.

3.3.2.3.2.2 End Step

When the **End Step** fallback is applied to a step, Leo ends the step and continues with the command specified in the step's **When ends** event. It is the default action when Leo successfully performs a step. **End Step** is useful as a fallback when the user closes the window that Leo is about to close after a bubble. In such cases, this fallback enables Leo to end the step instead of trying to detect the closed window.

Example: You recorded a Leo Wizard for specifying a file name and location. In this scenario, users might click **OK** to confirm the action instead of waiting for Leo to do

so. This closes the window in which Leo is supposed to **OK** for the user. If the **End Step** fallback was applied to this step, it is activated and Leo ends the step instead of trying to detect the **OK** button.

3.3.2.3.2.3 Go to Step

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

Example: You recorded a Leo Wizard that moves an email message to another folder. If there is an open message on the screen, the Leo Wizard continues by moving this message. If there is no open message and the **Go to Step** fallback was applied, it is activated and Leo goes to a step in which the user is prompted to select a message to move.

3.3.2.3.2.4 Use Keyboard Shortcut

When the **Use Keyboard Shortcut** fallback is applied to a step, Leo types a keyboard shortcut instead of clicking to perform the core action. This is useful when the action can be done with both a shortcut key and a mouse click. In this case, if Leo 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 changes its appearance and Leo cannot detect it. If the **Use Keyboard Shortcut** fallback was applied to this step, it is activated and Leo types the specified shortcut keys (e.g. **ALT+N+E**) to click the **Equation** button.

Figure 55: Large Insert Equation Button

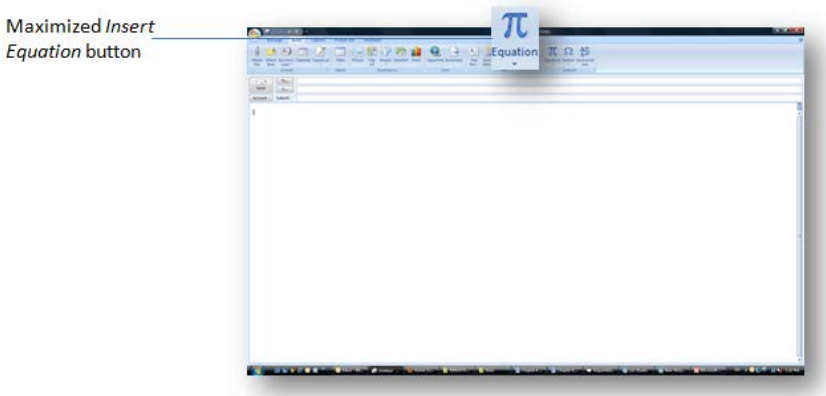
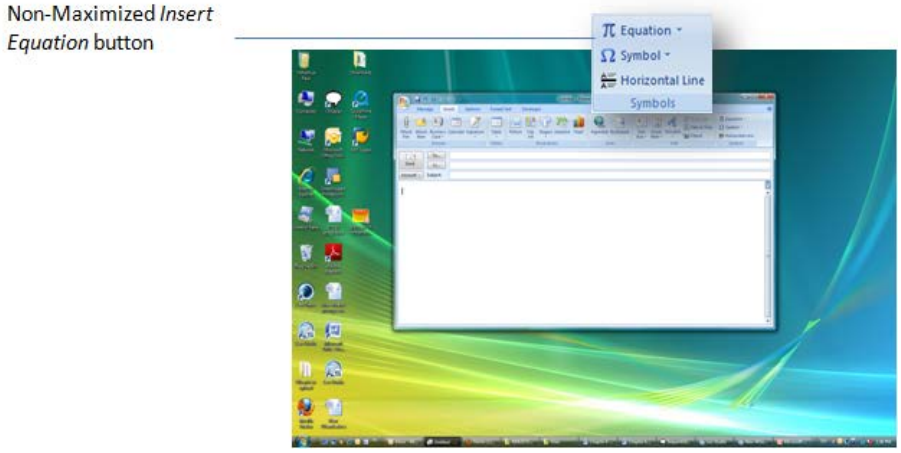


Figure 56: Small Insert Equation Button



3.3.2.3.2.5 End Current Wizard Successfully

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

Example: You recorded a Leo Wizard for saving messages, in which the final step is the Microsoft 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 **Use Keyboard Shortcut** fallback was applied to this step, it is activated and Leo ends the Leo Wizard at this point in the flow because there are no more actions to take.

For more information about embedded wizards, see Sections 3.6.1.4-3.6.1.6.

3.3.2.3.2.6 End Current Wizard Unsuccessfully

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

For more information about embedded wizards, see Sections 3.6.1.4-3.6.1.6.

3.3.2.3.3 Fallbacks Tab Procedures

3.3.2.3.3.1 Applying Fallbacks to a Step Action

For a description of the available fallbacks, see Section 3.3.2.3.2.

Apply a fallback to a step by doing the following:

1. Select the step whose visual detection you want to edit.
2. Do one of the following:
   a. From the **Flow** pane, access the Fallback display as described in Section 3.3.1.1.2.
   b. From the Flow pane, access the action's **Properties** pane as described in Section 3.3.2.4.
   c. From the **Properties** pane **Fallbacks** tab, locate the relevant fallback event.
3. From the fallback event's dropdown list, select a fallback command.

The fallback is applied to the step action.

3.3.2.4 Properties Pane Procedures

3.3.2.4.1 Accessing an Action's Properties Pane

To access the **Properties** pane of a step action, do the following:

1 From the **Flow** pane's **Flow** tab, do one of the following:

- Double-click the action.
- Right-click the action and select **Properties**.
- Select the action and from the **Menu** bar, select **Properties Pane**.

3.3.2.4.2 Accessing a Window's Properties Pane

To access the **Properties** pane of the step window, do the following:

1 From the **Flow** pane's **Window** tab, do one of the following:

- Double-click the window event.
- Right-click the window event and select **Properties**.

3.4 Wizard Views

In the Wizard Editor window, you can choose how to view the flow of a wizard, and control the panning and zooming of the image in the **Display** pane, as described in the following sections:

- Normal View
- Diagram View
- Advanced Commands View
- Advanced Commands View

Advanced Commands view displays the wizard flow with advanced commands editing capabilities. It displays the flow of each step, and enables you change each event's command to an "advanced command" and edit its contents.

See Advanced Commands

- Spotlight Display

3.4.1 Normal View

Normal view is the default view in which wizards are opened in the Wizard Editor. In this view, all wizard steps appear in a linear list of thumbnails in the **Navigation** pane. The selected step's image appears in the **Display** pane (see).

3.4.2 Diagram View

Diagram view displays the wizard flow in a diagram 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.

Diagram view displays the following for each step:

- **Thumbnail**: Includes click position, indicator icons, step number, and step name
- **Connectors**: Lead 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. Table 3 lists the shapes and connectors used in Diagram view.

The shapes and their functions are also shown in the diagram itself.

Table 3: Diagram Shapes and Connectors

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

3.4.2.1 Diagram View Functions

Diagram view enables you to perform the following functions:

- Double-clicking a step to open it in Normal view
- Appending Steps
- Running a Leo Wizard from a Specific Step
- Debugging a Wizard
- Saving a Wizard
- Opening a Locally Saved Wizard in the Wizard Editor

3.4.2.2 Diagram Viewing Options

Diagram view enables you to change the layout of a wizard and offers various viewing options. These options are:

- Panning
- Zooming
- Diagram Layout: Determines the layout of steps. The default view is Hierarchical.
- Arrow Layout: Determines the layout of connectors. The default view is Direct.

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

3.4.3 Advanced Commands View

Advanced Commands view displays the wizard flow with advanced commands editing capabilities. It displays the flow of each step, and enables you change each event's command to an "advanced command" and edit its contents.

See Advanced Commands

3.4.4 Spotlight Display

In Normal view, the **Spotlight** feature highlights the step's detected objects. The spotlight appears around each detected object in the **Display** pane.

3.5 Recording Wizards

Wizards are recorded by performing the process on your own application, while the Leo Recorder is running in the background, capturing every click and keystroke on the licensed application.

The actions captured by Leo during the recording are mouse clicks and keystrokes only. Other actions, including hovering or dragging the mouse, are not captured and need to be added by editing the core action.

In recording mode, as long as you do not click your mouse or use the keyboard Leo does not create new steps. Leo only captures new steps when you click your mouse or press keyboard keys on the target application.

The wizard recording procedures are described in the following sections:

- Before You Begin: Environment Setup
- Recording a New Wizard
- Viewing and Navigating a Wizard

3.5.1 Before You Begin: Environment Setup

Proper environment setup ensures that minimum optimization of recorded Leo content is required.

Leo content is designed to be run live on the end-user's real environment. As such, Leo is designed to handle a variety of environment settings that affect objects' position and appearance. The environment variations Leo is designed to handle include different operating systems, browsers, screen resolutions, and GUI settings. Leo possesses various tools such as docking, search area, and image and text matching tools to ensure detection when the live environment differs from the recorded one.

While Leo's detection and optimization tools cover many, if not most, of the environment variations, some objects may need to be tweaked and optimized for correct detection. To minimize the need for optimization, it is important to record Leo content on an environment that resembles as closely as possible the **most common** environment expected to be used by Leo's end users.

Environment setup involves finding out what are the environment settings that the **majority** of users have. The goal is to discover the most commonly used environment to record on.

The user environment components that must be determined before recording are:

- **Operating System**: The most commonly used operating system out of the following supported operating systems:
  - Windows XP
  - Windows Vista
  - Windows 7
  - Windows 8
  - Windows 10
  - Citrix environment:
    - Windows 2003
    - Windows 2008
    - Windows 2008R2
    - Windows 2012
- **Screen Resolution**: The minimum and maximum screen resolutions to be used.
- **Browser Type and Version**: If applicable. For Web applications, the most commonly used browser out of the following:
  - Google Chrome 24 and up
  - Mozilla Firefox 29 and up
  - Internet Explorer 7 and up

- Microsoft Edge

- **Font Size**: If applicable. The Windows DPI setting. This is only applicable to target applications whose text is affected by varying DPIs.
- **Application Version**: If applicable. Various versions of the target application that are available to end users.
- **User Permissions**: Different user types and settings that affect application behavior.

Once these settings are determined, the recording needs to be done on an environment that represents the majority of users. After the content is developed, testing and tweaking will be done for the less common end-user environments, thus saving valuable testing time.

3.5.2 Recording a New Wizard

What You Need

- A computer using the most common end-user environment.
  For more information about the required end-user environment, see Section 3.5.1.
- The supported application on which you want to record, launched on your computer and set to the first step of the process.
  For more information about the supported applications available to you, see Section 2.3.1.5.
- A new Leo Wizard added to the Leo Catalog, as described in Section 2.6.2.1.

Record a new Leo Wizard by doing the following:

1 Open the Wizard Editor, as described in Section 2.6.2.3.

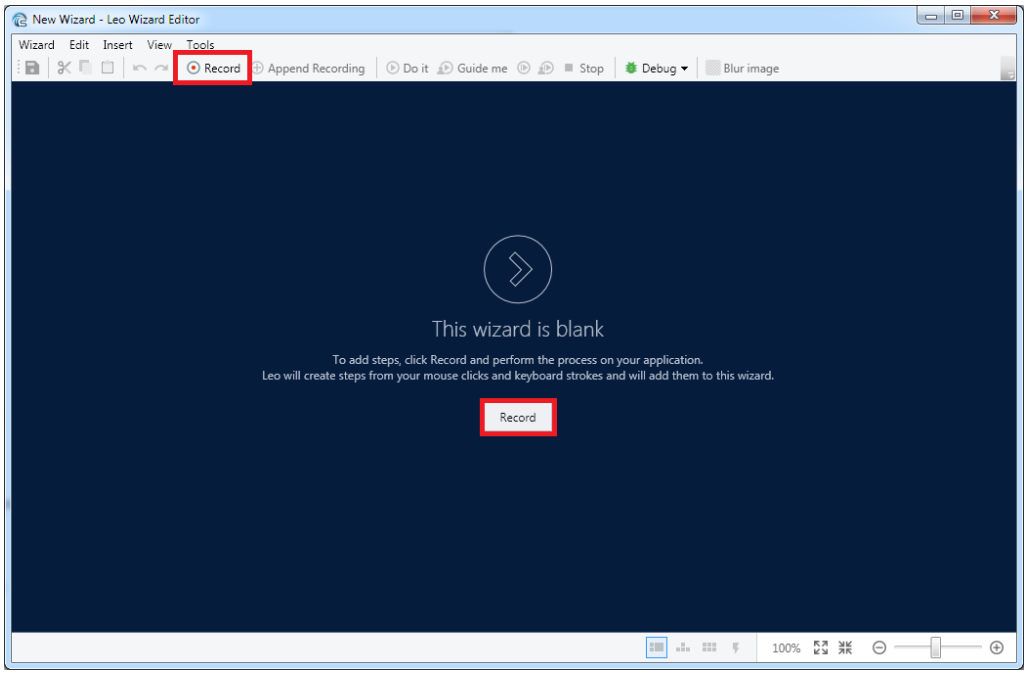
2 Do one of the following:

- On the toolbar, click **Record** (Figure 61).
- Click the **Record** button at the center of the **Display** pane (Figure 61).

This option is only available when you record a new Leo Wizard.

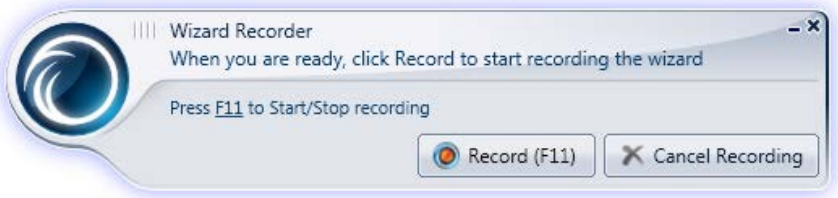
- On the Wizard Editor menu bar, select **Wizard > Record**.

Figure 57: Wizard Recording Buttons



The **Wizard Editor** window is minimized, and the **Wizard Recorder** bar appears (Figure 62).

Figure 58: Wizard Recorder Bar



3 Go to the application window in which you want to start recording.

4 Click **Record**.

In the **Wizard Recorder** bar, a 3-second countdown begins.

When the countdown ends, the **Recorder** icon appears on your taskbar (Figure 63), indicating that the recording is in progress.

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

Figure 59: Recorder Icon on the Taskbar

5 Perform the process in the application, while the Leo Recorder is running in the background.

6 Do one of the following:

- To pause the recording, do one of the following:
  - From the taskbar, click the **Recorder** icon.
  - Press **F11**.

To change the **Start/Stop Recording** hotkey, click the hotkey link in the **Wizard Recorder** bar and select a different hotkey from the dropdown list.

The recording is stopped and the **Wizard Recorder** bar appears (Figure 64).

Figure 60: Recording Stopped

When you are ready to continue recording, click **Resume**.

- To cancel the recording:

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

  ii The **Wizard Recorder** bar appears.

  iii Click **Cancel Recording** to cancel the recording and return to the Wizard Editor.

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

- To end the recording:

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

  ii The **Wizard Recorder** bar appears.

  iii Click **Finish** to end the recording and return to the Wizard Editor.

  The recording is ended and the **Wizard Editor** window appears, containing the steps you recorded.

Any steps containing scrolling or drag-and-drop actions are automatically disabled in the generated wizard. Automatic scrolling is performed automatically by Leo, as described in section 3.6.2.2.2.8. Drag-and-drop actions are not supported as **recorded** mouse actions. They can be executed by using the Drag & Drop advanced command, as described in section 5.1.3.4.

3.5.3 Viewing and Navigating a Wizard

When you select a step in the Wizard Editor, the full step image appears in the **Display** pane with the core action visible in the pane's default Normal view.

When the recorded window is larger than the **Display** pane view, you can move the view around the pane by either using the scrollbars or panning.

The Leo Wizard Editor offers several options for viewing the wizard and each step. These options are listed in the following sections.

3.5.3.1 Viewing a Wizard in Diagram Mode

View a wizard in Diagram mode by doing the following:

1 On the **Wizard Editor** bottom bar, click the **Diagram View** icon.
The wizard flow appears in diagram mode.

To return to Normal view mode, click the **Normal View** icon.

For more information about the diagram view options, see Section 3.4.2.

3.5.3.2 Displaying Disabled Steps

By default, disabled steps are hidden in the **Navigation** pane, displaying the step name and number only. This provides you with a cleaner view of the flow's active steps.

View disabled steps by doing the following:

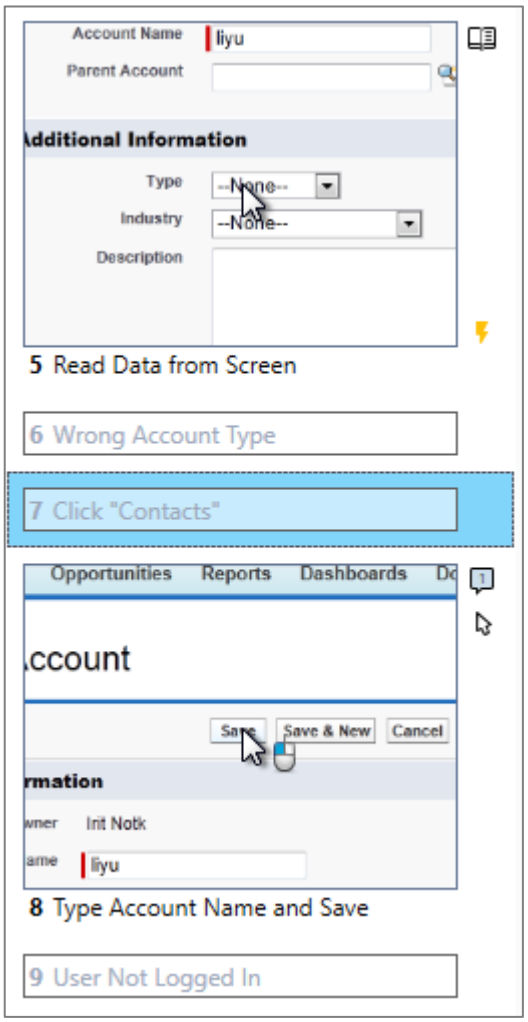
1 On the **Wizard Editor** menu bar, click **View**.

2 Click the **Hide disabled steps** option to deselect it.
In the **Navigation** pane, the thumbnails of all disabled steps are expanded to display their thumbnails.

2 In

Figure 65, Wizard steps 2-3 are disabled and displayed.

Figure 61: Disabled Steps Displayed



Clicking a hidden step in the **Navigation** pane displays its contents in the Editor's **Display** pane.

3.5.3.3 Disabling Spotlight Display

By default, the **Display** pane highlights the selected step's detected objects.

Disable the spotlight feature by doing the following:

1 On the Wizard Editor menu bar, click **View** > **Spotlight**.
Detected objects are no longer highlighted in the **Display** or **Navigation** pane.

3.5.3.4 Going to a Specific Step

While you can click steps to select or navigate them, Leo Studio also provides a menu for navigating wizard steps.

Navigation options are only available in **Normal** view.

Select a navigation option by doing the following:

1 On the **Wizard Editor** menu bar, select **Edit > Go to**.
2 Select the navigation option you want.
The relevant step is selected and appears in the **Display** pane.

3.5.3.5 Panning

To use panning, click and drag the **Display** pane to pan it.

3.5.3.6 Zooming

Zooming enables you to magnify or shrink the step view in the **Display** pane. The zoom options are controlled from the Zoom Toolbar (Figure 66).

Figure 62: Zoom Toolbar

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

3.6 Editing Wizards

When you finish recording, the wizard appears in the Wizard Editor and is ready for editing. Editing a wizard consists of organizing your recording and adding layers of functionality that turn your basic recording into a performance support, learning, or automation tool.

3.6.1 Editing the Wizard Flow

The Leo Wizard you recorded appears in the Wizard Editor window in the sequence in which you recorded.

One of the goals of editing a wizard is to review and organize the sequence of steps. When you edit the wizard, you might need to reorder parts of the wizard flow, depending on the process logic and your editing needs. You can do this by adding, moving, duplicating, categorizing, or deleting steps.

The following sections describe how to edit the flow of steps in a Leo Wizard.

3.6.1.1 Moving a Step

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

Move a step by doing the following:

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

3.6.1.2 Copying or Duplicating a Step

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

Copy a step by doing the following:

1 In the **Navigation** pane, select the step that you want copy.
2 Do one of the following:
   - On the Wizard Editor 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 Wizard Editor menu bar, select **Edit > Paste**.
   - On the toolbar, click the **Paste** icon.
   - Press **CTRL+V**.

To duplicate a step within the same wizard, select the step and press **CTRL+D**.

3.6.1.3 Appending Steps

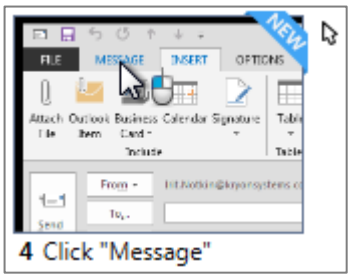
Leo enables you to add steps to an existing Leo Wizard. Use the same best practices for appending as you would for recording. Appended steps are edited as regular steps.

Append steps to a Leo Wizard by doing the following:

1 In the **Navigation** pane, select the step after which you want to append your step
2 Do one of the following:
   - On the Wizard Editor menu bar, select **Wizard > Append Recording**.
   - On the toolbar, click **Append Recording**.
3 Continue with Recording a New Wizard, 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 (Figure 67). The **New** label disappears when you select a different step.

Figure 63: Appended Step with New Label



3.6.1.4 Deleting a Step

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

⚠ Step deletion is permanent and cannot be reversed. To recreate a deleted step, you must append the step by re-recording it. For information about appending steps, see Section 3.6.1.3.

Delete a step from a Leo 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**.

  A confirmation message appears.

2 Click **OK** to permanently delete the step.

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

3.6.1.5 Embedding a Wizard

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

Embedded wizards are not available for sensors. For information about how to launch a wizard from a sensor, see Section 4.2.4.

Embed a wizard into the flow of another wizard by doing the following:

1 Open the Leo Wizard into which you want to embed another wizard, as described in Section 2.6.2.3.
2 In the wizard's **Navigation** pane, select the step after which you want to embed another wizard.
3 On the Wizard Editor menu bar, select **Insert > Embedded Wizard**.
The **Catalog** dialog box appears (Figure 27).
4 From the **Catalog** dialog box, select the Leo Wizard that you want to embed.

To search for a specific Leo Wizard, click **Find** or press **CTRL+F** and then type your search query.

The Leo Wizard you selected is embedded as a step into the containing wizard's flow. The embedded wizard's thumbnail appears in the Wizard Editor's **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 (Figure 122).

To preview the embedded wizard's steps, click the **Sample screenshots** dropdown list.

Wizards cannot be self-embedded.

3.6.1.6 Editing an Embedded Wizard

Edit an embedded wizard by doing one of the following:

- From the containing wizard's **Navigation** pane:
  a Select the embedded wizard step.
  b In the embedded wizard step, click **Edit Embedded Wizard**
  The embedded wizard is opened for editing in a new Wizard Editor window.
- From the Wizard Catalog, select the wizard to edit.

3.6.1.7 Categorizing Wizard Steps

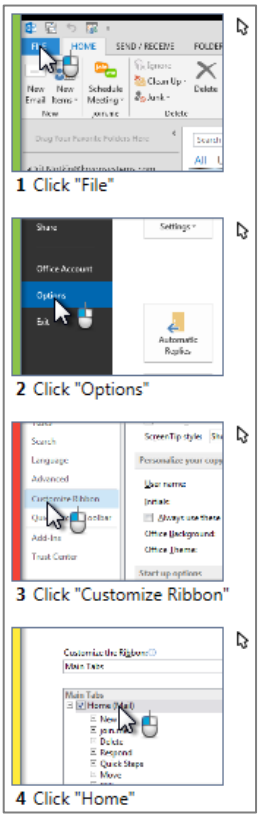
Steps in a Leo Wizard can be grouped by color categories for flow organization purposes. Step categories help you keep track of the wizard structure. Categories

enable you to follow sub-processes that branch out of the main wizard structure, created by decision points and fallbacks.

Categorize steps in a wizard by doing the following:

1 In the **Navigation** pane, right-click the step thumbnail to categorize and select **Color**.
2 From the **Color** list, select a color that categorizes the step.
3 The thumbnail's indicator bar is colored with the category color you selected (Figure 68).

Figure 64: Steps Categorized by Color



3.6.1.8 Undoing and Redoing Changes

Most actions you perform while editing a Leo Wizard can be undone and redone. However, actions can only be undone before you close the Leo Wizard. Once you close the wizard, the action cannot be undone.

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

- On the Wizard Editor menu bar, select either **Edit > Undo** or **Edit > Redo**.
- On the toolbar, click either the **Undo** or **Redo** icon.
- Press **CTRL+Z** to undo or **CTRL+Y** to redo an action.

3.6.2 Editing a Wizard Step

Editing wizard steps is the process of adding layers of Leo functionality to the basic recording of individual steps.

After you record, you must edit the wizard functionality to fit the purpose of the process. The editing process includes applying some or all of the following features:

- **User Instructions**: Bubbles
- **Branching Scenarios**: Decision points via bubbles, fallbacks, or blocks
- **Pitfall Coverage**: Fallbacks

- **Speed and Accuracy Verification**: Optimization tools
- **Behind-the-Scenes or Automation Logic**: Read-from-screen and advanced commands

The following sections describe the features available for applying functionality when editing individual steps in a Leo Wizard, and how to use these features correctly to ensure proper wizard functionality.

3.6.2.1 Step Actions

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

Step actions can be viewed and accessed for editing from the **Flow** pane, as described in Section 3.3.1.1.

The available step actions are:

- Step Start: The step's opening action. This action is inserted by default, and cannot be moved or removed.
- Blocks: Always occur immediately after the Step Start action. This action is optional. It can be inserted and removed, but cannot be moved.
  Multiple blocks can be added to a step.
- Bubbles: Occur either before or after the core action. This action is optional. It can be inserted, removed, and moved.
  Multiple bubbles can be added to a step.
- Core Action: The step's core action. This action is inserted by default, and can be moved and removed.
- Step End: The step's closing action. This action is inserted by default, and cannot be moved or removed.

3.6.2.1.1 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.

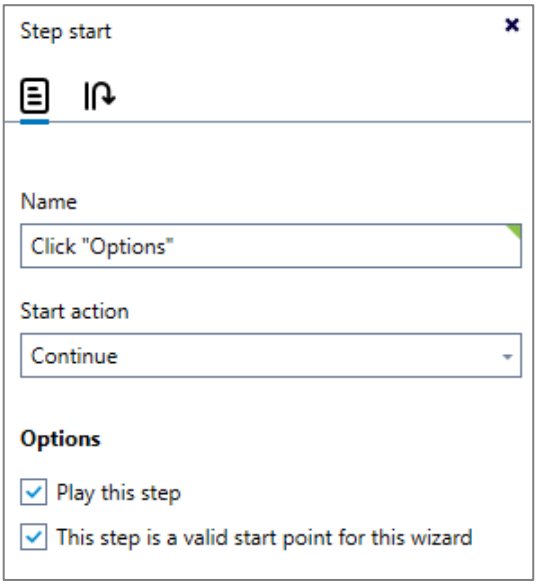
Step Start enables you to view the opening event, and to set an opening command for the selected step by selecting an option from the **When starts** dropdown list.

3.6.2.1.1.1 Step Start Properties

The Step Start **Properties** pane contains the following properties (Figure 69):

- Step Name: Contains a label that identifies the step.
- Step Start Action: Determines which command to trigger when the selected step begins.
- Step Start Options: Additional options related to the Step Start action.

Figure 65: Step Start Properties Pane

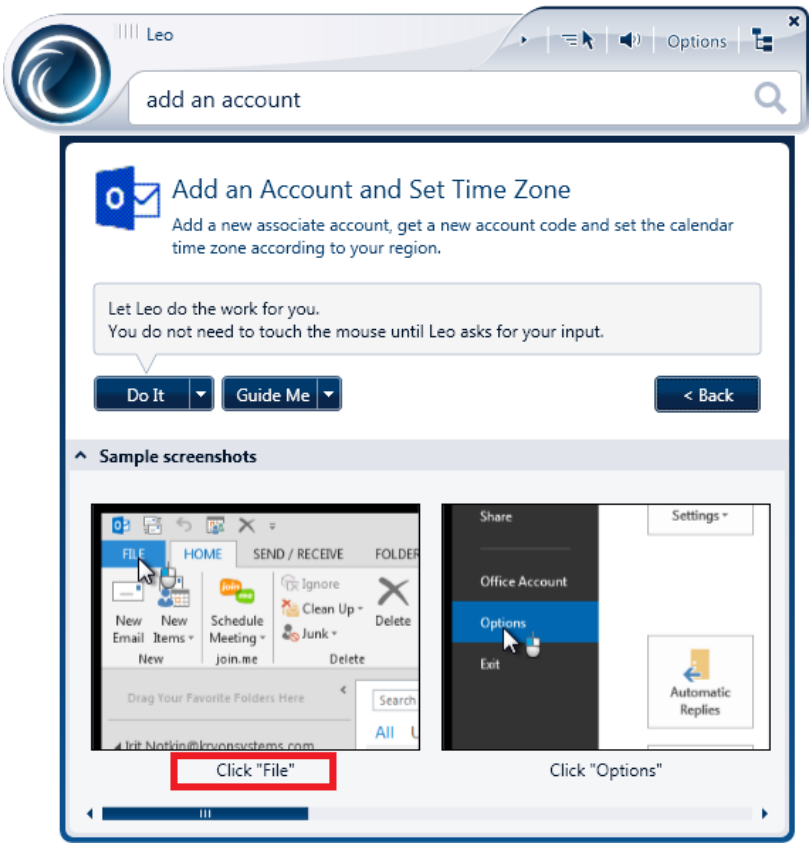


3.6.2.1.1.1.1 Step Name

Step names are labels that identify a wizard step according to its purpose. Step names are either blank or generated automatically by Leo, and can be customized by the content author. Each step name is visible to end users in the following locations:

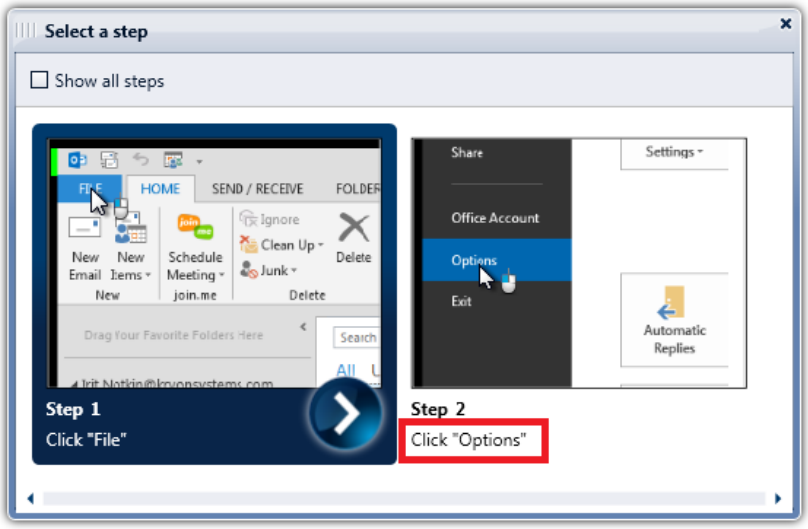
- **Published Wizards in Leo Player**:
  - Sample screenshots in the search bar or catalog (Figure 70):

    Figure 66: Step Name in Leo Player



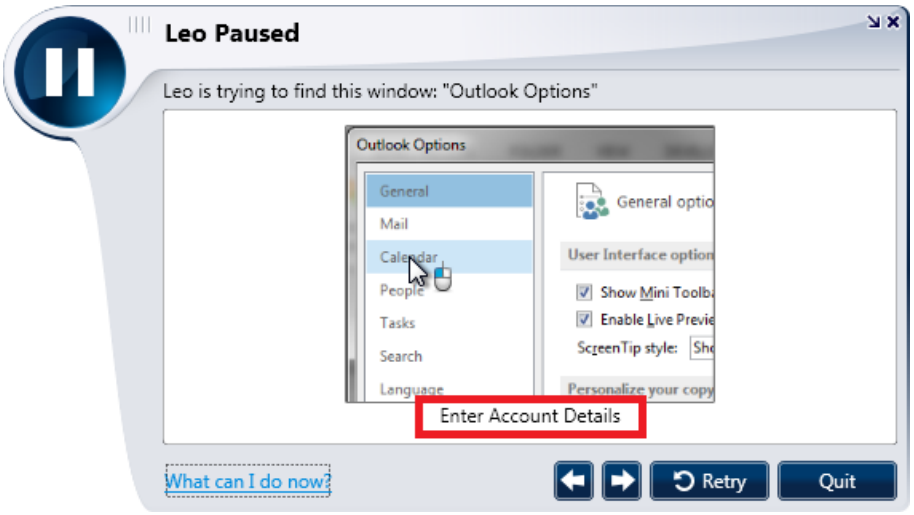
  - Do It/Guide Me from a selected step (Figure 71):

Figure 67: Step Name in Player Step Selection



- Leo Paused step image (Figure 72):

  Figure 68: Leo Paused Step Image



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

For more information about exported wizards and wizard document templates, see section 3.8.4.

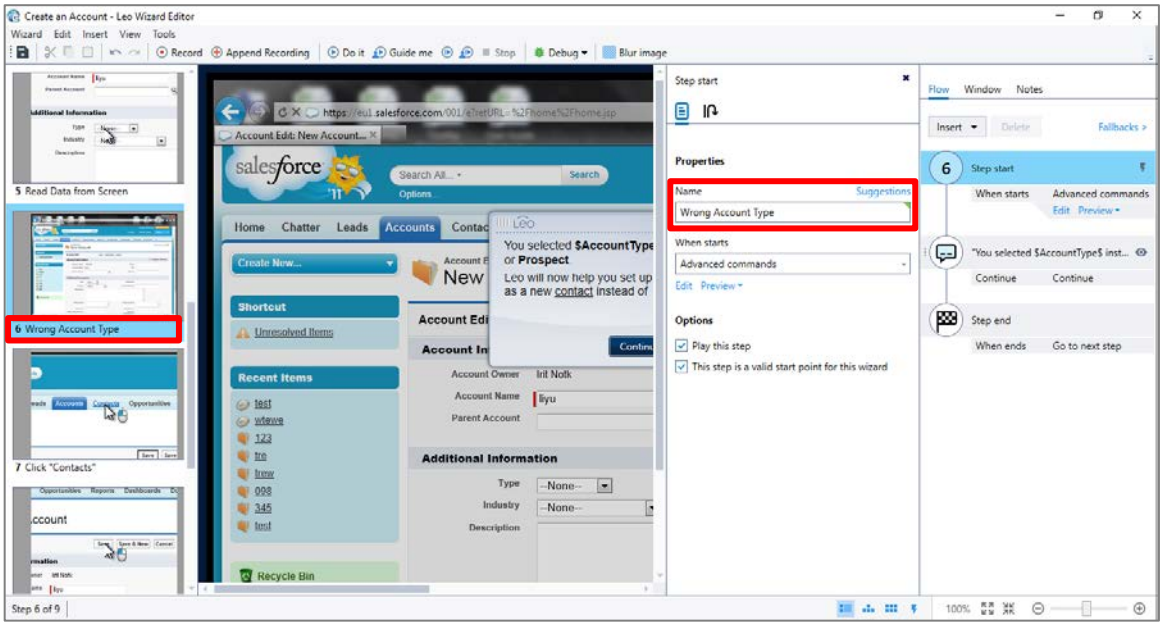
Sensor step names are not visible to end users in Leo Player.

Customize the default step name by doing the following:

1. Select the step that you want to rename.
2. In the **Flow** pane, access the **Properties** pane, as described in Section 3.3.2.4.
3. In the **Property** tab's **Name** field, type a name for the step.

   The updated step name appears in the **Navigation** pane under the step thumbnail (Figure 73).

Figure 69: Step Name



3.6.2.1.1.1.2 Step Start Action

The Start action enables you to set one of the following commands with which to open the selected step:

- **Continue**: When the step begins, Leo continues the flow to trigger other actions in the step in the order determined by the content author.
- **Advanced Commands**: When the step begins, Leo triggers a predefined set of advanced commands. When the advanced commands are completed, Leo continues by executing blocks, core action and bubbles (if any) in the order determined by the content author.

Set the step start action by doing the following:

1. Select the step.
2. From the **Flow** pane, do one of the following:
   - Access the **Properties** pane, as described in Section 3.3.2.4.
   - In the **Flow** tab, click the **Step Start** action.
3. From the **Start Action** dropdown list, select a command.

3.6.2.1.1.1.3 Step Start Options

The following sections describe additional Step Start options.

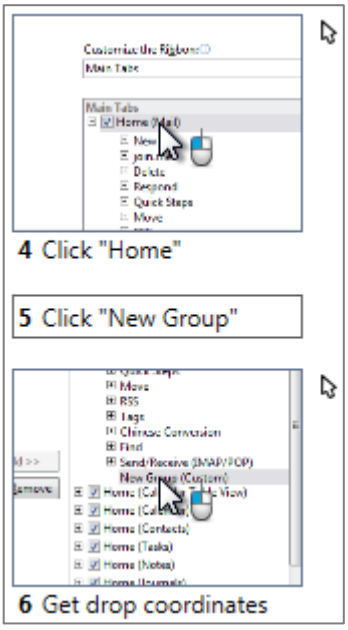
3.6.2.1.1.1.3.1 Enable a Disabled Step

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

All steps are enabled by default. Steps containing recorded scrolling are disabled by default. For information about scrolling in Leo, see Section 3.1.1.6.

Disabled steps are hidden in the **Navigation** pane by default (Figure 74).

Figure 70: Disabled Steps Hidden



To re-enable a disabled step in the wizard flow, select any other core action for the step.

In Diagram view mode, disabled steps are always completely hidden and cannot be shown.

For information about how to display disabled steps in the Normal view mode's **Navigation** pane, see Section 3.5.3.2.

3.6.2.1.1.3.2 Valid Wizard Starting Points

Leo 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 decision is controlled from the checkbox named **This step is a valid start point for this wizard**.

3.6.2.1.1.2 Step Start Fallbacks

The step start fallback events are:

- Window not Found
- Window Blocked

Step Start fallbacks are not available for embedded or launched wizards.

For a description of these fallback events, see Section 3.3.2.3.1.

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

3.6.2.1.2 Blocks

Blocks are optional actions that, if added, are always triggered immediately after the Step Start action. Blocks can be added and removed, but cannot be moved within the flow.

Multiple blocks can be added to a single step.

3.6.2.1.2.1 Example Uses of Blocks

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

- Ensure compliance
- Augment application functionality

The following is an example of block usage that both ensures compliance and augments application functionality:

In this scenario, when completing a form, the user clicks **OK** to submit it. The purpose of the block is to sense when the user clicks **OK**, in order to hold on to the click before allowing it to get through to the application, validate the form, and then decide whether to release the click to the application or deny it and prompt the user to fix any incorrect data.

- **Development**: In Studio, the content developer:
  a. Records the form window in a sensor/wizard.
  b. Applies a block to the form's **OK** button to sense the user's click.
  c. Applies a read-from-screen validation on the form fields.
- **Runtime**: In the user's environment:
  a. When the user completes the form, the user clicks **OK** to submit it.
  b. When the user's click on **OK** is sensed, the block suspends and stores the click to memory, not letting it get through to the application.
  c. The wizard/sensor runs a read-from-screen validation, to ensure that the form was completed correctly.
  d. If the form was completed correctly, Leo releases the user's click and allows it to get through to the application. If the form was not completed correctly, Leo prompts the user to fix it and then re-starts the block process from step a.

3.6.2.1.2.2 Block Types

Leo blocks are used to sense when a user performs an action on the target application, such as click a button or press a keyboard combination. The block senses this action and performs a logic accordingly. The block determines whether to allow the user's action to go through to the target application, or to deny it. Leo can also store the user's action to memory, and apply a logic to it.

Blocks can be added in wizards or multi-step sensors, and can be triggered either by a mouse click or a keyboard combination.

The Leo block types are:

- Block Properties

The block properties for each block type are described in the following sections.

- Mouse Block: Triggered when the user clicks a blocked area, window or the screen in the target application.
  Detection of the blocked GUI object follows the same click position criteria as any Leo-detected object.
- Key Block: Triggered when the user presses a key combination in the target application.

3.6.2.1.2.3 Block Properties

The block properties for each block type are described in the following sections.

3.6.2.1.2.3.1 Mouse Blocks

Mouse blocks are triggered when the user clicks a blocked area, window or the screen in the target application.

The mouse block types are:

- Object Block: A predefined area on the active application window is blocked.
- Window Block: The active application window is blocked.
- Screen Block: The user's entire screen is blocked.

The mouse block properties are:

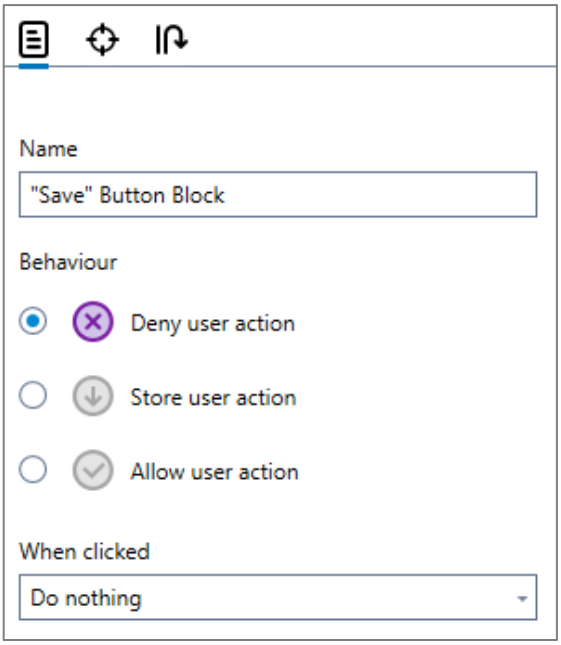
- **General**:
  - Name: The block name serves for reference. It is only visible to the content developer in Leo Studio, not to end users in Leo Player.
  - Event: The **When Clicked** event indicates how the block will be executed (Figure 75):
    - Behavior: When intercepting the user click, how to handle it:
      - Store: Stores the user click to memory in order to allow/deny it at a later time, according to a predefined logic.
      - Allow: Release the click so that it gets through to the application.
      - Deny: Denies the click from getting through to the application. The user's click is dismissed as if it did not occur.

The Deny options are described in the Window Block and Screen Block sections.

    - How to continue: Determine what to do once the click is intercepted.

The **Remove all blocks** option is activated per wizard run. All removed blocks are restored the next time you run the wizard.

Figure 71: Block Properties (Object Block Example)



3.6.2.1.2.3.1.1 Object Block

An object block blocks a predefined area on the active application window. Object blocks are invisible when run from Leo Player, but are indicated by a purple bounding box when run from Leo Studio.

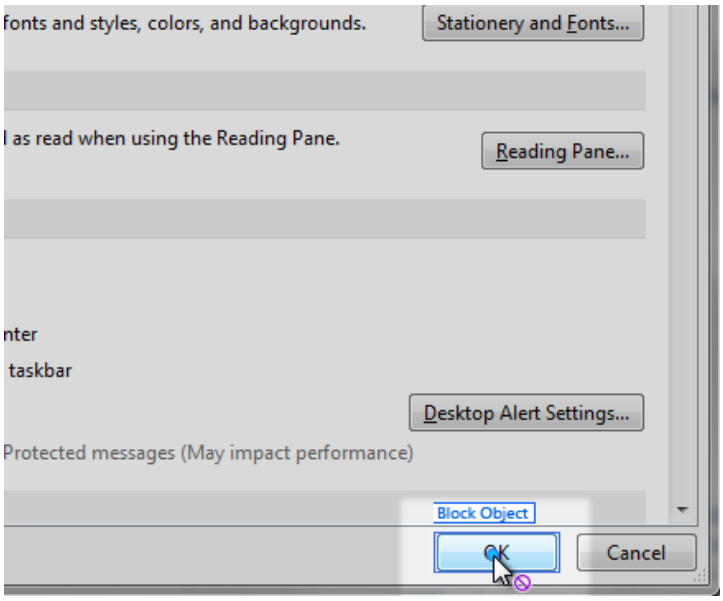
Object blocks can be combined with window or screen blocks.

In addition to the properties described in section 3.6.2.1.2.3, you can customize the object block's bounding box around the object that you want to block. This is

especially useful for steps in which the block position is manually determined using offset (as described in Section 3.6.2.1.4.9.6). In such cases, the bounding box might need to be changed or expanded.

Figure 76 shows an object block in the Studio Editor:

Figure 72: Object block in Studio Editor



You can resize and reposition the block's bounding box around the blocked object.

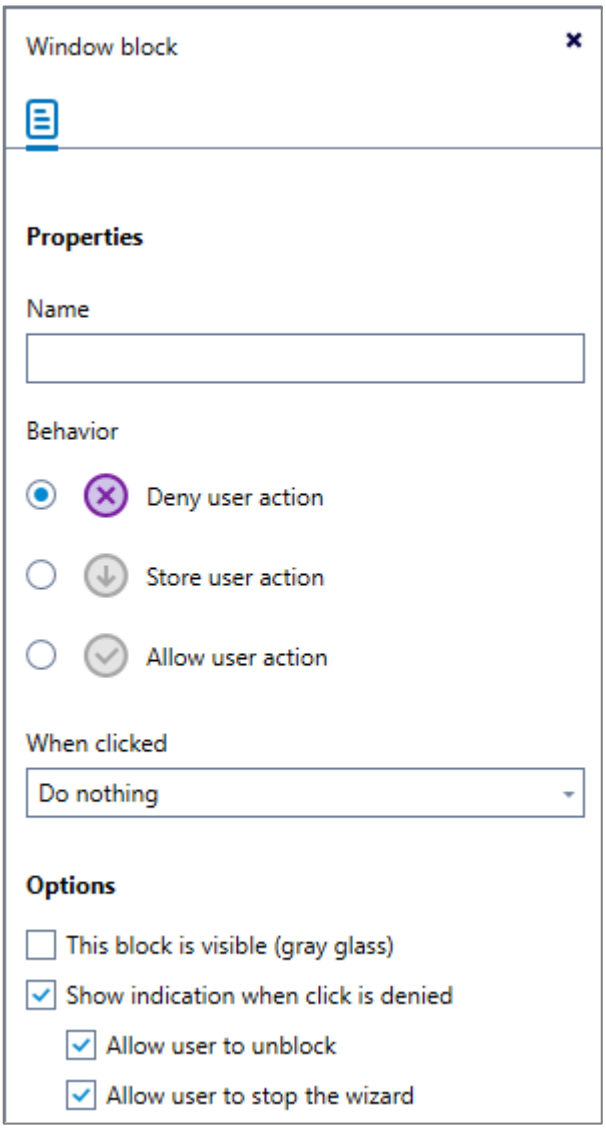
3.6.2.1.2.3.1.2 Window Block

A window block blocks the window of the active application. The window blocked is the one recorded in the step where the block was applied.

In addition to the properties described in section 3.6.2.1.2.3, you can customize the window block's Deny event as follows:

- **This block is visible (gray glass)**: Makes the entire block visibly gray.
- **Show indication when click is blocked**: When the block is clicked, Leo shows a configurable notification that allows the user to do one or both of the following:
  - Remove the block
  - Stop the wizard

Figure 73: Window Block Deny Properties

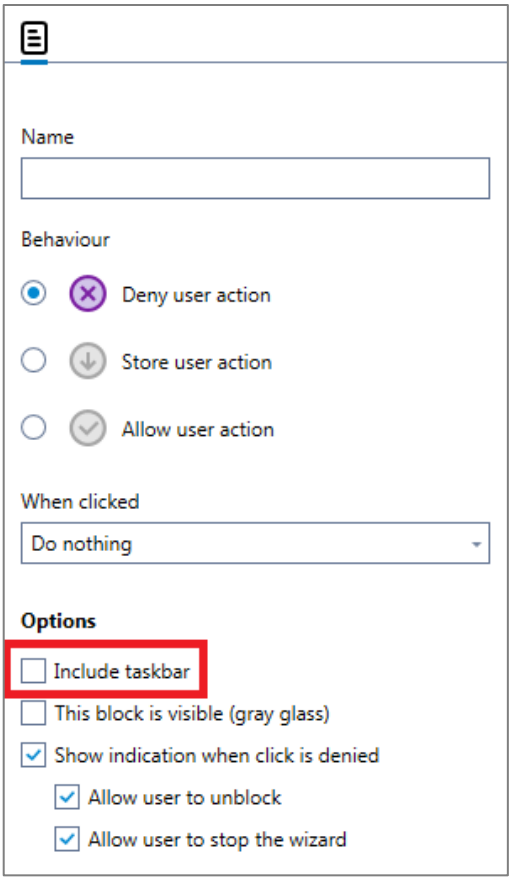


3.6.2.1.2.3.1.3 Screen Block

A screen block blocks the user's entire screen, regardless of the applications currently open or active on the screen.

In addition to the properties described in section 3.6.2.1.2.3, you can decide whether to include the taskbar in the block. You can also customize the window block's Deny event, as described in Section 3.6.2.1.2.3.1.2.

Figure 74: Screen Block Properties with Taskbar Option



3.6.2.1.2.3.2 Key Blocks

Key blocks are triggered when the user presses a keyboard key combination that affects the target application.

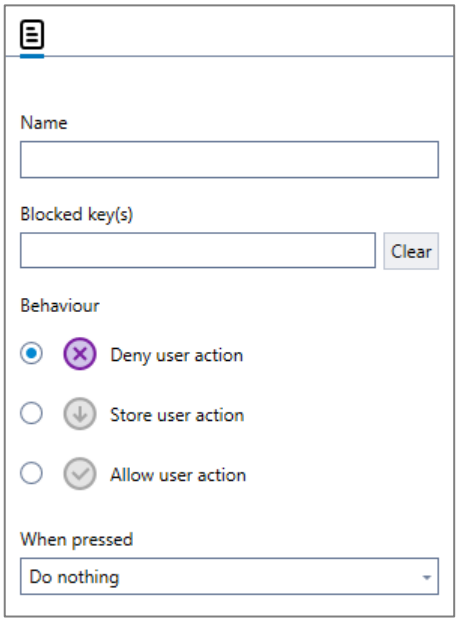
The key block properties are:

- **Name**: The block name serves for reference. It is only visible to the content developer in Leo Studio, not to end users in Leo Player.
- **Blocked Keys**: Leo intercepts a key combination specified by the content author (e.g. **ALT+P**).
- **Event**: The **When Pressed** event indicates how the block will be executed (Figure 79):
  - Behavior: When intercepting the key combination, how to handle it:
    - Store: Stores the key combination to memory in order to allow/deny it at a later time, according to a predefined logic.
    - Allow: Release the key combination so that it gets through to the application.

- Deny: Denies the key combination from getting through to the application. The user's keystrokes are dismissed as if it did not occur.

- How to continue: Determine what to do once the key combination is intercepted.

Figure 75: Key Block Properties



3.6.2.1.2.4 Block Position

The block position properties are identical to the core position properties. For a description of these properties, see Section 3.6.2.1.4.9.

3.6.2.1.2.5 Block Fallbacks

The block fallback event is Object Not Found.

Block fallbacks are not available for window, screen, or key blocks.

For a description of this fallback event, see Section 3.3.2.3.1.

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

3.6.2.1.2.6 Block Procedures

3.6.2.1.2.6.1 Accessing the Block Properties Tab

Access a block's **Properties** tab by doing the following:

1. Select the step containing the block that you want to edit.
2. From the **Flow** pane, do one of the following:
   - Double-click the block whose properties you want to access.
   - Right-click the block whose properties you want to access, and select **Properties**.

The **Properties** pane appears, displaying the block **Properties** tab.

3.6.2.1.2.6.2 Adding a Block

Add a block in to a wizard or sensor step by doing the following:

1. Select the step.
2. In the **Flow** pane, click **Insert** and select the type of block to add:

- Mouse block: In the **Display** pane, click the area where you want to add the block.

  The block appears in the **Display** pane and in the **Flow** pane.
  For information about mouse block properties, see Section 3.6.2.1.2.3.
- Key block: The block appears in the **Flow** pane.
  For information about key block properties, see Section 3.6.2.1.2.3.1.2.

3.6.2.1.2.6.3 Deleting a Block

Delete a block by doing the following:

1. Select the step.
2. Do one of the following:
   - In the **Flow** pane, click the block and press **DELETE**.
   - In the **Flow** pane, right-click the block and select **Delete**.
   - From the **Display** pane, select the block and press **DELETE**.

3.6.2.1.2.6.4 Customizing an Object Block's Surrounding Box

Customize an object block's surrounding box by doing the following:

1. Select the step whose highlight box you want to customize.
2. Access the block's **Position** tab, as described in Section 3.6.2.1.4.9.8.1.
3. In the **Surrounding box** area, click **Customize**.
   In the **Display** pane, the default block box appears with a **Block Object** label around the detected object.
4. Drag and resize the bounding box around the object you want Leo to block.

To return to the default blocked object box, click **Delete (Use Default)**.

3.6.2.1.3 Bubbles

Bubbles are optional actions that, if added, are triggered either before or after a core action. Bubbles can be added, removed, and moved.

Multiple bubbles can be added to a single step.

Bubbles are textual descriptions or instructions that appear on the user's screen when running a Leo 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 Leo 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 Leo Wizard or to other Leo Wizards in Leo Studio. You can add multiple bubbles to a step and choose where to position them in the screen.

The following sections describe bubbles, their properties, and how to use them.

3.6.2.1.3.1 Bubbles and Mouse/Keyboard Control

As a content 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 Leo?

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, hover, and type on the application for the duration of the bubble action.

If the bubble is controlled by Leo, this means that Leo 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 Leo to control it.

3.6.2.1.3.1.1 User-Controlled Timing

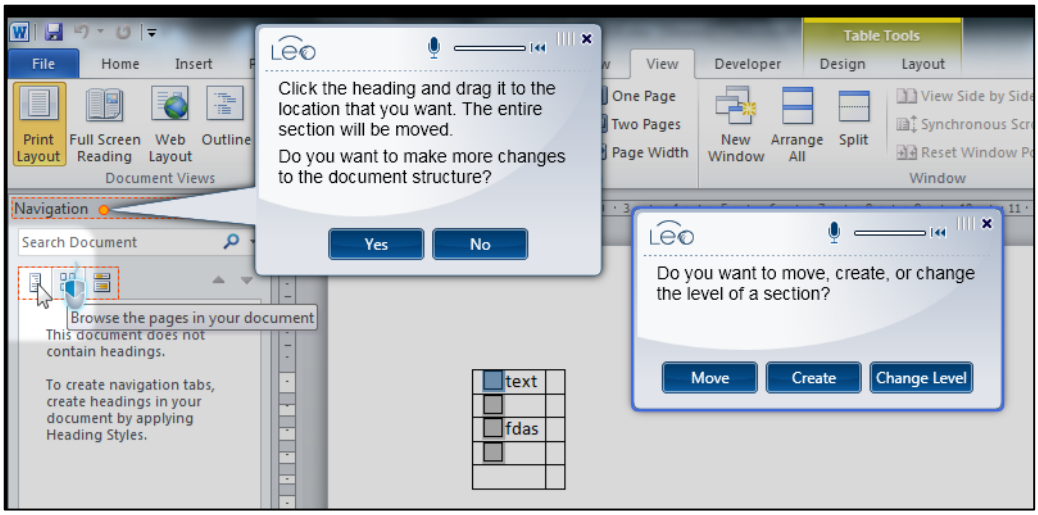
A bubble is user-controlled when it is set to hide on button click.

User-controlled bubbles appear before or after the step's core action or another bubble, and disappear when the user clicks one of their buttons. Once the bubble disappears, Leo resumes control of the mouse and keyboard, and proceeds to the next action in the wizard flow.

User-controlled bubble characteristics are:

- Contain interactive buttons
- Can contain input fields
- Appear one at a time in a Leo Wizard, in the order defined by the content developer
- Cannot appear at the same time as the core action or another user-controlled bubble
- Can appear at the same time as Leo-controlled bubbles
- Can be anchored or non-anchored (Figure 80)
- Can proceed when clicking/hovering the anchored object

Figure 76: User-Controlled Bubbles



3.6.2.1.3.1.2 Leo-Controlled Timing

A bubble is Leo-controlled when it is set to hide by any Hide property except **On button click**.

Bubbles that do not pause Leo runtime. Leo 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, Leo is paused.

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

Leo-Controlled bubble characteristics are:

- 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

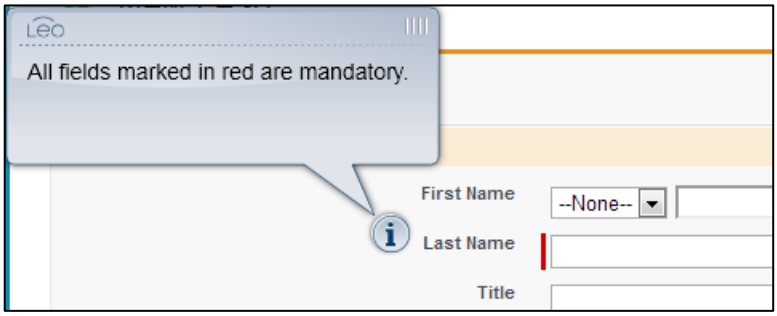
3.6.2.1.3.2 Bubble Types

The types of bubbles that you can insert into the step flow are:

- **Bubble**: A Leo- or user-controlled bubble that floats on the window, not pointing to any specific area in the window. The timing of this bubble can be controlled by either Leo or the user.
- **Anchored Bubble**: A Leo- or user-controlled bubble with a callout anchor that points to a specific area in the window.
- **Tooltip**: A Leo-controlled bubble that informs the user of additional information or validation errors. The tooltip appears as an icon on the screen (Figure 81), attached to an anchored bubble that appears when the user hovers over the icon with the mouse cursor (Figure 82).

Figure 77: Tooltip on Target Application

Figure 78: Tooltip on Target Application when Hovered Over



While the tooltip's timing is Leo-controlled, the user must be allowed control of the mouse while the tooltip is displayed, in order to hover over the tooltip icon and view its bubble. Therefore, tooltips must be timed to appear in one of the following ways:

- With a previous user-controlled bubble
- In a step whose core action is **Wait**.
  For more information about Wait steps, see Section 3.6.2.1.4.4.4.4.

Other tooltip characteristics are:

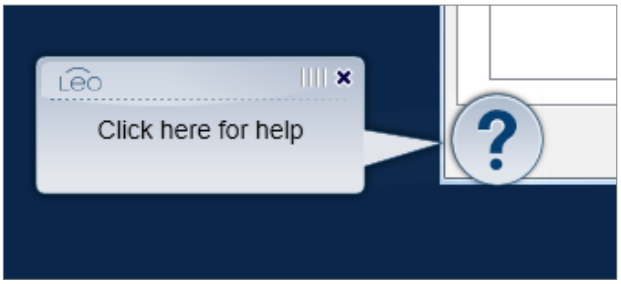
- Do not contain buttons or input fields.
- Are useful for tips and explanations that users can expand if needed.
- Cannot appear at the same time as the core action.
- Can appear at the same time as other bubbles or tooltips.
- Cannot be timed to disappear after a few seconds.
- Can be set to hide at the end of a subsequent step.
- Cannot be non-anchored or have a transparent anchor.
- The tooltip icon can be modified (Figure 83).

⚠ Tooltips can only be displayed with other bubbles, but never on their own. The only exception where a tooltip can be used on its own is in **Wait** type steps.

Figure 79: Tooltip Icons

- **Help Button**: A Leo-controlled button that allow the user to simultaneously display or hide all Leo-controlled bubbles and tooltips that are currently displayed. The Help button is anchored to a Hint bubble on the screen (Figure 84).

Figure 80: Help Button



By default, the Help button hides all Leo-controlled bubbles that precede it to it until clicked.
While the Help button's timing is Leo-controlled, the user must be allowed control of the mouse while the button is displayed, in order to hover over the button and view its hint bubble. Therefore, Help buttons must be timed to appear in one of the following ways:

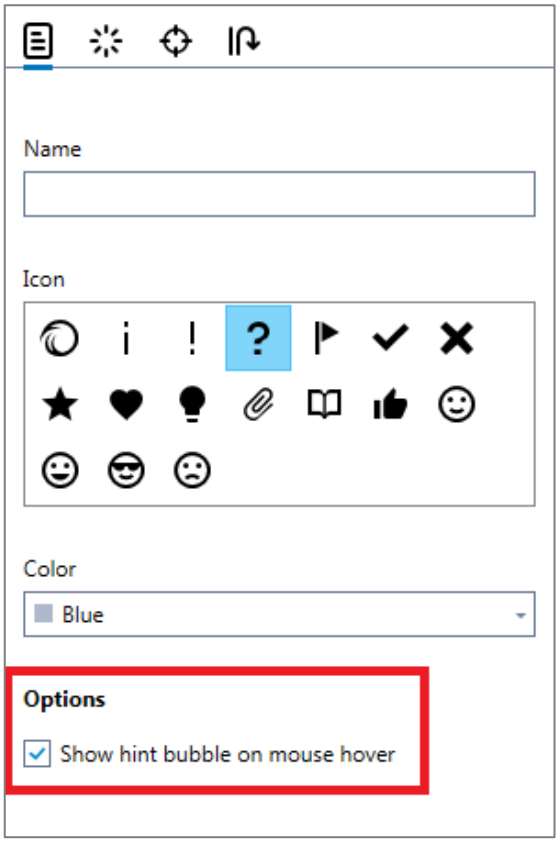
- With a previous user-controlled bubble
- In a step whose core action is **Wait**.
  For more information about Wait steps, see Section 3.6.2.1.4.4.4.4.

The Help button can be timed to disappear after a few seconds, or when another bubble is closed.
Other Help button characteristics are:

- Has an anchored Hint bubble, which has no buttons or input fields.
- Useful for tips and explanations that do not require user interaction.
- Can only appear with other Leo-controlled bubbles and/or tooltips.

- Can only be timed to disappear at the end of the step or with a preceding bubble.
- By default, when the Help button appears, all preceding Leo-controlled bubbles and tooltips are hidden until the Help button is clicked.
- The user can move the Help button around the screen during wizard/sensor runtime.
- The button's Hint bubble can be disabled (Figure 85).

Figure 81: Disabling Help Button's Hint Bubble



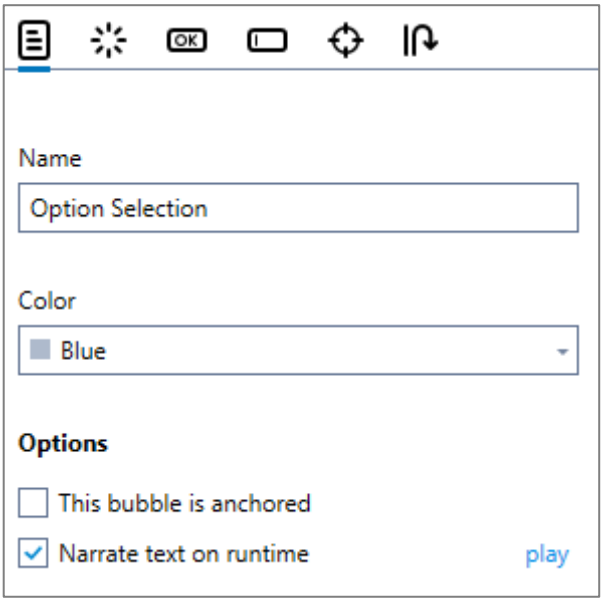
- Cannot appear at the same time as the core action.
- The button icon can be modified (Figure 83).

3.6.2.1.3.3 Bubble General Properties

The bubble general properties appear in the **Properties** pane's **Properties** tab, which contains the following options (Figure 86):

- **Bubble Name**: An optional label that identifies a bubble according to its purpose
- **Bubble Color**: An optional color that identifies the bubble type according to its purpose.
- **Bubble Options**: Additional options related to the bubble.

Figure 82: Bubble General Properties



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

For information about setting the default bubble options, see Section 3.7.4.

3.6.2.1.3.4 Bubble General Procedures

3.6.2.1.3.4.1 Accessing the Bubble Properties Tab

Access a bubble's **Properties** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Properties**.

The **Properties** pane appears, displaying the bubble's **Properties** tab.

3.6.2.1.3.4.2 Editing the Bubble Name

Edit the bubble name by doing the following:

1. Access the bubble's **Properties** tab as described in Section 3.6.2.1.3.4.1.
2. In the **Name** field, edit the bubble name.

This optional setting is only visible in Leo Studio.

3. Click **OK**.

3.6.2.1.3.4.3 Inserting an Image into a Bubble

The image size cannot be modified after adding it to the bubble. To modify the image size, resize it in an image editor before adding it to the bubble.

1. Access the bubble's text editing mode by performing the procedure described in Section 3.6.2.1.3.15.2.
2. In the bubble's **Text Editing** toolbar, click the **Image** icon (see Figure 101) and select the image to insert.
3. Click **OK**.
   The image is inserted into the bubble.

The image size cannot be modified within the bubble. To modify the image size, resize it in an image editor before inserting it into the bubble.

3.6.2.1.3.4.4 Inserting a Hyperlink into a Bubble

You can insert links into bubbles to help the user access information that is relevant to the Leo Wizard.

Insert a hyperlink into a bubble by doing the following:

1. Access the bubble's text editing mode as described in Section 3.6.2.1.3.15.2.
2. In the bubble's **Text Editing** toolbar, click the **Link** icon (see Figure 101).
   The **Link** dialog box appears (see Figure 28).
3. In the Link dialog box's Text to display field, type the text that you want to display for the link. For example:
   Microsoft Outlook 2010 Software
4. In the Link to field, type or copy and paste the link address. For example:
   http://office.microsoft.com/en-us/outlook/

To go to the address you inserted, click the **Go** button.

5. Click **OK**.
   The link is inserted into the bubble.

3.6.2.1.3.4.5 Adding or Removing a Bubble Anchor

To add an anchor or remove it from a bubble, do the following:

1. Access the bubble's **Properties** tab as described in Section 3.6.2.1.3.4.1.
2. Select or clear the **This bubble is anchored** checkbox.
   The bubble anchor is added or removed based on your selection.
3. Continue with editing the anchor position as described in Section 3.6.2.1.3.13.

3.6.2.1.3.4.6 Using a Transparent Anchor

To hide a non-anchored bubble when an object on the screen changes or is hidden by a window, use an anchored bubble with a transparent anchor. The bubble will get the same Show/Hide settings as any anchored bubble, but will appear non-anchored to the user.

Apply a transparent anchor to a bubble by doing the following:

1. Access the bubble's **Properties** tab as described in Section 3.6.2.1.3.4.1.
2. Verify that the selected bubble is anchored. If it is not anchored, select the **This bubble is anchored** checkbox.
3. From the bubble **Properties** tab, select the **Use Transparent Anchor** checkbox and click **OK**.

If you do not see this checkbox, verify that the bubble you selected is anchored.

The bubble's anchor remains visible in Leo Studio but is invisible when the wizard is run.

3.6.2.1.3.4.7 Editing the Bubble Narration Properties

When a Leo Wizard is played and a bubble appears, the text in the bubble is set to be narrated by default. Text that is formatted in angle brackets (**<>**) is not narrated.

Edit and test the narration of the bubble text by doing the following:

1. Access the bubble's **Properties** tab as described in Section 3.6.2.1.3.4.1.

2 Select or clear the **Narrate text on runtime** checkbox.
3 To play back the bubble text narration, click **Play**.

To exclude a portion of text from being narrated, edit the bubble text by adding angle brackets (<>) around the text to exclude. For information on how to edit bubble text, see Section 3.6.2.1.3.15.2.

4 Click **OK**.

3.6.2.1.3.5 Bubble Show/Hide Properties

The conditions for showing or hiding bubbles and bubble anchors, and fallbacks for when these conditions are not met. For information about how to edit the bubble show/hide properties, see Section 3.6.2.1.3.6.

You can determine at which point to show or hide a bubble in a step. You can also decide whether to show the bubble if the window is not detected.

The bubble Show/Hide properties are:

- **Show**: Determine at which point in the step 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**: Determine when to hide a bubble. These settings only apply to Leo-controlled bubbles. User-controlled bubbles are hidden when a button in the bubble is clicked, or if a fallback is applied:
  - On Button Click: Hide the bubble when the user clicks one of the bubble buttons.

This option only applies to bubbles and anchored bubbles.

  - With Previous Bubble: If the bubble is played with the previous bubble, hide both bubbles at the same time.
  - After [X] Seconds: Enter the number of seconds after which to hide the bubble.

This option only applies to bubbles and anchored bubbles.

  - When Narration Ends: Hide the bubble when the audio narration stops, or, if narration is disabled in the bubble, specify the number of seconds after which to hide the bubble.

This option only applies to bubbles and anchored bubbles.

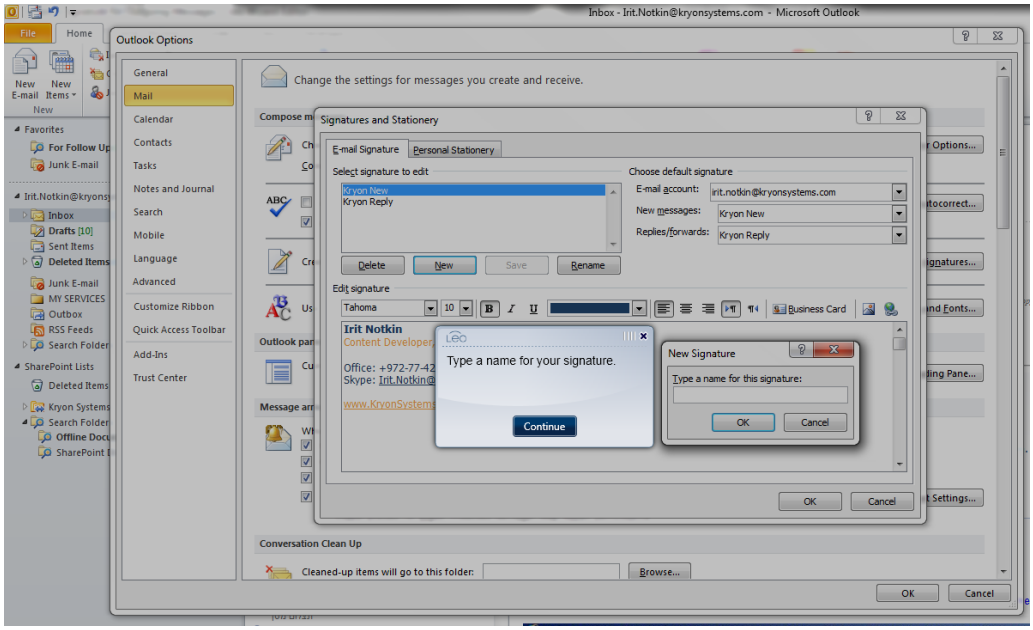
  - At the End of this Step: Hide the bubble when the step ends and Leo goes to the next step.
  - At the End of Step [X]: Hide the bubble at the end of a subsequent step. Applicable to Leo-controlled bubbles only. Determines the range of steps during within the bubble will appear.

This option only applies to bubbles, anchored bubbles, and tooltips

- **Options**:
  - Block Screen: The user's entire screen is blocked for as long as the bubble is displayed. The user cannot click anything on the screen around the bubble. During runtime, this feature is indicated by a grayed-out screen around the bubble (Figure 87).

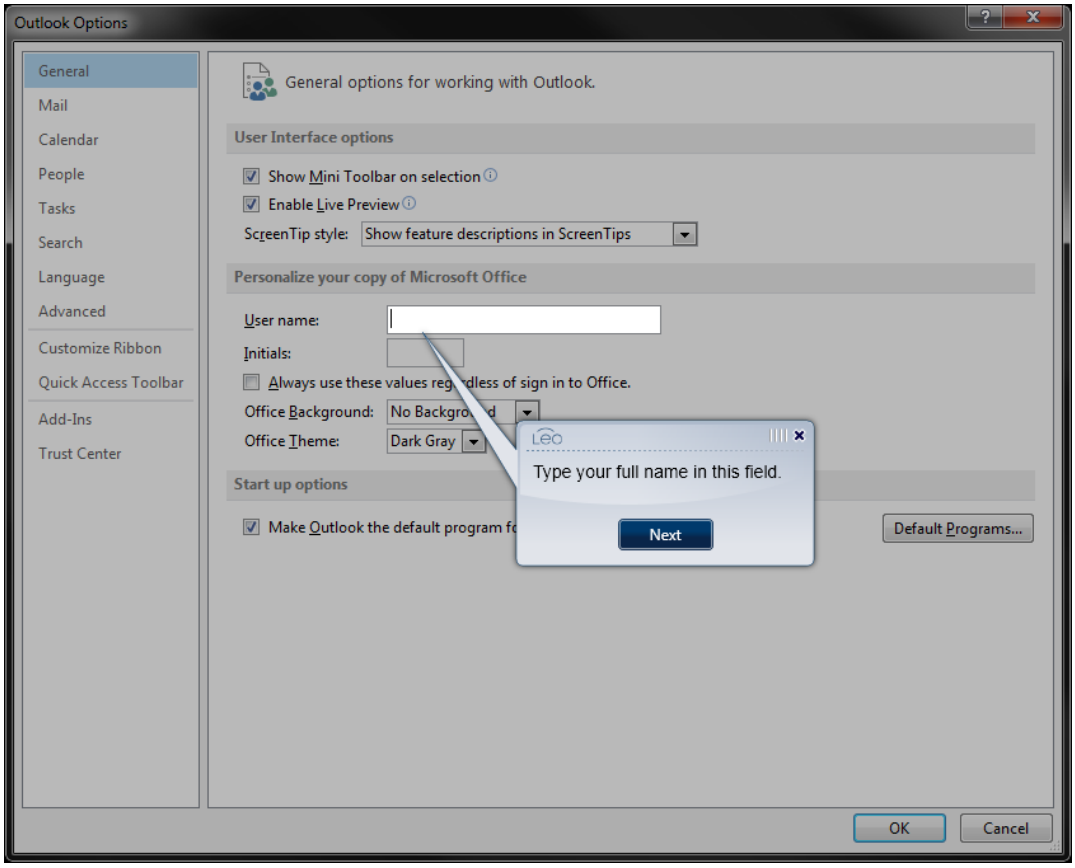
This option only applies to bubbles set to hide on button click.

Figure 83: Blocked Screen with a Non-Anchored Bubble



If the bubble is anchored and its highlight box is enabled, the entire screen around the anchor's highlighted box is blocked (Figure 88).

Figure 84: Blocked Screen with an Anchored Bubble and Highlight Box



For information about customizing an anchored bubble's highlight box, see Section 3.6.2.1.3.13.4.

- Show only if variable is TRUE: Displays the bubble only if a specific variable, which enforces a condition, returns a true value. If the variable returns a false value, the bubble is not displayed to the user.

- Bring window to front: Manually determine the activation of the window to which the bubble is related. The feature options are:
  - Default: Leo automatically determines whether it should bring to the front of the screen the window on which the bubble appears. If the window has already been brought to the front, Leo does not attempt to bring it to the front.
  - Always: Leo 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: Leo never attempts to bring the window to the front of the screen, and displays the bubble if it detects the window, regardless of whether it is brought to the front.

The bubble **Show/Hide** tab is indicated by the **Show/Hide** icon.

3.6.2.1.3.6 Bubble Show/Hide Procedures

3.6.2.1.3.6.1 Accessing the Bubble Show/Hide Tab

Access a bubble's **Show/Hide** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit, and then click the **Show/Hide** tab.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Show/Hide**.

The **Properties** pane appears, displaying the bubble's **Show/Hide** tab.

3.6.2.1.3.7 Bubble Button Properties

Bubbles can contain interactive buttons that determine how to continue the wizard flow. The button tells Leo which action to perform when the user clicks it. You can edit the button caption and event.

You can add an unlimited number of buttons to a bubble, and a minimum of 0. By default, a bubble contains one **Next** button, set to continue to the next action.

The bubble **Buttons** tab is indicated by the **Buttons** icon.

The following sections describe the button commands and options.

For information about how to edit buttons in a bubble, see Section 3.6.2.1.3.8.

3.6.2.1.3.7.1 Button Caption

The button caption is the text that appears on a button in the Leo bubble. You can edit this text in the Button tab's **Caption** field. Each button has a unique caption.

For information about how to globally edit the default button caption and other default bubble options, see Section 3.7.4.

3.6.2.1.3.7.2 Button Commands

Each button can be set to perform a specific command.

The following commands can be assigned to a bubble button:

- **Continue**: When the button is clicked, Leo continues to the next step action. This is the default setting.

- **Go to Step**: When the button is clicked, Leo 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.
- **Advanced Commands**: When the button is clicked, Leo triggers a predefined set of advanced commands. When the advanced commands are completed, Leo continues by executing the next step action.
- **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 Leo Wizard early in the flow, or to create a decision point that splits the wizard flow into multiple paths. If the Leo 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 content author.
- **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 content developer chose to design as an alternative ending to the wizard. This command is useful when you want to end the Leo Wizard before the end of the flow, or to create a decision point that splits the wizard flow into multiple paths. If the Leo 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 content author.

3.6.2.1.3.7.3 Single Button Options

The following options can be configured for individual buttons:

- **Validate Input Fields**: For each button, this option allows you to define whether to enforce field validation.
- **Auto-Proceed**: A few seconds after the bubble appears on the user's screen, Leo automatically clicks the bubble button. The number of seconds is configurable. During runtime, this option is indicated by a countdown indicator on the button (Figure 89).

  Figure 85: Auto-Click Countdown

- **Proceed by Key**: This option allows you to activate a keyboard shortcut for the button. When the shortcut is activated, the user can activate the bubble button by pressing the keyboard key assigned to the button. The button can also be activated by clicking it, as any regular button. The keyboard shortcut keys that can be assigned are **TAB** or **ENTER**.
  During runtime, this options is indicated by the keyboard shortcut name under the button (Figure 90):

Figure 86: Button Shortcut Key

- **Require Mouse/Keyboard Action to Proceed**: Leo requires user interaction with the application before allowing the user to close the bubble by clicking its button. The button event is disabled until the user performs a mouse click or keystroke on the application.
  During runtime, this option is indicated as follows:
  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 as is: The button action does not take place. A yellow bar appears under the bubble, providing users with the option to continue despite not having interacted with the application (Figure 91).

  Figure 87: "Require Mouse Click to Proceed" in Bubble

3.6.2.1.3.7.4 Group Button Options

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

- **Lock Buttons**: Leo disables the bubble button for a configurable number of seconds. The user cannot click the button during that time. During runtime, this feature is indicated by grayed-out buttons and a countdown indicator at the top of the bubble (Figure 92).

  Figure 88: Blocked Button

- **Save Clicked Button**: When this checkbox is selected, Leo reports usage statistics for any button clicked in that bubble by the user. This report is available in the Leo Report Generator under **User Runtime Actions**.
- **Pop Out Buttons**: Applicable to anchored bubbles only. When this checkbox is selected, the bubble's buttons are moved from the bubble onto the screen, providing a more intuitive user experience when user input is required.

If you do not use buttons in a bubble, you can configure its behavior by showing or hiding the bubble under specific conditions in the step. For more information about showing and hiding bubbles, see Section 3.6.2.1.3.9.

3.6.2.1.3.8 Bubble Button Procedures

3.6.2.1.3.8.1 Accessing the Bubble Buttons Tab

Access a bubble's **Buttons** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit, and then click the **Buttons** tab.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Buttons**.

The **Properties** pane appears, displaying the bubble's **Buttons** tab.

3.6.2.1.3.8.1.1.1 Adding Buttons to a Bubble

Add buttons to a bubble by doing the following:

1. Access the bubble's **Buttons** tab as described in Section 3.6.2.1.3.4.1.
2. From the **Buttons** area, click **+Add**.
   A button is added to the bubble, displaying the caption **New button**.

3.6.2.1.3.8.1.1.2 Editing Buttons in a Bubble

Edit buttons in a bubble by doing the following:

1. Access the bubble's **Buttons** tab as described in Section 3.6.2.1.3.4.1.
2. Type the button text in the **Caption** field.
3. From the When clicked dropdown list, select a command and click **OK**.
   The bubble buttons are updated with your changes.

3.6.2.1.3.8.1.1.3 Using Pop Out Buttons

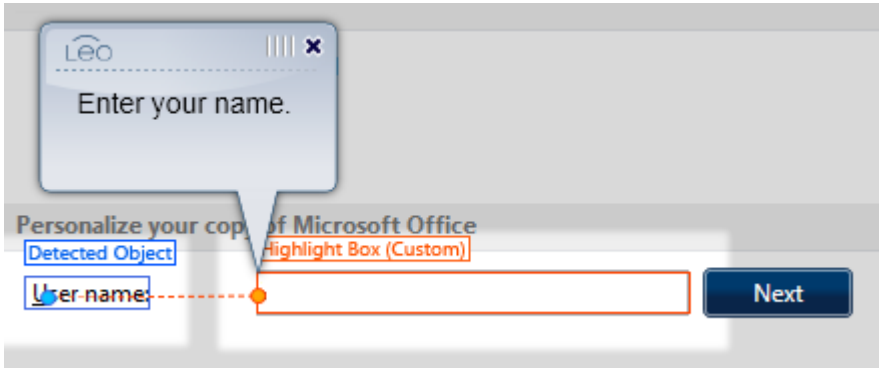
Use pop out buttons by doing the following:

4. Access the bubble's **Buttons** tab as described in Section 3.6.2.1.3.4.1.
1. From the anchored bubble's **Buttons** tab, select the **Pop Out Buttons** checkbox.
2. From the **Pop Out Buttons** dropdown list, select the button location outside the bubble.

The button location is determined in relation on anchored object's highlight box.

The buttons are moved outside the bubble onto the location you specified (Figure 93).

Figure 89: Pop Out Buttons

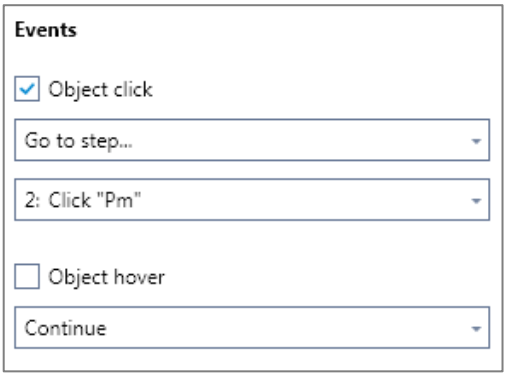


3.6.2.1.3.9 Bubble Events

For each anchored bubble that is set to hide on button click, you can determine actions when a certain event occurs.

The bubble events are available from Buttons tab (Figure 94: Bubble Events)

Figure 90: Bubble Events



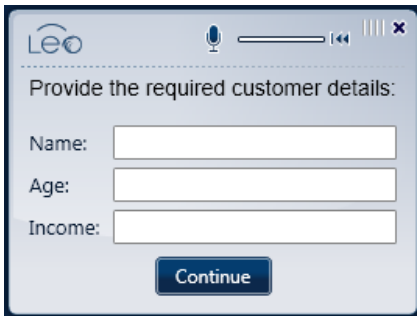
- **Object click**: when user clicks the bubble's detected object.
- **Object hover**: when user hovers above the bubble's detected object.

Each event can be set with a specific command. The available commands are same as the commands available for bubble buttons (See Button Commands)

3.6.2.1.3.10 Bubble Input Field Properties

Leo enables you to add input fields to user-controlled bubbles, which the user can then fill in (Figure 95). The user's input in bubble fields is automatically saved to a variable for use in advanced commands.

Figure 91: Bubble with Input Text Fields



The available input types are:

- **Text**: Allows user input of any type (letters, numbers, blank spaces, special characters, etc.), and displays it exactly as typed by the user.
- For example: **Hello World 123!#**
- **Option**: Allows the user to select one of multiple options displayed as radio buttons.
- **Checkbox**: Allows the user to select an option displayed as a checkbox.
- **Dropdown List**: Allows the user to select one of multiple options displayed in a dropdown menu.
- **Numeric**: Restricts user input to numbers, and displays it in standard, decimal, currency, or percent format.
  For example: **11,123.40%**

- **Date/Time**: Restricts user input to letters and/or numbers, and displays it in a predefined date format.
  For example: **Fri, 21 May 2010 18:04:40 GMT**
- **Custom Format**: Restricts user input to a predefined number and type of characters, and displays it in the custom predefined format.
  For example: **1234** (where the user can only type 4 digits)

The bubble **Input Fields** tab is indicated by the **Input Fields** icon.

For information about how to edit bubble input fields, see Section 3.6.2.1.3.11.

3.6.2.1.3.11 Bubble Input Field Procedures

3.6.2.1.3.11.1 Accessing the Bubble Input Fields Tab

Access a bubble's **Input Fields** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit, and then click the **Input Fields** tab.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Input Fields**.

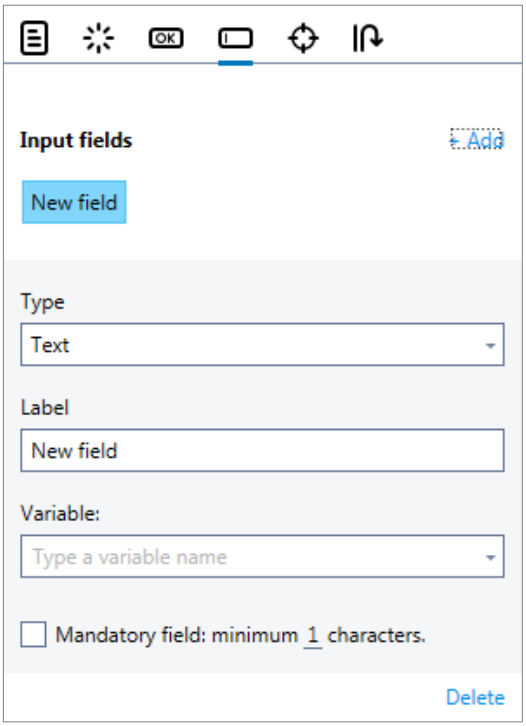
The **Properties** pane appears, displaying the bubble's **Input Fields** tab.

3.6.2.1.3.11.2 Adding Input Fields to a Bubble

Add input fields to a bubble by doing the following:

1. Access the bubble's **Input Fields** tab as described in Section 3.6.2.1.3.11.1.
2. From the **Input Fields** area, click **+Add**.
   A new field is added to the bubble, and appears in the **Input Fields** tab with the field attributes (Figure 96):

Figure 92: Input Fields Tab



3. From the **Type** dropdown list, select a field type.
4. In the **Label** field, provide a name for the new field.

5 In the **Variable** field, type or select a variable to which the user's input will be saved.
6 To designate the field as mandatory, select the relevant checkbox and enter the minimum number of mandatory characters.
If the user skips a mandatory field, Leo will not allow the user to dismiss the bubble until that field is populated with the required minimum number of characters.
7 If applicable to the field type selected, provide a range of values.
The updated field is displayed in the bubble.

3.6.2.1.3.11.3 Deleting an Input Field from a Bubble

Delete an input field from a bubble by doing the following:

1 Access the bubble's **Input Fields** tab as described in Section 3.6.2.1.3.11.1.
2 At the bottom of the tab, click **Delete**.
The field is deleted from the bubble.

3.6.2.1.3.12 Bubble Position Properties

Bubbles and bubble anchors can be moved around within a step by simply dragging them in the **Display** pane in the Leo Studio Wizard Editor. However, the bubble and bubble anchor position can also be set by positioning them at a fixed or relative distance from the window borders.

For anchored bubbles, Leo's visual detection algorithm can be used to identify the position of the bubble on the screen.

The bubble **Position** dropdown list contains the following options:

- **Detected Anchor Object**: Leo identifies the anchor position by visually detecting an image or text. This is the default setting for anchored bubbles.

For non-anchored bubbles, this option is disabled.

- **Fixed Position**: Leo identifies the bubble position by detecting a fixed distance from the window corner. The fixed distance is determined based on the bubble's current distance from that corner.
- **Relative Position**: Leo identifies the bubble position by detecting the bubble's current position in relation to the window size. This is the default setting for non-anchored bubbles.

In certain Leo Wizards, a bubble anchor is required to point to a specific area of the screen. The anchor placement is determined by visual detection of an object, using the same object detection criteria as any detected object.

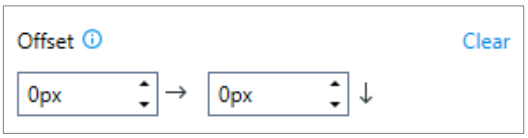
Anchors are sometimes required for areas that do not meet the detection criteria: Large empty fields, non-unique objects or objects whose appearance is dynamic (e.g. dropdown lists). To enable Leo to detect the anchored object for such objects, you must use position offset to place the bubble anchor.

To point to an empty field or large image, place the anchor over a nearby image or text that meets Leo's detection criteria, and then use position offset to adjust the anchor position and cause Leo to detect the correct location.

For a full description of the offset concept, see Section 3.6.2.1.4.9.6.

The **Position Offset** menu (Figure 118) becomes enabled for anchored bubbles whose position is set to **Detected Anchor Object** (see Section 3.6.2.1.3.13.3).

Figure 93: Position Offset in the Bubble Position Tab



By default, the offset is set to 0 pixels.

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

3.6.2.1.3.13 Bubble Position Procedures

3.6.2.1.3.13.1 Accessing the Bubble Position Tab

Access a bubble's **Position** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit, and then click the **Position** tab.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Position**.

The **Properties** pane appears, displaying the bubble's **Position** tab.

3.6.2.1.3.13.2 Editing the Bubble Anchor Offset

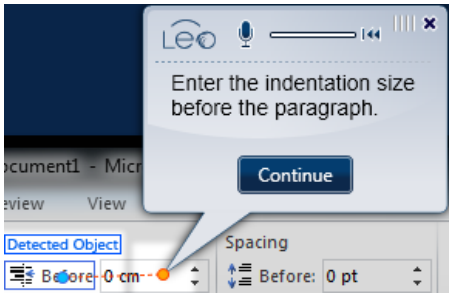
Edit the anchor's position offset in one of the following ways:

- From the **Display** pane:
  a. Select the step whose bubble position you want to edit.
  b. In the Display pane, place your cursor over the blue dot within the box surrounding the detected object.
  c. Press **SHIFT** and click and drag the blue dot to move the anchor position to a new location.
  d. The orange dot indicating the anchor offset position is set to the new location (see Figure 99).

- From the **Flow** pane:
  a. Access the bubble's **Position** tab as described in Section 3.6.2.1.3.13.1.
  b. In the **Offset** area, enter the number of pixels by which to move the anchor position vertically and/or horizontally.
  The orange dot indicating the anchor position is moved by the number of pixels you entered (see Figure 99).

To move the position right and/or down, use positive numbers. To move the position up and/or left, use negative numbers.

Figure 94: Bubble Anchor Offset

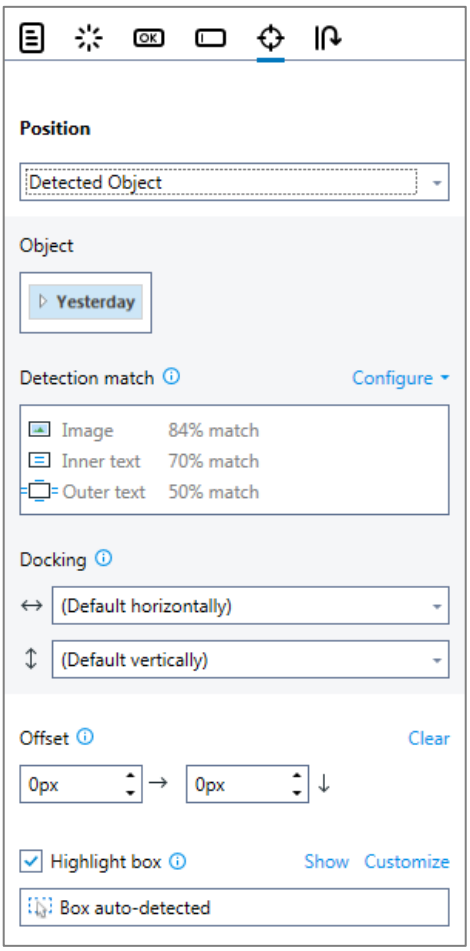


3.6.2.1.3.13.3 Editing the Bubble's Position Settings

Edit the bubble's position settings by doing the following:

1 Access the bubble's **Position** tab as described in Section 3.6.2.1.3.13.1.

Figure 95: Bubble Position Tab (Anchored Bubble)



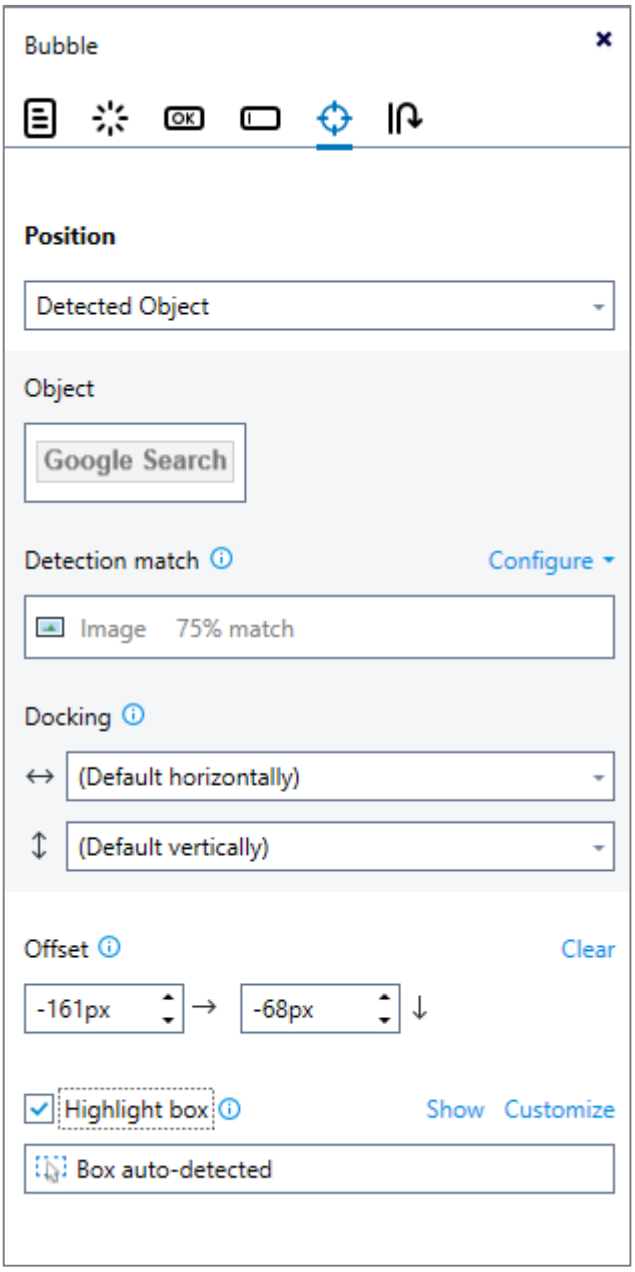
2 From the **Position** dropdown list, select an option.

If you select **Fixed Position**, in the **Fixed Position** graphic click the arrow that represents the corner relative to which the click position will be calculated. The upper-left corner is selected by default.

Figure 96: Fixed Position Graphic

3.6.2.1.3.13.4 Enabling an Anchored Bubble's Highlight Box

In order to enable the Anchored Bubble highlight box, open the Bubble properties tab, and check the option "Highlight box". The identified object will be then marked by a yellow line. If you want to change the marked frame size or location, press the "Customize" button and change the highlighted area accordingly.



3.6.2.1.3.14 Bubble Fallbacks

For each anchored or non-anchored bubble, you can determine fallbacks for showing or hiding the bubble and its anchor.

The bubble fallback events are:

- **If Window Is Not Found**: If Leo is unable to detect the window, choose whether to show or hide the bubble.
- **If Anchor Object Is Not Found**: If the bubble has an anchor and Leo is unable to detect the anchor object, choose whether to show the bubble non-anchored or hide the bubble altogether.
- **If Anchor Object Is Changed**: If the bubble has an anchor and the anchor's target object changes, choose a bubble behavior.
  For example: If the bubble's anchor object is an empty field and the user types text in that field, the anchor object changes. You can choose to ignore the change and continue showing both the bubble and its anchor.
- **If Anchor Object Is Hidden by Window**: If a new window appears during the step, hiding the bubble's anchor object in the process, choose a bubble behavior.
  For example: A bubble's anchor points to a button that, when clicked, opens a new window. The new window hides the button, which is the anchor object. In this case, you can choose to hide the bubble temporarily while its anchor object is hidden.
- **If Window Is Closed**: If Leo or the user closes the window that the bubble is associated with, choose whether to continue showing the bubble.

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

3.6.2.1.3.14.1 Accessing the Bubble Fallbacks Tab

Access a bubble's **Fallbacks** tab by doing the following:

1. Select the step containing the bubble that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the bubble that you want to edit, and then click the **Fallbacks** tab.
   - From the **Flow** or **Display** pane, right-click the bubble that you want to edit, and select **Fallbacks**.

The **Properties** pane appears, displaying the bubble's **Fallbacks** tab.

3.6.2.1.3.15 Bubble Procedures

3.6.2.1.3.15.1 Adding Bubbles

Add a bubble to a step by doing the following:

1. In the wizard or sensor editor, select the step in which you want to add a bubble.
2. In the **Flow** pane, click **Insert** and select the type of bubble to add.
3. In the **Display** pane, click the area where you want to add the bubble.
   The bubble appears in the **Display** and **Flow** panes.

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

3.6.2.1.3.15.2 Editing Bubble Text

Edit the text in a bubble by doing the following:

1. Select the step containing the bubble that you want to edit.
2. In the **Display** pane, double-click the bubble whose text you want to edit.
   The bubble becomes enabled for editing, and the **Text Editing** toolbar appears above the bubble (Figure 101).

Figure 97: Bubble Text Editing Toolbar

3 Type and format the bubble text.
4 Click anywhere in the **Display** screen to exit text editing mode.

For information about inserting links into bubbles, see Section 3.6.2.1.3.4.3.

3.6.2.1.3.15.3 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's **Detected Object** box to a different location.

For information about how to change the anchor's target object, see Section 3.6.2.1.3.12.

3.6.2.1.3.15.4 Showing and Hiding Bubbles in the Display Pane

In some steps, bubbles in a single step are grouped in one location on the screen. Grouping bubbles causes bubbles to obstruct each other from view in the **Display** pane. To view a bubble that is behind another bubble, you can either move the bubble, bring it to front (or send other bubbles to back), or hide the bubble in the **Display** pane.

Hiding a bubble in the **Display** pane does not delete it. The bubble remains active and visible to end users in Leo Player during wizard runtime.

Hide or show a bubble in the **Display** pane by doing the following

1 Select the step containing the bubble that you want to edit.
2 From the **Flow** pane, click the **Eye** icon for the bubble that you want to show or hide in the **Display** pane.

Bubbles are shown in the **Display** pane by default, with the **Eye** icon enabled.

3.6.2.1.3.15.5 Bringing a Bubble to Front

Bring a bubble to front for editing purposes by doing one of the following:

- From the **Flow** pane, click the bubble that you want to bring to front.
- From the **Display** pane, click any area of the bubble if visible.
- From either the **Display** or **Flow** pane, right-click the bubble and select **Bring to front**.

Bringing a bubble to front does not change its position in the step flow during wizard runtime. The bubble is displayed to users in the same order as listed in the **Flow** pane. For information about reordering bubbles in a step, see Section 3.6.2.1.3.9.

3.6.2.1.3.15.6 Reordering Bubbles in a Step

Bubbles in a step appear to the user in the order in which they are listed in the bubble list, either before or after the core action. You can reorder bubbles in a step so that they appear before or after the core action or another bubble in the step.

Reorder bubbles in a step by doing the following:

- In the **Navigation** pane, select the step containing the bubble that you want to move.
- In the **Flow** pane, click and drag the bubble to a new location before or after the core action or another bubble.

You can also click and drag the core action to reorder it in the step flow.

3.6.2.1.3.15.7 Reordering Bubbles in a Leo Wizard or between Leo Wizards

You can reorder bubbles in a Leo Wizard or from one Leo Wizard to another by cutting and pasting them.

Reorder a bubble in a wizard or between wizards by doing the following:

1. From either the **Display** or **Flow** pane, 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.**
2. In the current wizard or in another wizard, select the step to which you want to move the bubble.
3. Paste the bubble to its new location by doing one of the following:
   - On the Wizard Editor menu bar, select **Edit > Paste**.
   - On the toolbar, click the **Paste** icon.
   - Press **CTRL+V**.

3.6.2.1.3.15.8 Copying Bubbles

You can create new bubbles based on existing bubbles by copying bubbles. You can copy bubbles within a Leo Wizard and from one Leo Wizard to another. Copying bubbles is useful when you want to create new bubbles based on existing bubbles.

Copy a bubble by doing the following:

1. In the **Navigation** pane, select the step containing the bubble that you want to copy.
2. In either the **Display** or **Flow** pane, select the bubble that you want to copy.
3. Do one of the following:
   - On the **Wizard Editor** menu bar, select **Edit > Copy**.
   - On the toolbar, click the **Copy** icon.
   - Press **CTRL+C**.
4. In the current wizard or in another wizard, select the step to which you want to move the bubble.
5. Paste the bubble by doing one of the following:
   - On the Wizard Editor menu bar, select **Edit > Paste**.
   - On the toolbar, click the **Paste** icon.
   - Press **CTRL+V**.

3.6.2.1.3.15.9 Deleting Bubbles

Any bubbles that you added to a Leo Wizard can be deleted in the Wizard Editor window.

Bubble deletion can be undone and redone (see Section 3.5.3.5). The deletion becomes permanent when you save or close the Leo Wizard.

Delete a bubble by doing the following:

1. In the **Navigation** pane, select the step containing the bubble that you want to copy.
2. In either the **Display** or **Flow** pane, select the bubble that you want to delete.
3. Do one of the following:
   - On the Wizard Editor menu bar, select **Edit > Delete**.
   - Press **DELETE**.

   A confirmation message appears.
4. Click **Yes** to delete the bubble.

3.6.2.1.4 Core Action

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. The default action is either a mouse click or keyboard keystrokes.

Leo enables you to choose how to run a step on the application. Your process may require changing the core action of certain steps, so that they run differently from how they were recorded. For example, if you recorded a click in order to capture a menu, but during wizard runtime you need Leo to hover instead of click, you will need to change the core action from Click to Hover.

The core action types are:

- **Mouse Actions**:
  - Click
  - Hover
  - Detect Object
  - Read from Screen
- **Keyboard Action**:
- **Wait**:

If you change the core action for any steps in the Leo Wizard, make sure to run the wizard to verify that the modified step works properly. For more information about running Leo Wizards, see Section 3.7.1.

3.6.2.1.4.1 Click

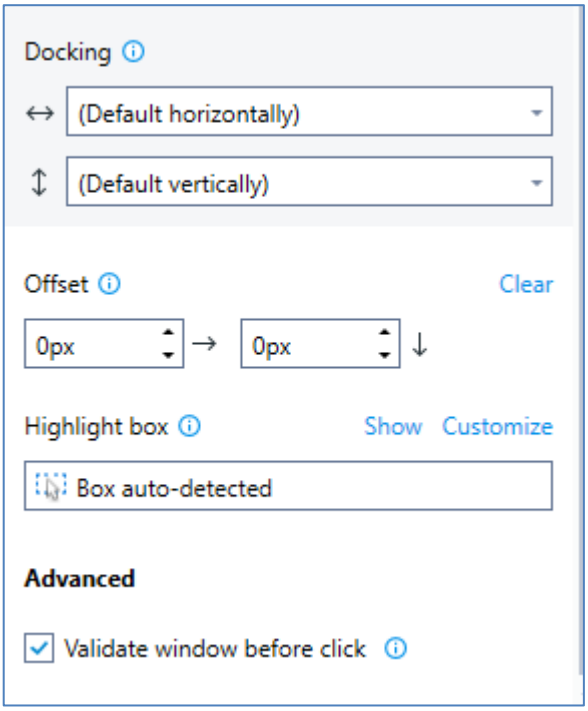
The Click core action is the default for steps recorded by clicking the mouse. It is applied when the recorded action is a mouse action, and reflects the type of mouse click recorded: single, right-click, left-click, double-click, or other.

Not applicable to Leo Sensors.

Skip Windows Validation on Mouse Click Action

As default, before any Mouse Click action Leo validates the current window is the same as the windows which was detected when the step was initiated. In order to deactivate that validation, clear the mark in the property of the Click Core Action, called **Validate windows before click**.

Figure 98: Validate Window Before Click



3.6.2.1.4.2 Hover

The Hover core action moves the cursor on the screen to where a click was recorded in the step, but does not perform the clicking action for the user. This is useful when the mouse needs to hover over options in order to display menus, such as the **Home** tab's **New Items > More Items** menu in Microsoft Outlook 2013.

Not applicable to Leo Sensors.

3.6.2.1.4.3 Detect Object

The Detect Object core action identifies the location of the click in the step, but does not move the cursor. This is useful when you need to make sure that Leo identifies a certain object that is required to continue the process (e.g. the **Developer** tab in Microsoft Office).

3.6.2.1.4.4 Read from Screen

The Read from Screen core action instructs Leo to read the data from a field on the user's screen, and use that data as a variable in advanced commands. The field value is read from the end user's live application when the user plays the wizard in real time.

Read from screen is useful when you want to use information from the user's application screen to determine what Leo should do next.

Setting Leo to read from screen is done for an individual Wizard step that contains a screen capture of the relevant field. Read from Screen is done for specific value types, as described in Section 3.6.2.1.4.4.1.

3.6.2.1.4.4.1 Value Types

Leo can read specific types of values from fields on the screen. For each read-from-screen action, Leo reads one type of value. This value can be text, numbers, or simply a check of whether the field is blank.

Each read-from-screen value type is set up to work with specific types of input data, as described in each of the following sections.

3.6.2.1.4.4.1.1 List

Leo reads the text value in the selected field, compares it to a predefined list in order to properly identify it, and returns the value into a variable.

Example use: If the user selects **A** from the selected dropdown list, Leo performs **a**.

To read text from screen, you must use the following components:

- **Expected Values Condition**: For each value, you can type either the entire value or the part of the value that you want Leo to work with. To work with part of the value, select an option from the **Expected Values Condition** dropdown list.
- **Expected Values**: The list of predefined values that Leo can read and accept input from. This list must include all the values that you need Leo to read.
- **Detection Match**: If you provide only some of the possible values from a list, select **This is a partial list** from the dropdown menu. You can then adjust the Match Threshold from the slider. If you provide all possible values, select **This is the full list**.
- **Match Threshold**: When Leo reads text from screen, it compares it to the nearest value specified in the **Expected Values** list. For more information about the match threshold, see Inner Text Threshold.

Figure 99: Match threshold for partial values list

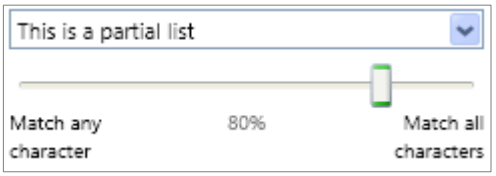


Table 4: List Input Options

Value Type	Accepted Input	Rejected Input

Text	• Items in a predefined list of values, where:
	  • Only English or Hebrew alphabet is accepted.
	  • If your comparison method is **Equals**, each item must be a single type of alphabet.
	  • If your comparison method is **Contains**, **Starts With**, **Ends With**, the item value can include different alphabets.
	• A variable name whose value is a predefined list.
		If the user enters anything other than items in the predefined list, the variable is returned as empty.

3.6.2.1.4.4.1.2 Text

Leo reads any free-form text from the detected text field, and returns the value into a variable.

Functionality of the Text read-from-screen feature depends on the text element's behind-the-scenes technology. For certain text elements, some of the reading methods might not work as expected while others will. This must be tested and determined during the content development process, while using the feature.

Table 5: Text Input Options

Value Type	Accepted Input	Rejected Input
Text	Any free-form text	Images of text

When this feature is used, it accesses the target application's memory process.

This feature is not supported on the following technologies:

Application Technology	Browser
Java Microsoft Silverlight Adobe AIR WebKit	Google Chrome Mozilla Firefox

3.6.2.1.4.4.1.3 Number/Date

Leo reads any **Number** or **Date** from the detected text field and returns it into a variable.

Example use: If the user types an amount greater than **X** in the field, Leo performs **Y**.

Table 6: Number/Date Input Options

Value Type	Accepted Input	Rejected Input
Number/Date	• The numbers **0-9** • The characters **\+-/.,**	If the user enters anything other than accepted input, the variable is returned as empty. If the first or last character is rejected input (e.g. currency sign), Leo removes it. For example: • **€12.3** is turned into **12.3**. • **12.3%** is turned into **12.3**.

3.6.2.1.4.4.1.4 Text Field Empty

Leo checks if the detected text 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

Example use: If the user leaves the selected field blank, Leo does not go to the next step but prompts the user.

Table 7: Is Empty Input Options

Value Type	Accepted Input	Rejected Input
Is Empty	Any input	None

3.6.2.1.4.4.1.5 Checkbox Selected

Leo checks if the detected checkbox is selected or cleared, and returns TRUE or FALSE into the variable as follows:

- Selected checkbox = TRUE
- Cleared checkbox = FALSE

3.6.2.1.4.4.1.6 Radio Button Selected

Leo checks 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

3.6.2.1.4.4.2 Field Offset

If the step core action is Read from Screen, you must specify the field label that Leo must detect and then specify the location of the value in relation to that label, where Leo can read the field value. The location is determined by using field offset in the step containing the position of the field value.

Field Offset is part of the Reading Data from Screen feature. This is one of Leo's more advanced capabilities, and involves a different process than other core actions.

For more information about Field Offset and how to use it to read data from screen, see Section 5.1.4.

3.6.2.1.4.4.3 Field Box

When a step's core action is Read from Screen, the area within which Leo reads the field value is determined by the field box.

For more information about defining the field value, see Section 3.6.2.1.4.4.4.2.

You can resize and reposition the step's field box around the field value position that you want Leo to read data from.

3.6.2.1.4.4.4 Read from Screen Procedures

Set Leo to read from screen by performing the following procedures:

1 Setting the Step Core Action to Read from Screen
2 Setting the Field Value's Position
3 Setting Advanced Commands for the Field Variable

To use Read from Screen, you **must** perform all three procedures in sequential order.

3.6.2.1.4.4.4.1 Setting the Step Core Action to Read from Screen

The first step in reading data from the screen is to set the relevant step's core action. This includes both changing the step's status and defining a variable that will be used to perform a command using the field's value.

Set a step's core action to Read from Screen by doing the following:

4 Select the step.

4 In the **Flow** pane, click **Insert**.

5 From the **Core Action** list, select **Read from Screen**.

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

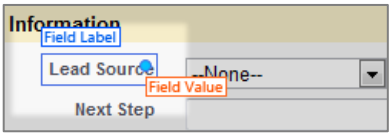
5 From the **Read** dropdown list, select the type of value to read.

If you select **Text**, type all possible values that might be available for that field (for example, the items in a dropdown list).

For a description of the data types and values, see Section 5.1.4.

In the **Display** pane, the **Detected Object** label turns into **Field label**, and a **Field Value** label appears (Figure 104). The **Read from Screen** icon appears in the step thumbnail's upper-left corner.

Figure 100: Field Label and Value



6 In the **Return result in variable** field, type a name for the variable or select an existing variable name from the dropdown list.

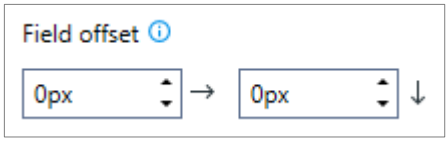
7 Set the field value's position as described in Section 3.6.2.1.4.4.4.2.

3.6.2.1.4.4.4.2 Setting the Field Value's Position

The second step in reading data from screen is to specify the position of the value that you want Leo to read. Field offset tells Leo where to detect the field value, in relation to the location of the field's label. By default, the field value is set to the same location as the field label. Using Field Offset, you can create the required distinction between the field's label and value. Field Offset moves the field value away from the field label by a fixed number of pixels.

Figure 105 shows the **Field Offset** area in the **Position** tab.

Figure 101: Field Offset Settings

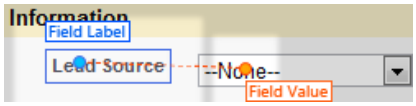


By default, field offset is set to 0 pixels.

Adjust the field offset by doing one of the following:

- From the **Display** pane:
  a In the **Field Label** box, place your cursor over the blue dot.
  b Press **SHIFT** and drag the blue dot to move the field value position to a new location.
  The **Field Value** label and dot are moved (Figure 106).

Figure 102: Field Offset Display



- From the **Position** tab:

a Verify that the step's core action is set to Read from Screen, as described in Section 3.6.2.1.4.4.4.1.

b In the **Offset** area (Figure 105), enter the number of pixels by which to move the field value position vertically and/or horizontally.
The **Field Value** label and dot are moved by the number of pixels you selected (Figure 106).

To move the offset position right/down, use positive numbers. To move it up/left, use negative numbers.

To determine the size of the field from which Leo reads values, see Section 3.6.2.1.4.4.4.4.

3.6.2.1.4.4.4.3 Setting Advanced Commands for the Field Variable

The third step in reading data from screen is to use the defined field value and variable in advanced commands. This step takes the data that Leo read from screen and inserts it into advanced commands that perform advanced actions.

For information about advanced commands and how to set them for the field variable, see Section 5.1.

3.6.2.1.4.4.4.4 Customizing the Field Box

Optionally customize a step's field box by doing the following:

1 Access the core action Position tab, as described in Section 3.6.2.1.4.9.8.1.
2 In the **Field box** area, click **Customize**.
3 In the **Display** pane, the default field box appears with a **Field box** label around the detected object.
4 Drag and resize the box around the field whose value you want Leo to read.

To reposition the field offset within the field box, drag the orange dot.

3.6.2.1.4.5 Keyboard

The Keyboard core action is the default for steps recorded by pressing keyboard keys.

Not applicable to Leo Sensors.

3.6.2.1.4.6 Wait

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

- **Wait for Block Removal**: Leo waits for all blocks to be activated or removed before continuing the wizard flow. This event is only valid if the step contains blocks. As long as a block is not activated or removed, the Wait action persists. Once a block is activated, the wizard flow continues.
For more information about blocks and how to use them, see Section 5.
- **Wait for Window to Close**: Waits for the step's detected window to close, with the ability to specify the maximum amount of time to wait. If the window is not closed by the time specified, Leo continues to the next action in the flow.

By default, the time-limit checkbox is not selected and Leo waits indefinitely.

- **Wait for Object to Disappear**: Waits for the step's detected object to disappear, with the ability to specify the maximum amount of time to wait. If the object does not disappear by the time specified, Leo continues to the next action in the flow.

By default, the time-limit checkbox is not selected and Leo waits indefinitely.

3.6.2.1.4.7 Custom Guide Me Bubbles

When a Leo 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. Leo enables you to customize the Guide Me bubble text for each step.

The default Guide Me bubble text is **Click inside the box**.

Not applicable to Leo Sensors.

3.6.2.1.4.8 Core Action Properties

3.6.2.1.4.8.1 Do It Mouse Movement

Leo enables you to determine how to move the mouse cursor to its click position on the screen. This is useful when the mouse needs to hover over options in order to display menus, such as the **Home** tab's **New Items > More Items > Choose Form...** menu in Microsoft Outlook 2013. The mouse movement options are:

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

Not applicable to Leo Sensors or keyboard steps.

3.6.2.1.4.9 Core Action Position

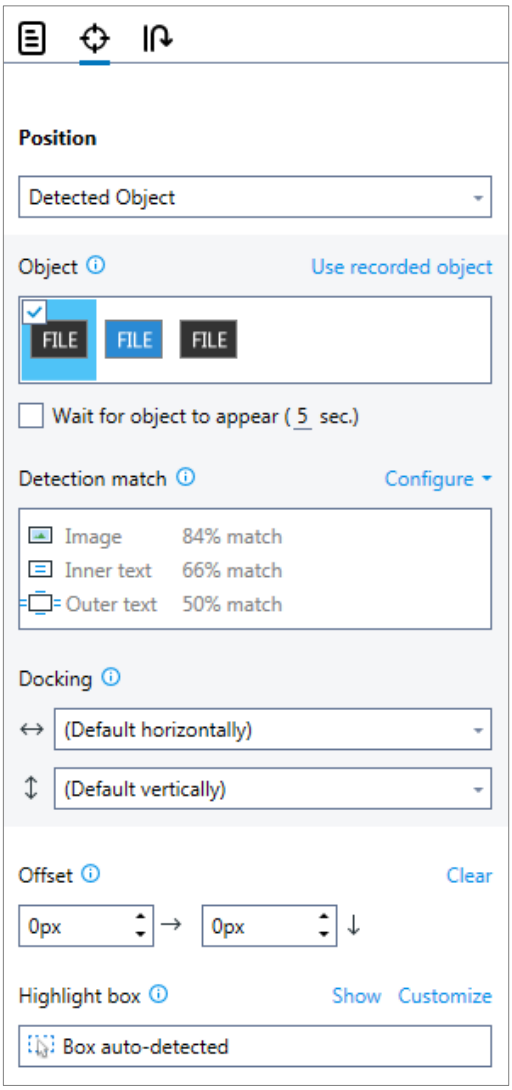
By default, Leo uses a proprietary, patent pending visual detection algorithm to detect objects on the screen.

For a description of how Leo detects objects on the screen, see Section 3.1.2.

The **Position** properties enable you to optimize object detection, thereby improving Leo performance in terms of speed and accuracy.

Figure 107 shows the **Position** tab in the **Properties** pane.

Figure 103: Position tab



The **Position** tab settings are described in the following sections.

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

3.6.2.1.4.9.1 Position

By default, Leo uses its visual detection algorithm to visually identify the object to use for the step, regardless of its location on the screen. You can disable visual detection and set Leo to click at a fixed or relative distance from the window borders.

The **Position** dropdown list contains the following options:

- **Detected Object**: Leo 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**: Leo identifies the click position by detecting a fixed distance from the window corner. The fixed distance is determined according to the click position that was recorded.
- **Relative Position**: Leo identifies the click position by detecting a position that is relative to the window size.

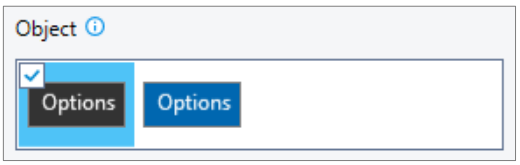
When selecting **Fixed Position**, verify that Leo is able to detect the click position when the window is resized or when a different environment is used. Otherwise, Leo might either click incorrectly or show an error message.

3.6.2.1.4.9.2 Object

When a Leo Wizard is recorded, Leo 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. The object state captured immediately after the click is, by default, the selected image.

Example: In Microsoft Office Word 2010, the **Margins** button changes appearance when it is idle, hovered or clicked. Figure 108 shows the image variations of the detected object in the **Position** tab's **Object** area.

Figure 104: Object Image Variations



Section 3.6.2.1.4.9.3 describes how to select an image variation of the detected object.

Leo also lets you customize the detected object image by selecting it in the step's **Display** pane. Section 3.6.2.1.4.9.8.4 describes how to customize the object that you want Leo to detect.

When the Leo Wizard is run, Leo searches for an image on the user's screen that matches the wizard's object image in order to detect the object. The object detection match settings are described in Section 3.6.2.1.4.9.8.5.

3.6.2.1.4.9.3 Wait for Object to Appear

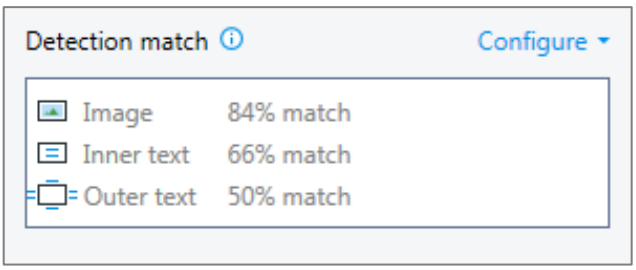
The **Wait for Object to Appear** checkbox causes Leo to wait for the step's detected object of the step window to appear (Figure 109). This setting lets you specify the maximum amount of time to wait. If the object does not appear by the time specified, Leo continues to the next core action. If the object appears before the maximum amount of time is up, Leo immediately continues to the next action without waiting the remaining amount of time.

Figure 105: Wait for Object to Appear

3.6.2.1.4.9.4 Detection Match

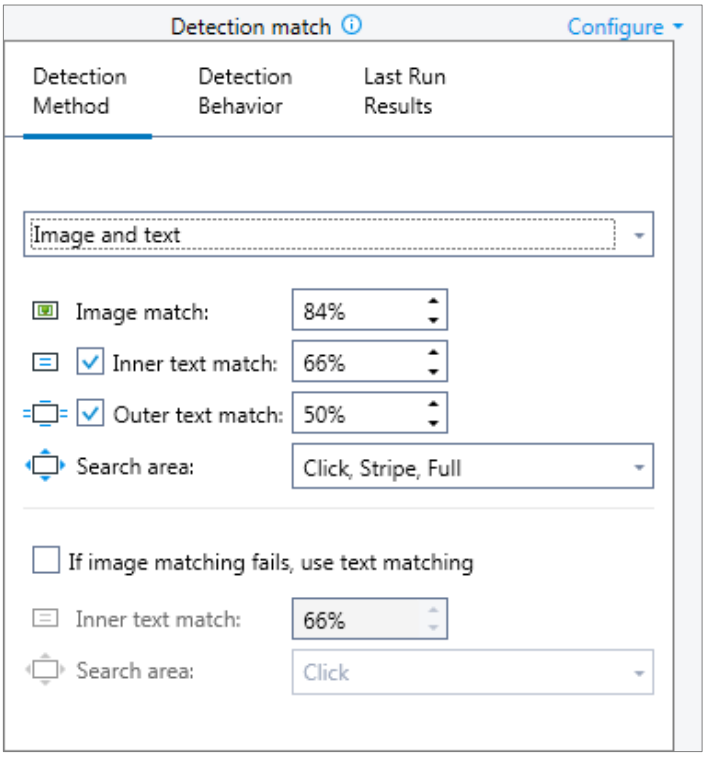
When a Leo Wizard is played and Leo looks for the click position, it analyzes the screen for an image and text that match the captured image and text selected in the **Position** tab's **Detected Object** area. The detection is based on the minimum match threshold for images and text, which appears in the **Detection Match** area (Figure 110).

Figure 106: Detection Match Area



The **Object Detection Match Configuration** menu (Figure 111) contains settings that allow you to adjust the flexibility of Leo's detection capabilities, that is, 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 Leo run and whether it succeeded or failed.

Figure 107: Detection Match Configuration Menu



The following sections describe the detection settings available in the **Configuration** menu:

- Detection Method: Allows you to determine the image/text detection method, the detection match threshold 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 Leo's screen blocking is activated.
- Last Run Results: Shows you the match percentages that Leo detected the last time the Leo Wizard or Sensor was run from Leo Studio. This is useful for optimization and debugging purposes.

3.6.2.1.4.9.4.1 Detection Method

The **Detection Method** tab (Figure 111) allows you to determine the image/text detection method, the detection match threshold and the size of the search area.

These settings in this tab are:

- **Method**: When Leo attempts to identify the image on the user's application, it attempts by default to identify the text both in and around the image. You can choose to detect the click position based on both image and text, or text only.
- **Match Thresholds**: The threshold by which Leo expects the detected object to match the recorded object. Match thresholds exist for the following detected objects:
  - Image Match: The visual image of the object and of any text in it. The default image threshold is 84%.

- Inner Text Match: OCR (image-to-text conversion) for text within the detected object. For example, a sentence on the screen or text inside a button (e.g. **Cancel**).
  The default inner text threshold is 60%~80%, depending on the text length.

If the clicked image does not contain text, Leo automatically clears the **Inner Text** checkbox.

- Outer Text Match: OCR for text surrounding the detected object. Outer text is used to confirm the location of the image when there are similar images nearby.
  The default outer text threshold is 50%.

The outer text threshold only affects a small handful of scenarios. In most cases, Leo will ignore it. If outer text might be dynamic, it is recommended to clear the **Outer Text** checkbox.

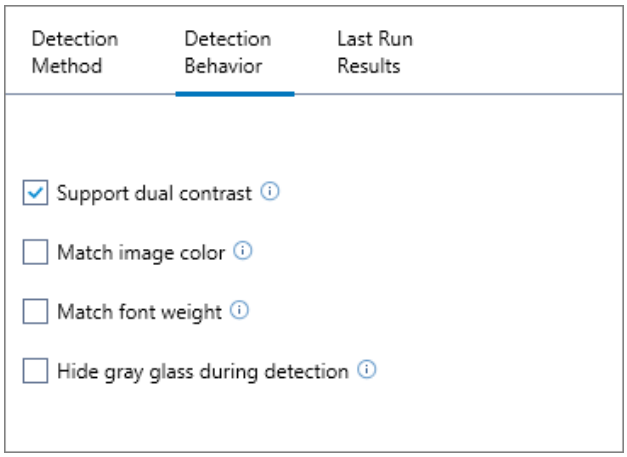
Decreasing the minimum match threshold under 70% is not recommended, and in some cases restricted by Leo. If the minimum threshold is less than 70%, Leo might click the wrong area of the screen, in which case you must increase the percentage. If you use 70% or above and Leo is still unable to detect the click position, it is recommended to add a fallback, as described in Section 3.3.2.3.2.

- **Search Area**: If the object location is expected to change, Leo can search the entire window to detect it. If the object location never or barely changes, limiting the search to the click area improves performance.

3.6.2.1.4.9.4.2 Detection Behavior

The **Detection Behavior** tab (Figure 112) allows you to include detection of color inversions, limit detection to a specific color and font weight, and optimize detection when Leo's screen blocking is activated.

Figure 108: Detection Behavior Tab

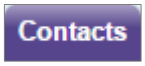


The sections of this tab are:

- **Support Dual Contrast**: Ensures detection if the object's colors are inverted. For example:
  - This **Contacts** tab, when idle (that is, not clicked), displays dark text on light background:

    Contacts

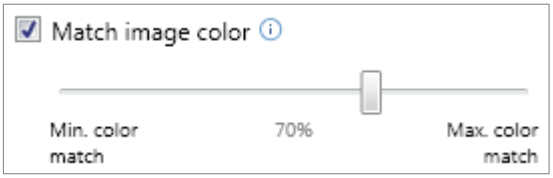
- The **Contacts** tab's colors get inverted when clicked, resulting in light text on dark background:

The **Support Dual Contrast** checkbox allows Leo to detect the object in either idle/clicked state.

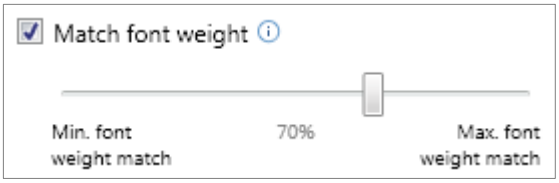
- **Match Image Color**: Ensures detection of the same image color as recorded, to ensure color-based detection. This setting allows you to prevent detection of objects with similar, but not identical colors, and to exclude minor color variations.
  This setting includes the ability to specify a color match percentage (Figure 113). The default color match percentage is 70%. Adjust this setting to increase or decrease the expected color match between the recorded object and the run result.

  Figure 109: Match Image Color Setting

- **Match Font Weight**: Ensures detection of the same font weight as recorded, to ensure font-weight-based detection. This setting allows you to prevent detection of objects whose font weight does not match the weight of the recorded font. This enables you, for example, to detect the object only when its text is in bold, but to ignore it if its text is not bold.
  This setting includes the ability to specify a font-weight match percentage. The default color match percentage is 70%. Adjust this setting to increase or decrease the expected color match between the recorded object and the run result.

  Figure 110: Match Font Weight Setting

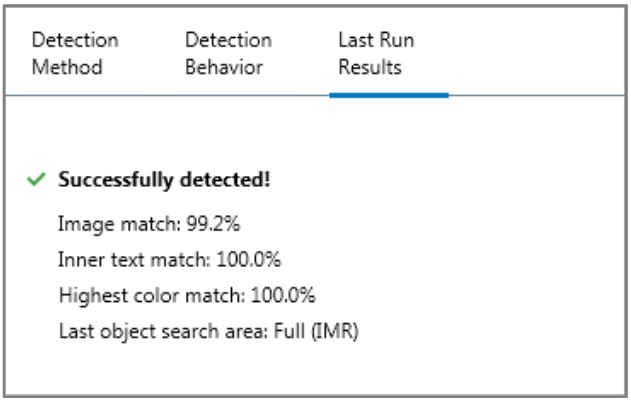
  

- **Hide Gray Glass**: This option is relevant for steps where Leo screen blocking is activated, for rare cases where the gray glass used for screen blocking might interfere with object detection. This setting enables you to temporarily disable the screen blocking only for the duration of object detection.

3.6.2.1.4.9.4.3 Last Run Results

The **Last Run Results** tab (Figure 115) displays the match percentages that Leo detected the last time the Leo Wizard was played from Leo Studio. This data is useful for optimization and debugging purposes, by helping you determine the desired match percentages and fix detection errors.

Figure 111: Last Run Results Tab



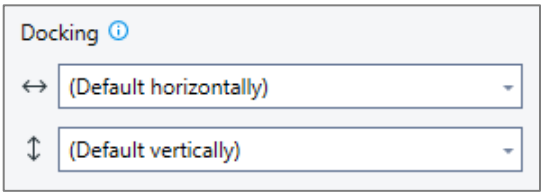
3.6.2.1.4.9.5 Docking

Object docking indicates to Leo which window border the object is docked, or pinned to. Use docking when performance speed or accuracy needs to be improved.

If the clicked object is not found in the expected area, Leo searches the entire screen.

Figure 116 shows the **Docking** area in the **Position** tab.

Figure 112: Object Docking



An object can be docked as follows:

- **Horizontally**: The object is located at a fixed distance from the left/right window borders (**Docked left/right**) or from the center (**Docked to center**).
- **Vertically**: The object is positioned at a fixed distance from one of the top/bottom window borders (**Docked to top/bottom**) or from the center (**Docked to center**).

By default, no specific position is selected.

3.6.2.1.4.9.6 Click Offset

In certain Leo Wizards, Leo needs to click detect non-unique objects (e.g. empty fields or dropdown arrows) or objects whose appearance changes (e.g. dropdown lists), recording a click on these types of areas does not ensure detection accuracy. To enable Leo to click them, you must edit the step using offset.

Offset moves the click or target position to a different location by a fixed number of pixels from the detected object. This enables Leo to click on non-unique or changing objects at a fixed distance from a detected unique, fixed object nearby (Figure 117).

Figure 113: Click Position with Click Offset

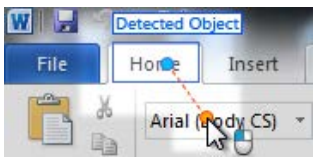


Figure 118 shows the **Position** tab's **Offset** area.

Figure 114: Click Offset Settings

By default, the click offset is set to 0 pixels.

Adjust the click offset by doing one of the following:

- From the **Display** pane: Place your cursor over the blue dot within the box surrounding the detected object. Press **SHIFT** and drag the blue dot to move the click/target position to a new location.
- The orange dot indicating the click offset position is to the new location (Figure 117).
- From the **Position** tab: In the **Offset** area, specify the number of pixels by which to move the click/target position vertically and/or horizontally (Figure 118).
- The orange dot indicating the click offset position is moved by the number of pixels you entered (Figure 117).

To move the offset position right/down, use positive numbers. To move it up/left, use negative numbers.

After using offset, verify that the highlight box is placed properly around the offset target. For information about customizing the highlight box, see Section 3.6.2.1.4.9.7.

3.6.2.1.4.9.7 Highlight Box

When users play a Leo 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 is determined according to the click position, and automatically adjusts itself to the object's borders.

You can customize a step's highlight box around the position that you want to indicate to the user. This is especially useful for steps in which the position is manually determined using offset (as described in Section 3.6.2.1.4.9.6). In such cases, the highlight box might need to be changed or expanded.

3.6.2.1.4.9.8 Core Action Position Procedures

3.6.2.1.4.9.8.1 Accessing the Core Action Position Tab

Access a core action's **Position** tab by doing the following:

1. Select the step containing the core action that you want to edit.
2. Do one of the following:
   - From the **Flow** pane, double-click the core action that you want to edit, and then click the **Position** tab.
   - From the **Flow** or **Display** pane, right-click the core action that you want to edit, and select **Position**.

The **Properties** pane appears, displaying the bubble's **Position** tab.

3.6.2.1.4.9.8.2 Editing the Core Action's Position Settings

Edit the core action's position settings by doing the following:

1. Access the core action's **Position** tab as described in Section 3.6.2.1.4.9.8.1.
2. From the **Position** dropdown list, select an option.

If you select **Fixed Position**, in the **Fixed Position** graphic click the arrow that represents the corner relative to which the click position will be calculated (Figure 119). The upper-left corner is selected by default.

Figure 115: Fixed Position Graphic

3.6.2.1.4.9.8.3 Selecting an Object Image Variation

Select a variation of the object image by doing the following:

3 Access the core action's **Position** tab as described in Section 3.6.2.1.4.9.8.1.
4 In the **Position** dropdown list, verify that **Detected Object** is selected.
5 In the **Object** area, select the checkbox of the image variation to detect (Figure 108).

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

3.6.2.1.4.9.8.4 Customizing the Detected Object

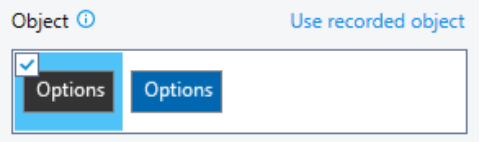
Customize the detected object by doing the following:

1 Select the step whose detected object you want to customize.
In the step's **Display** pane, the current detected object is highlighted by a blue bounding box and label, and contains a blue dot that represents the cursor click position during runtime in Do It mode (Figure 35).
2 Drag and resize the highlight box around the new object to detect, and move the blue dot within the box to reposition the cursor click.

During customization, a highlighted square marks the window area within which you can select a new anchor object.

The custom image of the detected object and any available variations appear in the **Object** area, along with the **Use Recorded Object** link (Figure 120).

Figure 116: Custom Object Image



To remove a customized object and restore the originally recorded object, click the **Use Recorded Object** link in the **Object** area.

3.6.2.1.4.9.8.5 Defining Image Detection Settings

Define the image detection settings by doing the following:

1 Access the core action's **Position** tab as described in Section 3.6.2.1.4.9.8.1.
2 In the **Position** dropdown list, verify that **Detected Object** is selected.
3 In the **Detection Match** area, click **Configure** (Figure 111).
4 From the **Method** dropdown menu, do one of the following:

- Select **Image and text** to detect the click position by identifying both the image and text, and then do the following:

i Set the image, inner text and outer text percentages.

If the image is small, it is recommended that you increase the image match percentage.

ii From the **Search area** dropdown list, choose which area to search:

- Click: The click area only
- Click, Stripe: The stripe if the click search fails

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: The entire window if both the click and stripe search fail.

iii To set a fallback for image matching failure, select the **If image matching fails** checkbox, set the fallback inner text percentage, and select the search area.

- Select **Text only** to detect the click position by identifying the inner text only, and set the inner text percentage and search area.
- The **Image** and **Outer text** settings are disabled.

3.6.2.1.4.9.8.6 Editing Object Docking Settings

Edit the object docking settings by doing the following:

1. Access the core action's **Position** tab as described in Section 3.6.2.1.4.9.8.1.
2. In the **Position** dropdown list, verify that **Detected Object** is selected.
3. In the recorded application window, move each of the window's borders back and forth, and check whether the distance between the clicked object and the border changes.
4. In the **Docking** area, from the **Horizontal** ↔ dropdown list, select a docking position as follows:
   - If the distance between the object and the left border does not change, select **Docked left**.
   - If the distance between the object and the right border does not change, select **Docked right**.
   - If the distance between the object and the center does not change, select **Docked to center**.
5. In the **Docking** area, from the **Vertical** ↕ dropdown list, select a docking position as follows:
   - If the distance between the object and the top border does not change, select **Docked to top**.
   - If the distance between the object and the bottom border does not change, select **Docked to bottom**.
   - If the distance between the object and the center does not change, select **Docked to center**.

3.6.2.1.4.9.8.7 Customizing the Highlight Box

Customize a step's highlight box by doing the following:

1. Access the core action's **Position** tab as described in Section 3.6.2.1.4.9.8.1.
2. In the **Highlight box** area, click **Customize**.
3. In the **Display** pane, the default highlight box appears with a **Highlight box** label around the detected object.

4 Drag and resize the box around the object that you want Leo to detect.

To view the current highlight box, whether customized or auto-detected by default, click **Show**.

To return to the default highlight box, click **Delete (Use Default)**.

3.6.2.1.4.10 Core Action Procedures

3.6.2.1.4.10.1 Accessing the Core Action Properties Tab

Access a core action's **Properties** tab by doing the following:

1 Select the step containing the core action that you want to edit.
2 From the **Flow** pane, do one of the following:
- Double-click the core action whose properties you want to access.
- Right-click the core action whose properties you want to access, and select **Properties**.

The **Properties** pane appears, displaying the core action's **Properties** tab.

3.6.2.1.4.10.2 Changing or Restoring the Step's Core Action

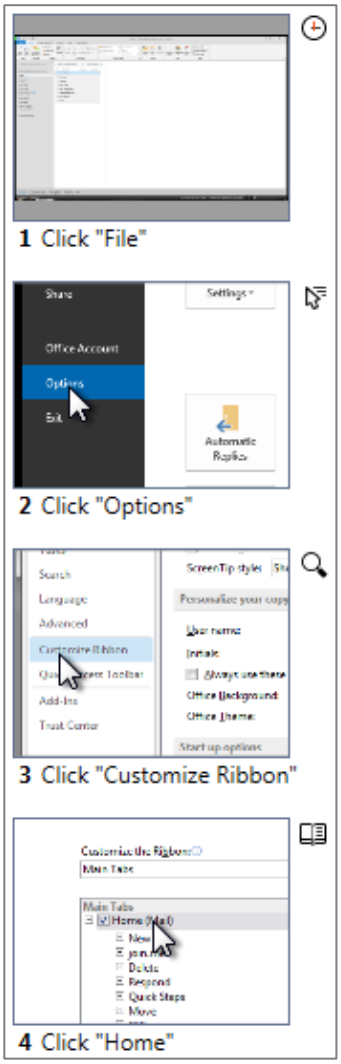
In a Leo step, any core action can be replaced with another core action, or restored if it was previously deleted.

Change or restore the step's core action by doing the following:

1 Select the step.
2 In the **Flow** pane, click **Insert**.
3 From the **Core Action** list, select a core action.

The step core action is updated with the selected action, and the action icon appears in the step thumbnail's upper-right corner.
Figure 121 shows step thumbnails with various core action icons.

Figure 117: Core Action Icons



3.6.2.1.4.10.3 Customizing the Guide Me Bubble Text

Customize a step's Guide Me bubble text by doing the following:

1. Select the step.
2. Access the core action Properties tab, as described in Section 3.6.2.1.4.10.1.
3. From the **Guide Me bubble text** dropdown list, select **Custom**.
4. In the **Custom** field, type the custom bubble text.
   The custom bubble text appears for this step when the wizard is played in Guide Me mode.

3.6.2.1.4.10.4 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 Leo.

Example: When you delete a rule in Microsoft Outlook, Outlook prompts you to confirm the deletion. If you record a Leo Wizard for deleting a rule, you might want to provide users with a choice and not let Leo perform the deletion for them. Instead of playing the step where Leo clicks **OK** to confirm the deletion, you can add a bubble instructing the user to confirm. In this scenario, delete the core action so that the bubble appears but Leo does not click **OK** for the user.

Delete a core action by doing the following:

1 Select the step and do one of the following:

- In the **Flow** pane, click the core action and press **DELETE**.
- In the **Flow** pane, right-click the core action and select **Delete**.
- From the **Display** pane, select the core action's detected object and press **DELETE**.

The core action is deleted from the step.

To restore a core action to the step, see Section 3.6.2.1.4.7.

3.6.2.1.4.11 Core Action Fallbacks

The core action fallback events are:

- Window not Found
- Object not Found

For a description of these fallback events, see Section 3.3.2.3.1.

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

3.6.2.1.5 Embedded Wizards

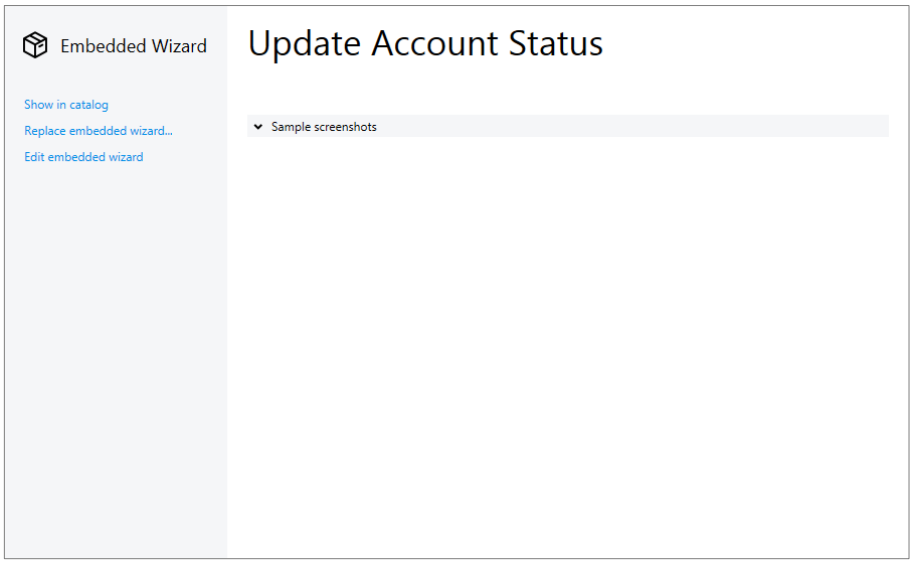
Embedded wizards provide you with a single source that is recorded once and then reused it in multiple locations, without having to edit it separately in each location.

Any wizard from any library can be used as a building block within another wizard, regardless of the application it was recorded on.

The embedded wizard can be made invisible to end users in Leo Player, by hiding it from the search results.

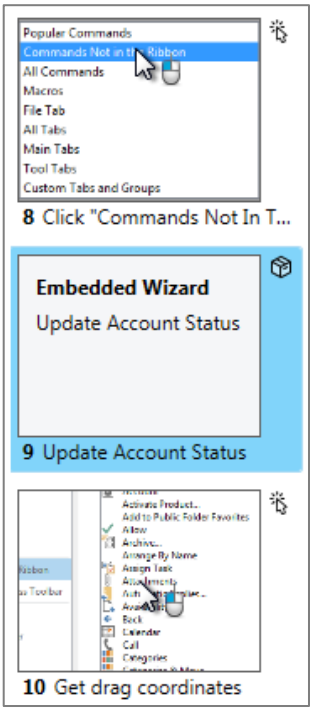
A Leo Wizard that is embedded into another wizard has advanced features that can be managed from its control pane and **Properties** tab (Figure 122).

Figure 118: Embedded Wizard Control Pane in the Display Pane



To view the embedded wizard's control pane, select that **Embedded Wizard** step in the Wizard Editor's **Navigation** pane (Figure 123).

Figure 119: Embedded Wizard Step Thumbnail



Embedded wizards are not available for sensors. For information about how to launch a wizard from a sensor, see Section 4.2.4.

3.6.2.1.5.1 Embedded Wizard Properties

The embedded wizard control pane and **Properties** tab contains the following:

- **Show in Catalog**: A link that, when clicked, selects the embedded wizard at its location in the Leo Studio catalog.
- **Replace Embedded Wizard**: A link that, when clicked, opens the **Catalog** dialog box (see Figure 27), enabling you to replace the embedded wizard you selected with another embedded wizard. For more information about embedding Leo Wizards, see Section 3.6.1.4.
- **Edit Embedded Wizard**: A link that, when clicked, opens the embedded wizard for editing in a new Wizard Editor. For more information about editing the flow of a Leo Wizard, see Section 3.6.
- **Sample Screenshots**: A dropdown list that displays preview screenshots of the embedded wizard's steps and step names or bubble text.

3.6.2.1.5.2 Embedded Wizard Fallbacks

The embedded wizard fallbacks are:

- **Wizard Not Found**: Leo attempts to detect the embedded wizard before it activates it. If Leo fails to detect the embedded wizard, it performs a fallback. This might occur because the user does not have permissions to access the embedded wizard, or because the embedded wizard was not imported, and so on. The default fallback is to continue the flow of the containing wizard.
- **Wizard Ended Unsuccessfully**: Leo attempts to continue the flow defined by the content author. If the content author defines the flow to end unsuccessfully, or if Leo fails to continue the flow, it performs a fallback. The default fallback is to continue the flow of the containing wizard.

For more information about embedded wizards, see Sections 3.6.1.4-3.6.1.6.

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

3.6.2.1.6 Step End

Step End is a mandatory action that is added by default. It is triggered at the end 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 **End Action** dropdown list.

For information about Step End properties, see Section 3.6.2.1.6.1.

3.6.2.1.6.1 Step End Properties

The **Step End** action enables you to choose one of the following closing actions for the selected step:

- **Go to Next Step**: When the step is completed, Leo goes to the next step in the wizard flow. This is the default action.
- **Go to Step**: When the step is completed, Leo goes to any step in the flow that you specify.
- **Advanced Commands**: When the step is completed, Leo triggers a predefined set of advanced commands. When the advanced commands are completed, Leo continues the wizard flow in the order determined by the content author.
- **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, Leo triggers a predefined keyboard key combination (e.g. **ALT+P** to trigger the print command).
    - Keyboard Language: Determines whether Leo will accept input in the language recorded or in any language the keyboard is set to.

Not applicable to mouse steps.

- **End Wizard Successfully**: When the step is completed, Leo exits the wizard and reports the ending as successful.
- **End Wizard Unsuccessfully**: When the step is completed, Leo exits the wizard and reports the ending as unsuccessful.

Set the step's end action by doing the following:

1. Select the step.
2. In the **Flow** pane, click the **Flow** tab (see Figure 48).
3. From the **End Action** dropdown list, select an option.

3.6.2.1.6.2 Step End Fallbacks

Step End actions do not have fallbacks.

3.6.2.2 Window

3.6.2.2.1 Window Properties

Each set of window data is made up of window properties (Figure 124). These properties can be viewed and accessed for editing from the following locations:

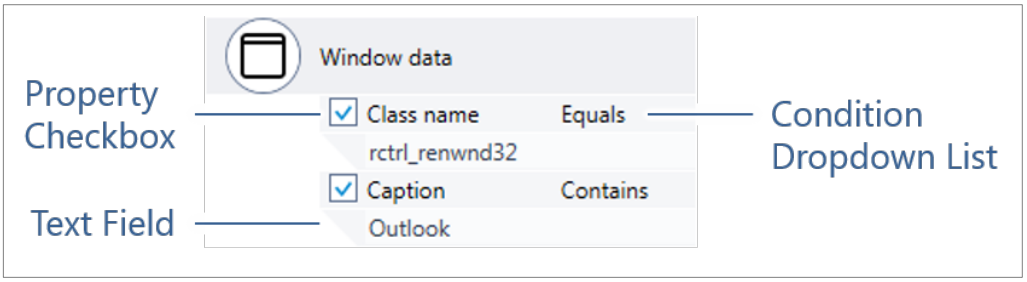
- The **Flow** pane's **Window** tab (see Section 3.3.1.2.)
- The selected window data's **Properties** pane

The window properties are:

- **Property Checkbox**: Determines whether this property will be included or ignored during window detection.

- **Condition Dropdown List**: Contains conditions that determine which part of the text field will be included or ignored during window detection. The **Condition** dropdown list contains the following conditions:
  - Equals: This property in the detected window is expected to contain this exact text string only. This is the default value.
  - Contains: This property in the detected window is expected to contain this exact text string, either on its own or wrapped between other text, somewhere in this property's beginning, middle or end. This condition is automatically selected when you make a change to the property's text field.
  - Begins With: This property in the detected window is expected to begin with this exact text string, either on its own or followed by other text.
  - Ends With: This property in the detected window is expected to begin with this exact text string, either on its own or preceded by other text.
  - Use Wildcards: In this property in the detected window, you can replace any character or string of characters with a wildcard, where:
    - An asterisk (*) represents zero or more characters.
    - A question mark (?) represents any single character.
- **Text Field**: Contains the text string that Leo will look for in order to detect the window property, qualified by one of the conditions from the **Condition** dropdown list.

Figure 120: Window Property Components



For information about how to edit window detection properties, see Section 3.6.2.2.2.1.

3.6.2.2.2 Window Procedures

3.6.2.2.2.1 Editing the Detected Window Data

Edit the window data in a step by doing the following:

1. Select the step whose window data you want to edit.
2. In the **Flow** pane, click the **Window** tab (Figure 31).
3. Verify that the checkboxes of properties used to detect the window are selected.
4. From the **Condition** dropdown list, select a condition.

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.

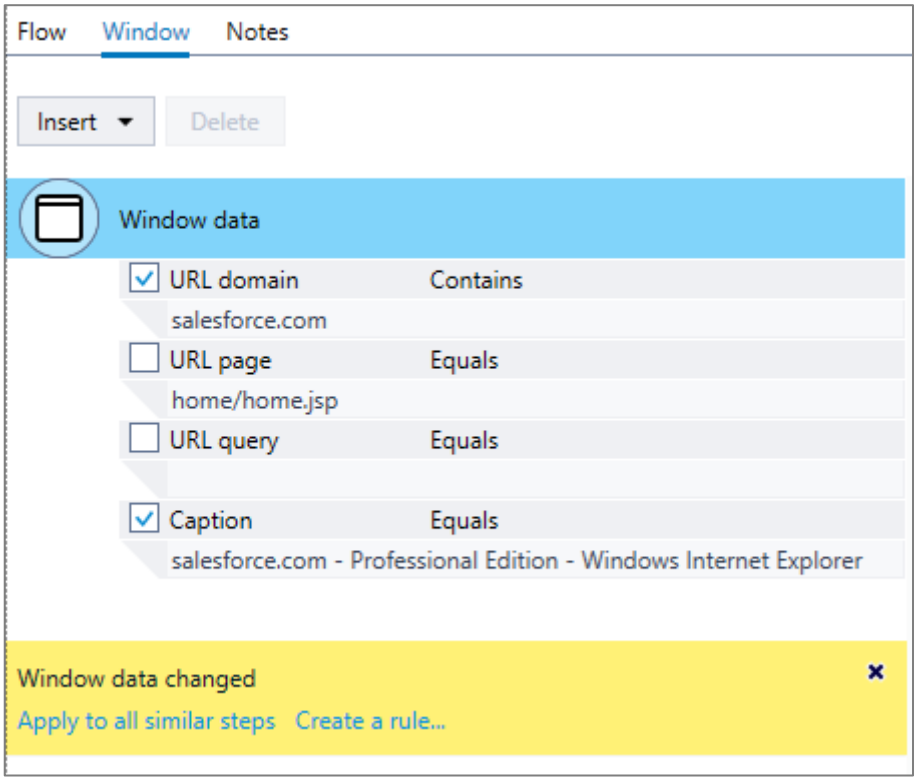
5. In the property's text field, make sure that the text entered is as generic as possible and does not contain user-specific or environment-specific text that might fail the detection under different circumstances. (Figure 32).

The text field is not case sensitive.

The data is changed, and the following occurs:

- If you removed any text from the text field, the **Condition** dropdown list changes from **Equals** to **Contains**.
- If this property is being changed for the first time, the **Suggest Rule** link appears.
- If the window in this step is shared by other steps in the current wizard, the **Apply to All Similar Steps** link appears (Figure 125).

Figure 121: Apply to All Similar Steps Link



6 If the **Suggest Rule** link appears, and you want to create a rule that applies the property to all Leo Wizards that share similar window detection values, continue with Section 3.6.2.2.2.4.

7 If the **Apply to All Similar Steps** link appears, and you want to apply the property to all steps in the current wizard that share this window, click **Apply to All Similar Steps**.

For information about the window data components, see Section 3.1.1.3.

3.6.2.2.2.2 Adding a Set of Window Data

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

Add window data to a step by doing the following:

1 Select the step in which you want to add window data.

2 In the **Flow** pane, click the **Window** tab (Figure 31).

3 In the **Window** tab, click **Insert > Window Properties**.

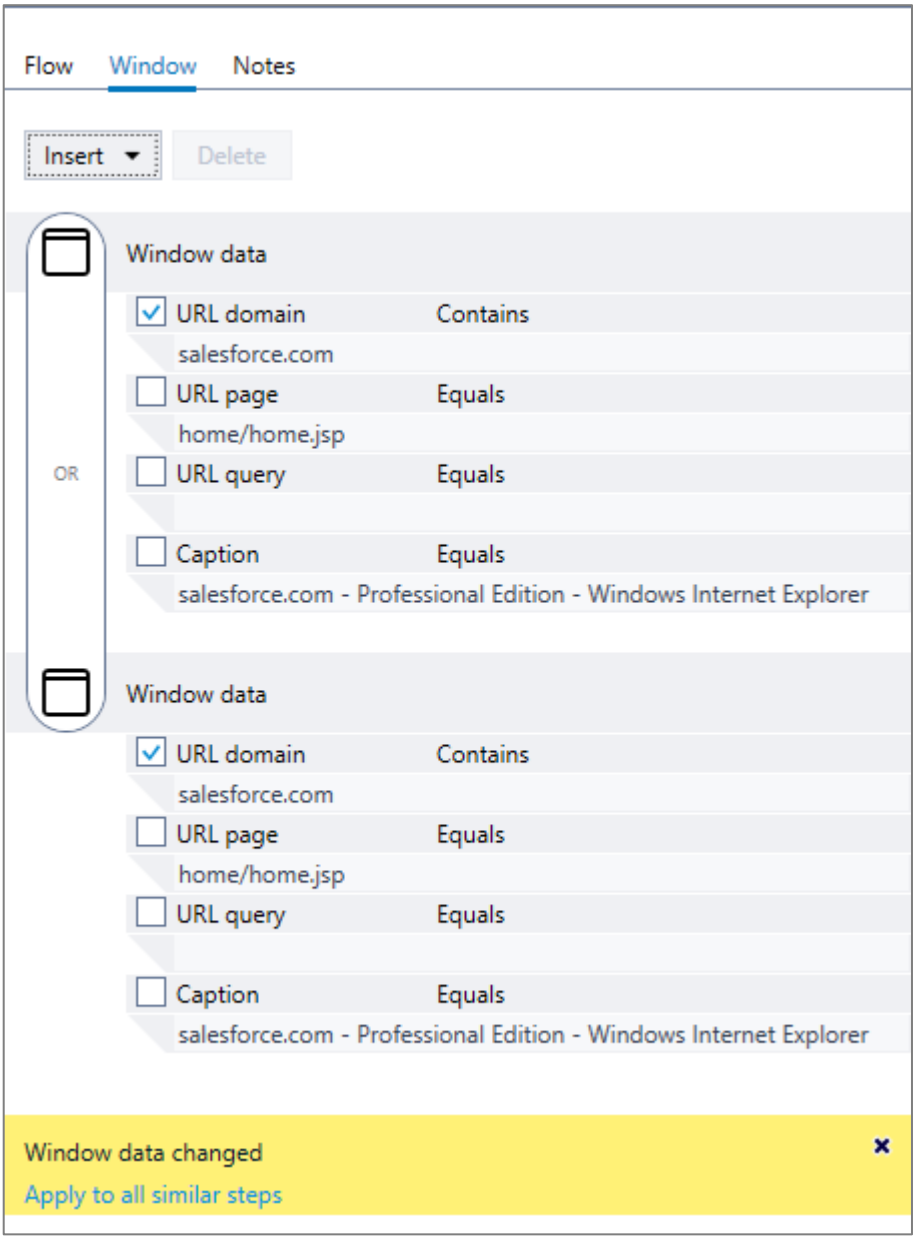
A new window detection area appears, containing one of the following:

- Class name and window caption name fields for desktop applications
- URL and window caption name fields for Web browser windows in Web applications

4 For each set of window data, add/remove the relevant details.
The window data is updated, and the following occurs (Figure 126):

- If this window data is being updated for the first time, Leo prompts you to add a rule from the **Suggest Rule** link (see Section 3.6.2.2.2.3).
- If the window data in this step is shared by other steps in the current wizard, Leo prompts you to apply the change to all similar steps (Figure 125).

Figure 122: Window with Multiple Data Sets for Different Window Captions



To remove window data that was added to a step, click that data and either press **DELETE** or click the **Delete** button that appears.

3.6.2.2.2.3 Adding Visual Window Detection

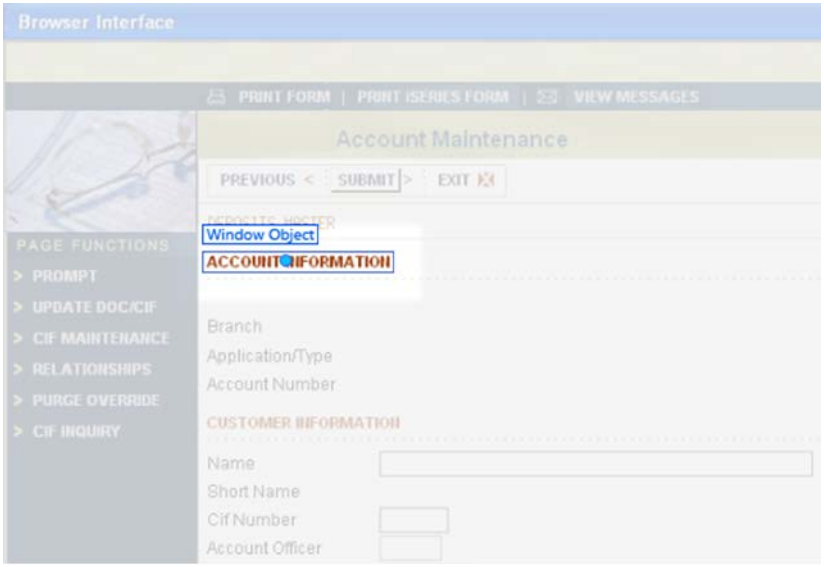
In some applications, multiple windows share the same window class/caption. In such cases, Leo cannot rely on window properties alone to detect the correct window. For such cases, you can specify a visual object in the window that lets Leo identify the correct window.

Add a visual window object to the **Window** tab by doing the following:

1. Select the step in which you want to add visual window detection.
2. In the **Flow** pane, click the **Window** tab (Figure 31).
3. From the **Window** tab, click **Insert > Window Visual Object**.
   The mouse cursor turns into a **plus** (+).
4. In the **Display** pane, click the object to detect as a window object.

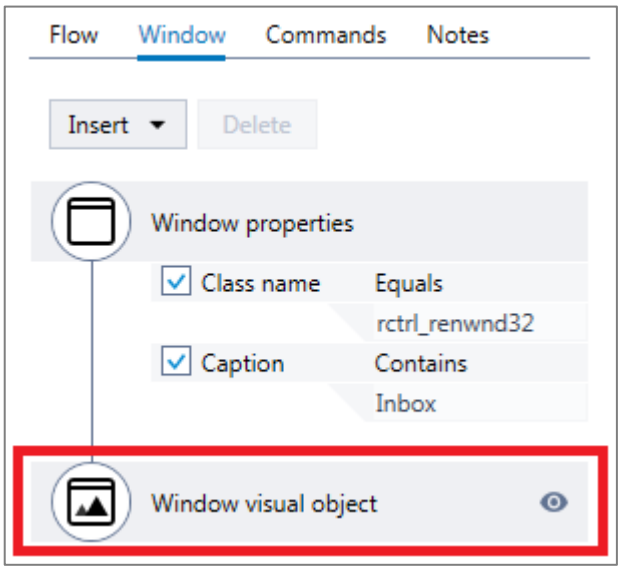
The **Window Object** box appears around the object (Figure 127).

Figure 123: Window Object in the Application



5 Drag the **Window Object** box or its borders to reposition or resize it.
The object is added to the step's window detection properties (Figure 128).

Figure 124: Window Object in the Window Tab



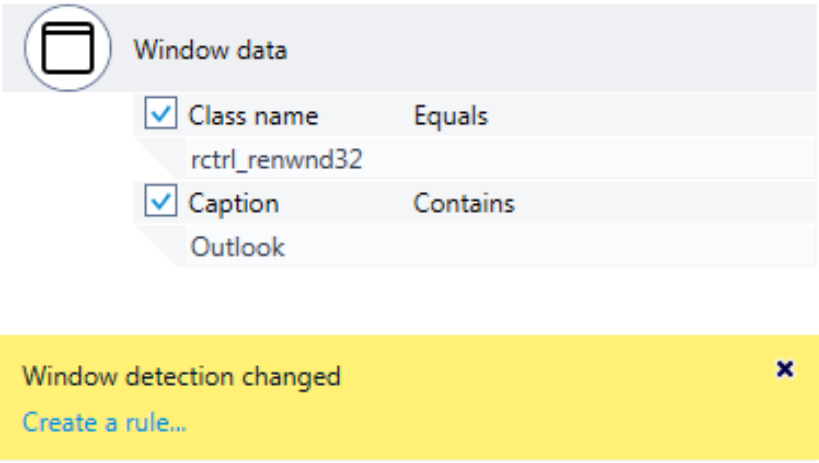
You can add multiple window objects by repeating this procedure. Each additional window object is a fallback in case the preceding window object is not detected.

3.6.2.2.2.4 Adding Window Detection Rules

Window detection rules enable you to automatically apply the same conditions and actions to all Leo Wizards that share similar window detection values.

The option to add a rule becomes available the first time you manually change the window detection value of a step (Figure 129).

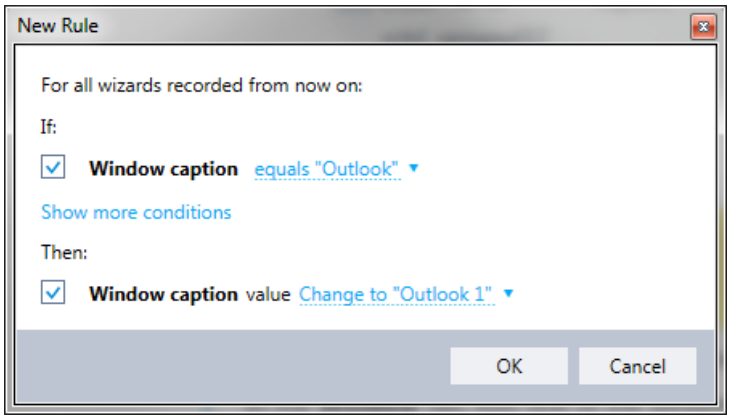
Figure 125: Rule Suggestion in the Page Detection Menu



Add window detection rules by doing the following:

1 In the **Window** tab, edit one of the window detection properties.
2 Click the **Create a Rule** link that appears (see Figure 126).
The **New Rule** dialog box appears (Figure 130).

Figure 126: New Rule Dialog Box



3 In the **New Rule** dialog box, select or clear the **If** checkboxes to add or remove rule conditions.
Example: If the default window caption is **Inbox – Mailbox – Jane Smith – Microsoft Outlook**, select the **ends with Microsoft Outlook** operative to make the condition more generic (that is, applicable to a broader range of scenarios).
4 Click the rule condition's operator dropdown list to select an operator, which qualifies the rule condition.
5 Click the **Show more conditions** link to add rule conditions.
The available rule conditions appear.
6 Select or clear the **Then** checkboxes to add or remove rule actions.
7 Click a rule action's operator dropdown list to select an operator, which qualifies the rule action.
8 Click **OK** to apply the rule.
The rule is added to the Leo Studio Rule Manager and is applied to every new Leo Wizard that is recorded.

The rule is not applied to the Leo Wizard in which you add the rule.

For more information about viewing and managing rules in the Leo Studio Rule Manager, see Section 2.6.3.

3.6.2.2.2.5 Bring Window to Front

The Bring Window to Front checkbox enables you to manually determine the activation of the window on which Leo performs the action. The feature options are:

- **Automatic**: Leo automatically determines whether it should bring to front, that is, activate, a window in order to perform the action. If the window is already activated, Leo does not attempt to activate it.
- **Always**: Leo always attempts to activate the window. This is useful for preventing cases in which the window might be unexpectedly deactivated by another window that is brought to front.
- **Never**: Leo never attempts to activate the window, and performs the action on the window only if it is already activated.
- Not applicable to Leo Sensors.

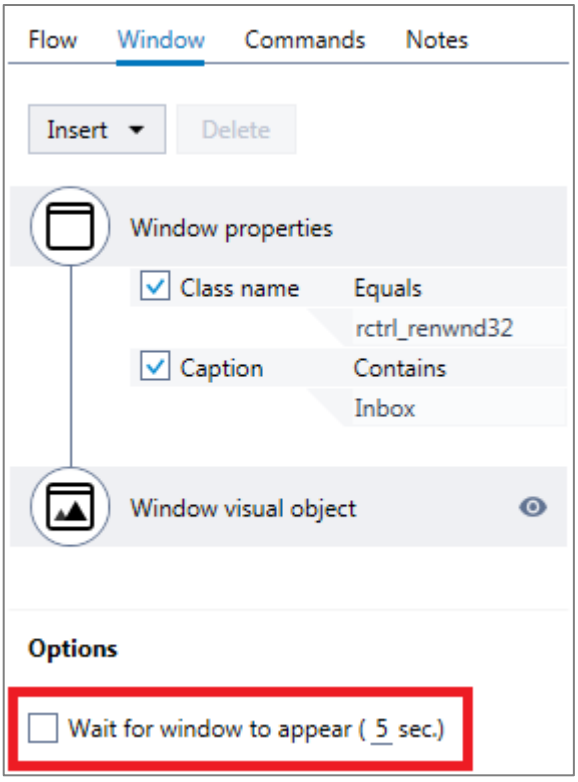
Choose whether to bring the step window to front by doing the following:

1 Select the step.
2 In the **Flow** pane, click the **Window** tab (Figure 31).
3 From the **Bring window to front** dropdown list, select an option.

3.6.2.2.2.6 Waiting for Window to Appear

The **Wait for Window to Appear** checkbox (Figure 131) ensures that Leo waits for the step's detected window to appear, with the ability to specify the maximum amount of time to wait. If the window does not appear by the time specified, Leo continues to the next action in the flow.

Figure 127: Wait for Window to Appear Checkbox



3.6.2.2.2.7 Waiting 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 to automatically wait for the Web page to download before starting the step.

The Web page download options are:

- **Automatically**: Leo waits while the page is loading. As soon as the page is downloaded, Leo immediately starts the step. This is the default setting.
- **Always**: Leo 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**: Leo never waits for the page to download and immediately attempts to start the step.

Not applicable to Leo Sensors.

Set the Web page download options of a step by doing the following:

1 Select the step.
2 In the **Flow** pane, click the **Window** tab (Figure 31).
3 From the **Wait for web page to download** dropdown list, select an option.

3.6.2.2.2.8 Enabling or Disabling Automatic Scrolling

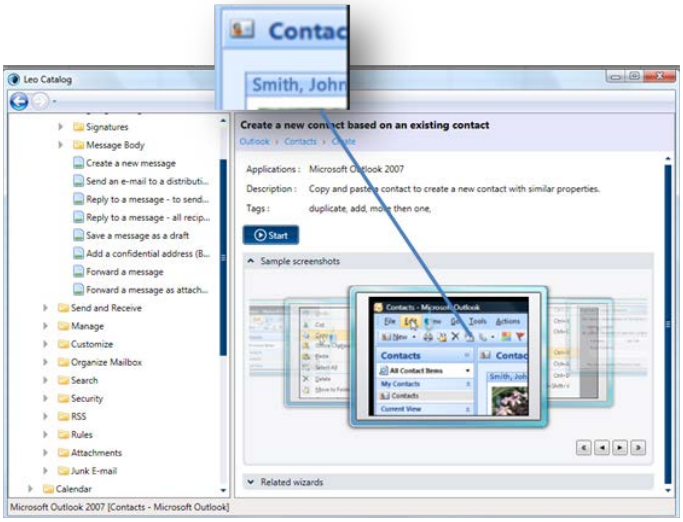
Choose whether to enable automatic scrolling for a step by doing the following:

1 Select the step.
2 In the **Flow** pane, click the **Window** tab (Figure 31).
3 From the **Vertical scrollbar** dropdown list, select an option.

3.6.2.3 Blurring Sensitive Data

The steps in the Leo Wizards you record can be viewed by other Leo users in the Leo Catalog or Studio. Steps recorded on your own applications might sometimes contain sensitive information, such as contact names and emails, which you might need to protect (Figure 132).

Figure 128: Contact Name Visible in Leo

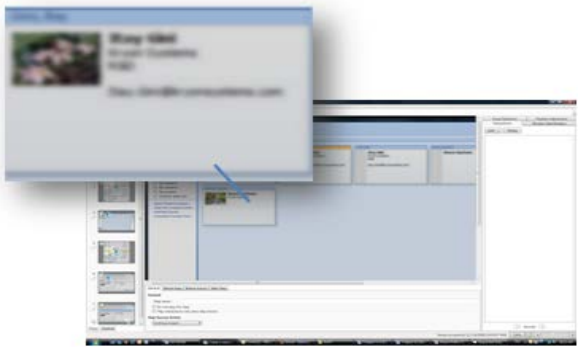


You can blur any area in the display screen that contains sensitive information.

Do not blur objects that you expect Leo to detect during runtime.

Figure 133 shows a blurred area in the **Display** pane.

Figure 129: Blurred Step



Blurring can be undone at any time until you save the Leo Wizard. When you save the Leo Wizard, the blurring becomes permanent. For more information about saving Leo Wizards, see Section 3.7.3.

Blur an area in a step by doing the following:

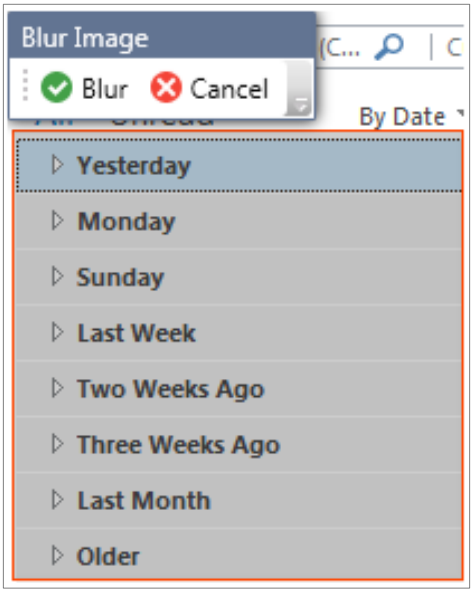
1 In the **Navigation** pane, select the step containing the area that you want to blur.

Do not blur images that you want Leo to detect when the Leo Wizard is played, or areas to which you want to anchor bubbles.

2 Do one of the following:

- On the Wizard Editor menu bar, select **Edit > Blur Image**.
- On the toolbar, click **Blur Image**.
- The blur bubble and **Blur Image** toolbar appear in the **Display** pane (Figure 134).

Figure 130: Blur Bubble and Toolbar



3 Drag and resize the bubble over the area that you want to blur.

4 On the **Blur Image** toolbar, click **Blur** to set the blur bubble. To cancel the blurring, click **Cancel**.

3.6.3 Warnings and Errors

The Notifications mechanism alerts you about issues and errors that might prevent Leo from functioning properly.

Notifications are provided for both a single step and the entire wizard.

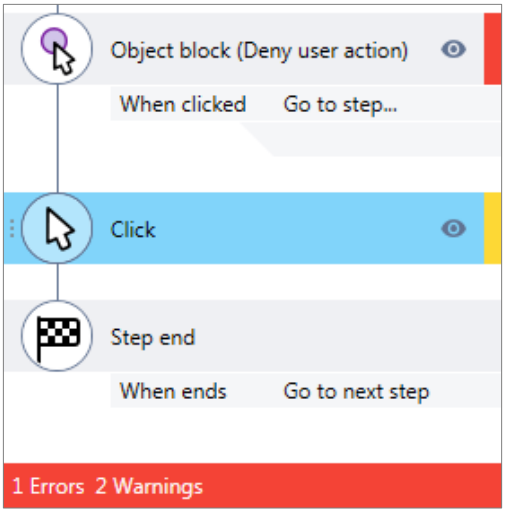
The notification types are:

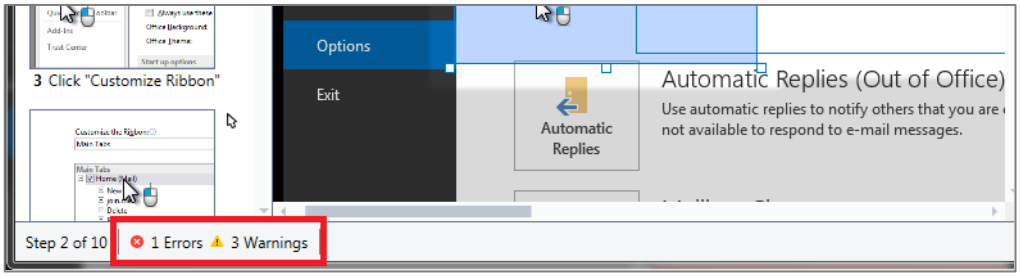
- **Wizard/Sensor Errors**: Notifications of critical errors that prevent Leo from functioning. These errors must be fixed before saving or closing the wizard or sensor.
  Wizard/sensor error notifications are red.
- **Algorithm Warnings**: A mechanism that examines the detected objects used in a step, and notifies of visual detection issues that need to be corrected to ensure accurate Leo detection. Fixing these issues is not mandatory but strongly recommended.
  Algorithm warnings are yellow.

The **Notifications** pane lists the following details:

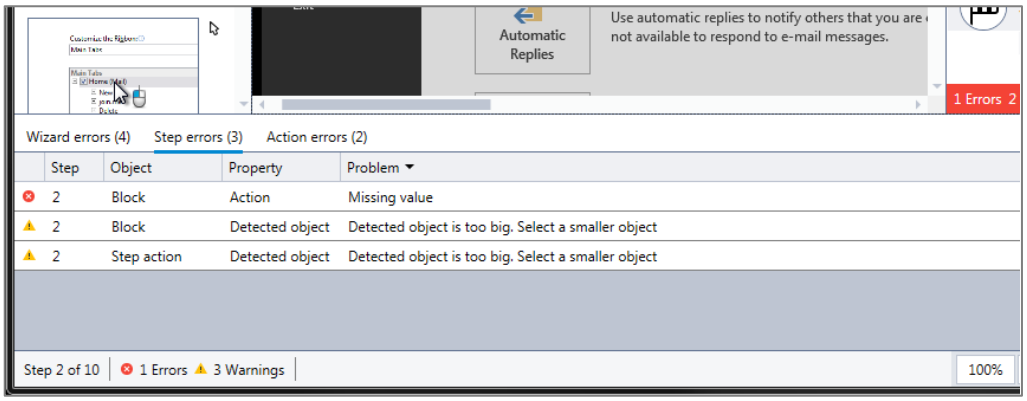
- Number of the step in which the issue occurred
- The wizard or sensor element in which the issue occurred
- A description of the issue and, in some cases, a suggestion

Errors and warnings appear in the following locations in the **Wizard Editor** window:

- In the **Flow** tab, both in the step action and at the bottom of the tab (Figure 135).
  Figure 131: Flow Tab Notification
  

- In the **Wizard Editor** window's status bar (Figure 136).
  Figure 132: Wizard Editor Window Notification
  

- In the **Notifications** pane, which appears after clicking the notification on either the **Flow** tab or the status bar.
  Figure 133: Notifications Pane in the Wizard Editor



3.6.3.1 Viewing Step Warnings and Errors

View all errors and warnings in a step by doing the following:

1 Do one of the following:

- From the **Flow** tab, click the notification at the bottom of the tab (Figure 135).
- At the bottom of the **Wizard Editor** window:

  i Click the **Errors and Warnings** notification (Figure 136).

  ii From the **Notifications** pane that appears, select the **Step Errors** tab.

The **Notifications** pane appears at the bottom of the **Wizard Editor** window with the **Step Errors** tab selected, listing all errors and warnings for the selected step (Figure 137).

2 Review the issues listed in the **Problem** column. Double-click an issue to access the step on which it appears, and follow any suggestions for correction.
The fixed issues are removed from the **Notifications** pane.

3.6.3.2 Viewing Wizard Warnings and Errors

View all errors and warnings in a wizard by doing the following:

1 Do one of the following:

- From the **Flow** tab:

  i Click the notification at the bottom of the tab (Figure 135).

  ii From the **Notifications** pane that appears, select the **Wizard Errors** tab.

- At the bottom of the **Wizard Editor** window, click the **Errors and Warnings** notification (Figure 136).

The **Notifications** pane appears at the bottom of the **Wizard Editor** window with the **Wizard Errors** tab selected, listing all errors and warnings for the wizard (Figure 137).

2 Review the issues listed in the **Problem** column. Double-click an issue to access the step on which it appears, and follow any suggestions for correction.
The fixed issues are removed from the **Notifications** pane.

3.6.4 Setting the Leo Studio Editor Options

You can modify the default Editor options in order to set the default bubble font, show/hide settings, wizard language, and more.

The Leo Studio Editor options are:

- Recording Settings
- Wizard Language
- Default Bubble Settings

Set the Leo Studio Editor options by doing the following:

1. On the Wizard Editor menu bar, select **Tools > Options**.
2. Select the **Playback** or **Editor** tab, and set the relevant options.
3. Click **OK**.

3.7 Running and Testing Wizards

After you record and edit a Leo Wizard, you must run the Leo Wizard to test it before it can be made available to users.

It is recommended that you test wizards from both Leo Player and Leo Studio, in order to test functionalities that do not exist in both tools. For example, Leo Player can prompt the user to choose how to launch the application before playing a wizard, while Leo Studio does not.

You can set the Leo Studio playback options to define how a Leo Wizard is played in Leo Studio, as described in Section 3.7.4.

Testing a Leo Wizard consists of the following procedures:

1. Running Leo Wizards from Leo Studio: After it is recorded and edited, each Leo Wizard must be played in Leo Studio to check the flow for errors.
2. Testing Leo Wizards in Expected Environments: After it is played from Leo Studio, each Leo Wizard must be tested in conditions that are different from those it was recorded in, such as another resolution and operating system.

For Leo Wizard playback troubleshooting, see Section 3.8.4.

3.7.1 Running Leo Wizards from Leo Studio

After it is recorded and edited, each Leo Wizard must be run in Leo Studio to check the flow for errors.

3.7.1.1 Leo Run Modes

Leo Wizards can be run in one of two modes:

- **Do It**: Leo performs the actions for the user by moving the user's mouse and navigating the target application to complete the task.
- **Guide Me**: Leo 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.

It is recommended that you do not run Leo Wizards in Turbo speed only. In Turbo speed, the mouse path and clicks are not displayed during runtime, which might prevent you from verifying the accuracy of the flow. For information about the cursor speed and other playback settings, see Section 3.7.4.

It is recommended that you test Leo Wizards in both Guide Me and Do It mode. It is highly recommended that you **always** test Leo Wizards in Guide Me mode.

3.7.1.2 Running a Leo Wizard from the Beginning

Play a Leo Wizard from the beginning by doing one of the following:

- In the Wizard Editor toolbar, click either **Do It** or **Guide Me**.
- On the Wizard Editor menu bar, select either **Wizard > Do It** or **Wizard > Guide Me**.
- Press **F5** for Do It mode or **F6** for Guide Me mode.
- The following occurs:

- The Leo Player icon appears on the taskbar.
- The "Wizard is running" bar appears at the top of the Wizard Editor window.
- Leo plays the Leo Wizard on the relevant application.

If two windows of the relevant application are open and neither window is currently active, Leo prompts you to select the window in which to play the Leo Wizard. If one of the application windows is active, Leo will play the Leo Wizard on the active window.

Example: If you are playing a Leo Wizard that requires an open mail message, and there are two open messages on your desktop, Leo prompts you to select the message on which to play the Leo Wizard.

If an open dialog box prevents Leo from playing the Leo Wizard on the application, Leo prompts you to close the dialog box and play the Leo Wizard.

3 Follow the instructions in any bubbles that appear when the Leo Wizard is played.

3.7.1.3 Running a Leo Wizard from a Specific Step

Play a Leo Wizard from a specific step by doing one of the following:

- In the Wizard Editor toolbar, click either **Do it from this step** or **Guide me from this step**.
- In the **Navigation** pane, right-click the step from which you want to start playing the Leo Wizard, and select either **Do it from this step** or **Guide me from this step**.
  - The following occurs:
  - The Leo Player icon appears on the taskbar.
  - The "Wizard is running" bar appears at the top of the Wizard Editor window.
  - Leo plays the Leo Wizard on the relevant application.

If two windows of the relevant application are open and neither window is currently active, Leo prompts you to select the window in which to play the Leo Wizard. If one of the application windows is active, Leo will play the Leo Wizard on the active window.

Example: If you are playing a Leo Wizard that requires an open mail message, and there are two open messages on your desktop, Leo prompts you to select the message on which to play the Leo Wizard.

If an open dialog box prevents Leo from playing the Leo Wizard on the application, Leo prompts you to close the dialog box and play the wizard.

4 Follow the instructions in any bubbles that appear when the Leo Wizard is played.

3.7.1.4 Pausing a Leo Wizard

Pause a Leo Wizard during runtime by doing one of the following:

- If a bubble is displayed, click the **Minimize** button in its upper-right corner. A floating **Pause** button appears on the screen (Figure 138), and the wizard is paused.

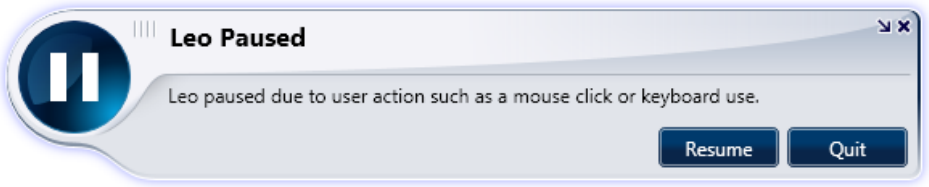
Figure 134: Floating **Pause** Button

When you are ready, click the floating **Pause** button to continue playing the Leo Wizard.

- Click the **Leo Player** icon from your taskbar.
- Click anywhere on the screen or press any keyboard key (in Guide Me mode, you might need to do this more than once).
- If a bubble is displayed, click the X in its upper-right corner.

The **Pause** bar appears on the screen (Figure 139), and the Leo Wizard is paused.

Figure 135: Paused Wizard



When you are ready, click **Resume** to continue playing the Leo Wizard, or click **Quit** to abort.

To minimize the **Pause** bar while it is paused, click **Minimize**. The **Pause** bar is minimized to a floating **Pause** button on your taskbar (Figure 138).

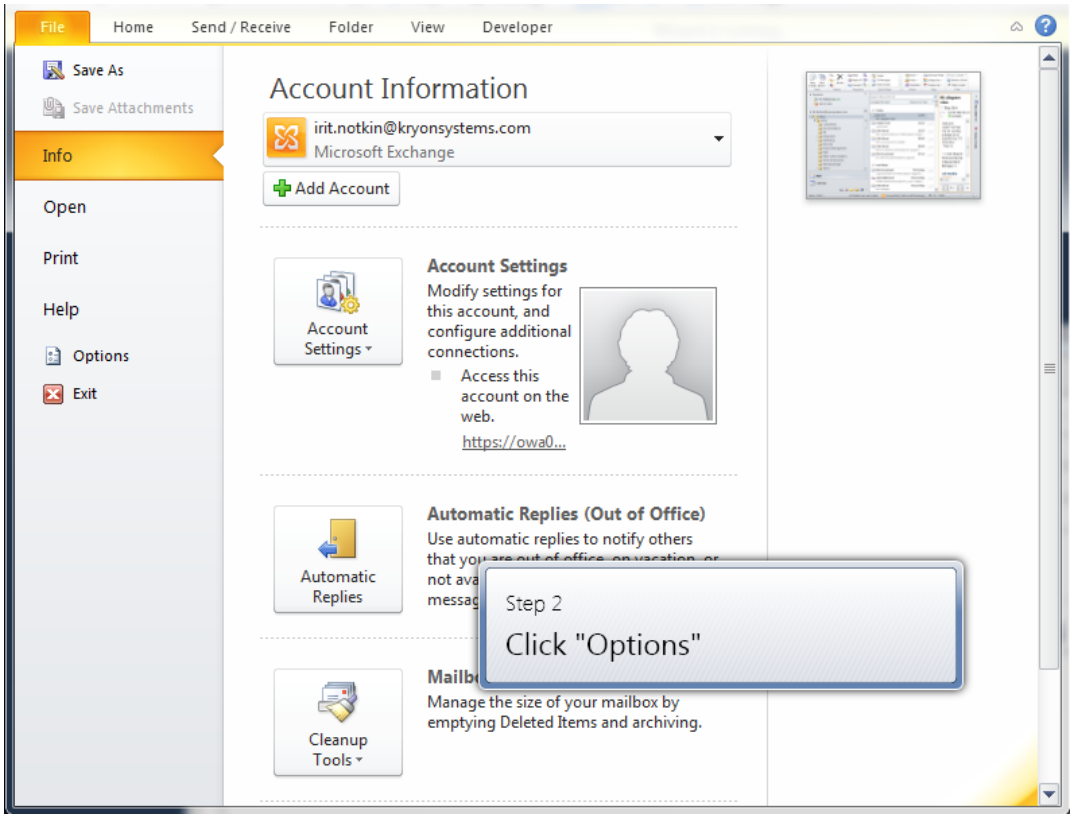
For information about troubleshooting Leo Wizard failures, see Section 3.8.4.

3.7.1.5 Debugging a Wizard

Debug mode is useful for testing and optimizing a wizard. This mode animates the wizard flow step by step, while the wizard is running, to highlight the following:

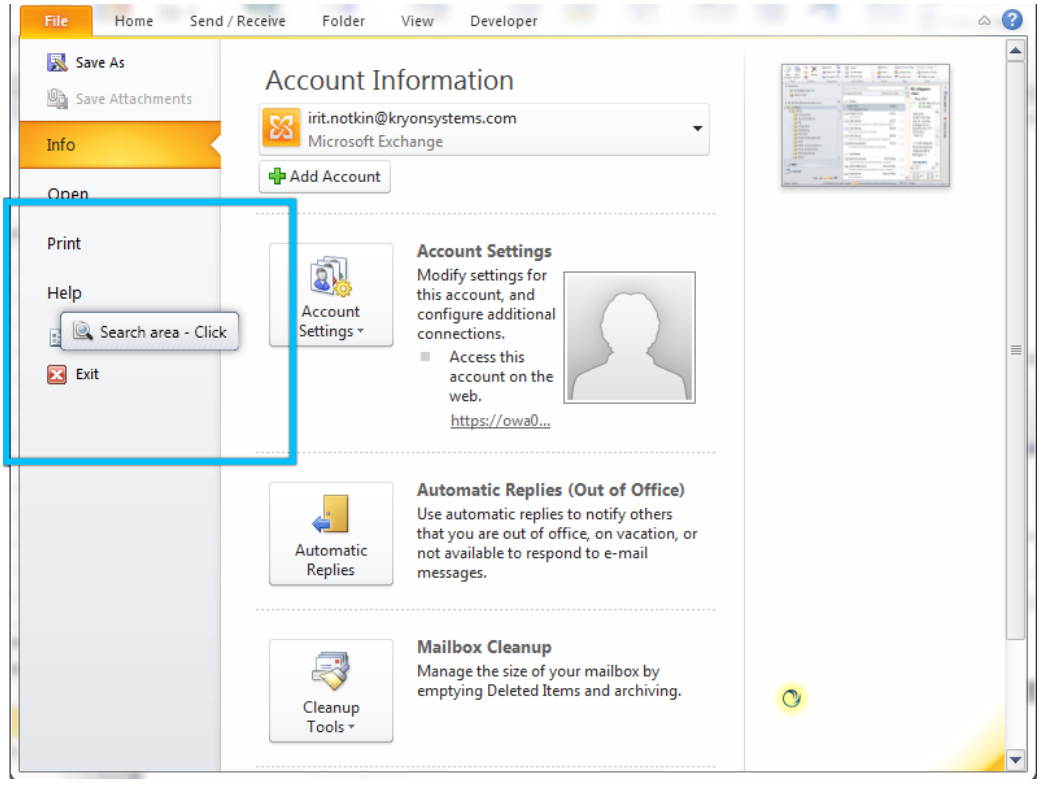
- Which steps are being played: Step Name (if applicable) and number (Figure 140)

Figure 136: Step Number and Name



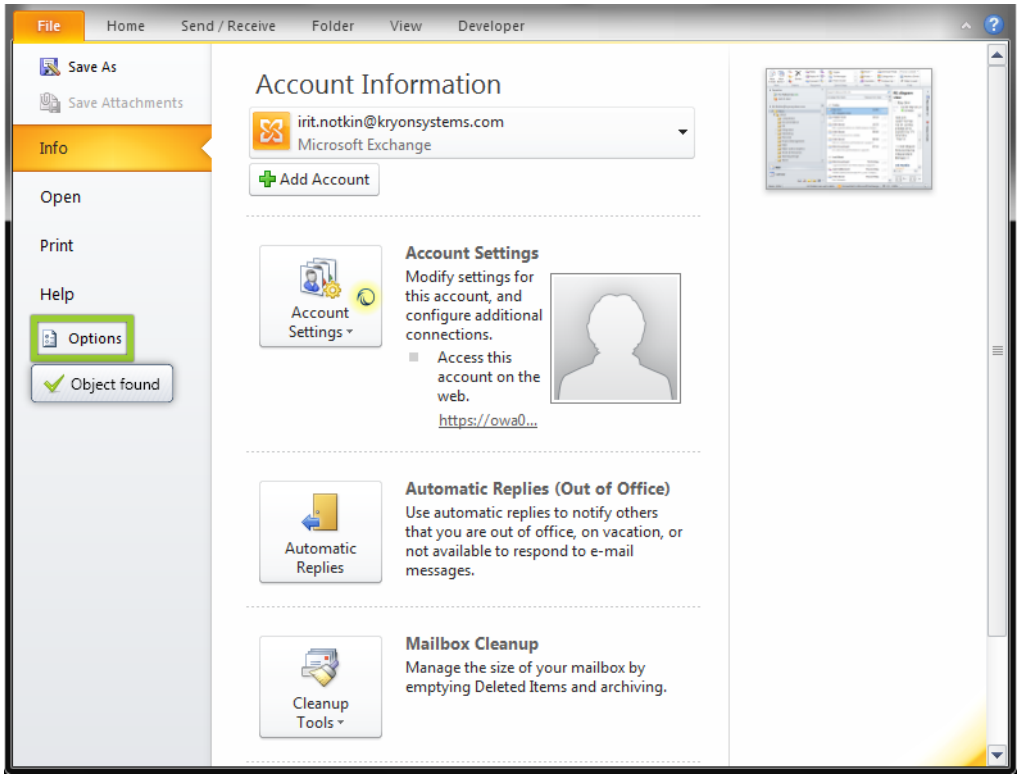
- Which areas of the screen Leo is searching at each step: Search area: Click, Stripe, Full. (Figure 141)

Figure 137: Search Area (Click Example)

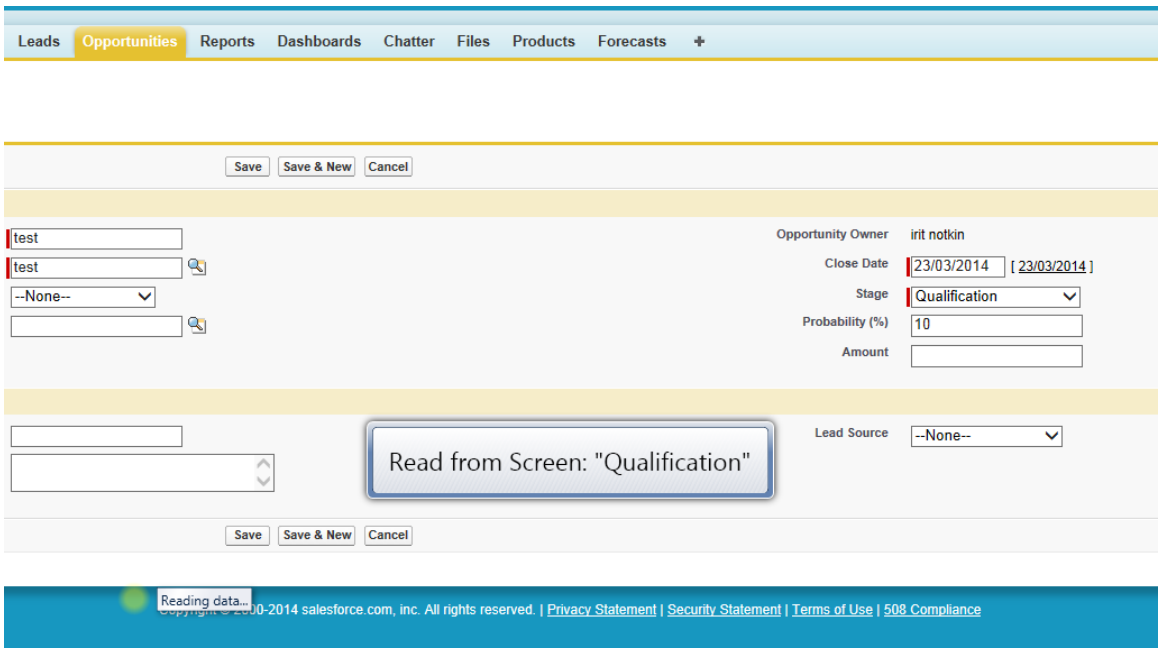


- Whether the object is found (Figure 142)

Figure 138: Click Position

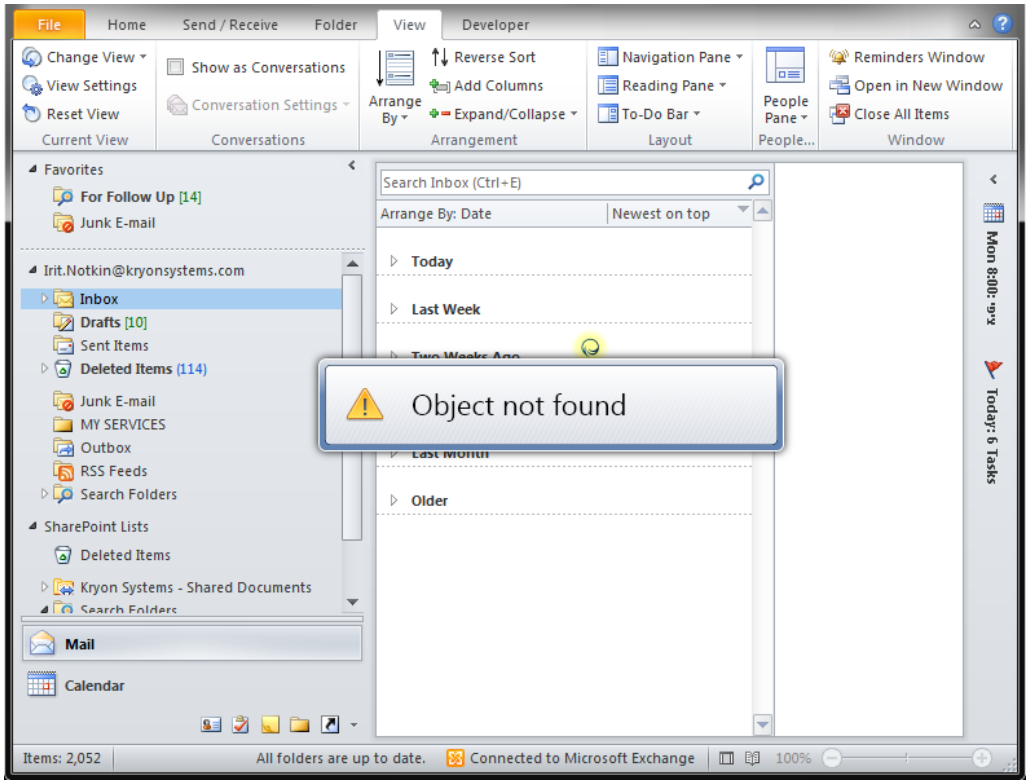


- When Leo reads data from the screen:



- Whether a fallback is triggered:

Figure 139: Fallbacks



Run a wizard in debug mode by doing the following:

1 In the **Wizard Editor** window, select the step from which to start running the wizard in debug mode.
2 On the Wizard Editor menu bar, click **Debug**.
3 Choose how to run the debug mode.
4 The wizard begins to run from the step you selected, showing one by one:
   a The step being played
   b The search area
   c Whether the object is found
   d Whether a fallback is triggered

3.7.1.6 Using Temporary Variables for Testing

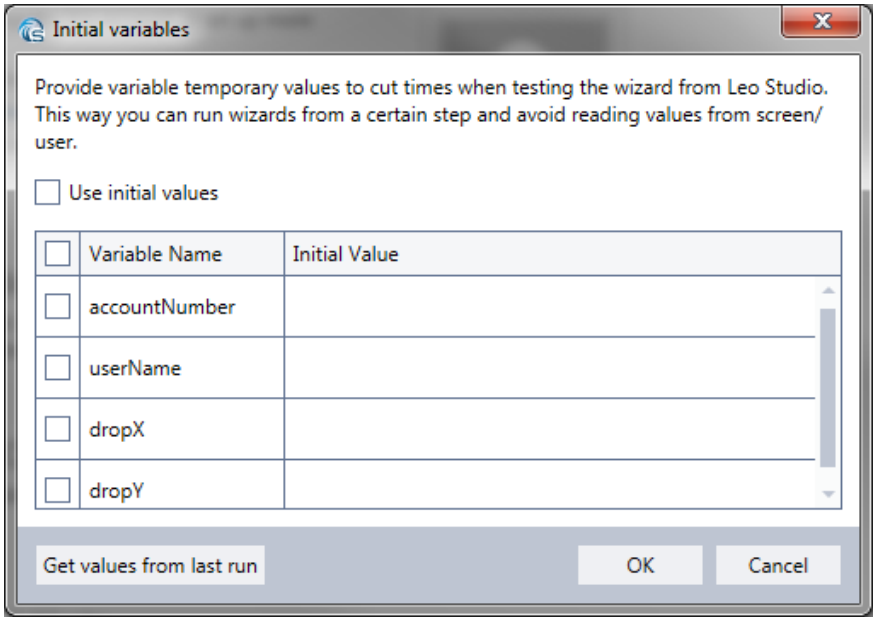
When testing a wizard, you will sometimes need to test from the middle while the variables required for testing get their values in the beginning of the process. For example:

You are testing a wizard of 50 steps called **Creating a New Account ID**, and are testing and tweaking steps 30-35. In the wizard's step 5, Leo reads the new account ID from screen and stores it to a variable called **AccountID**. If you test the wizard from the beginning, the AccountID variable gets the value that was entered by the user (e.g. **123456**). However, if you need to test the wizard from step 30, the variable value is not in memory yet since Leo did not go through step 5 to retrieve the value. In this case, you may want to set up a temporary, fake value (e.g. **000000**) for the AccountID variable, to successfully test the wizard without having to run it from step 5.

Set up a temporary variable as follows:

1 From the Wizard Editor window, click **Tools > Set Initial Variables...**.
The **Initial Variables** dialog box appears (Figure 144).

Figure 140: Set Up Initial Variables



2 Select the checkbox for the variable for which to set a temporary value.

3 Under the **Initial Value** column, enter a temporary value for the variable you selected.

The temporary variable is set up.

4 To use the temporary values you selected, select the **Use Initial Values** checkbox (see Table 8).

Table 8: Using Initial vs. Actual Values

If...		Then...	
Initial Values Checkbox Is:	Actual Values Are:	Actual Values Are:	Initial Values Are:
Selected	Retrieved	Used	Ignored
Cleared	Not retrieved	None	None
Selected	Not retrieved	None	Used
Cleared	Retrieved	Used	None

3.7.2 Testing Leo Wizards in Expected Environments

After verifying the basic functionality, it is recommended that you verify your Leo Wizard works properly under the system environments that can be expected for the target audience.

The environment components that must be tested are:

- **Window Size and Resolution**: Some objects might change their appearance and position or disappear entirely when the window size or screen resolution is changed. Play the Leo Wizard in different window sizes and screen resolutions to check if Leo plays it correctly. If the Leo Wizard fails, append steps and add fallbacks and bubbles as needed.
- **Window Caption**: Window captions might vary from one user to another, and some captions are user specific. Verify that window captions in the Leo Wizard are as generic as possible, and play the wizard to check if Leo detects all windows correctly. If the Leo Wizard fails, edit the relevant window captions as needed.

- **Object Modifications**: Some object in applications are customizable and can be moved, reformatted or removed by the user. Play the Leo Wizard when the object is in a different area of the screen or missing, to check if Leo plays it correctly. If the Leo Wizard fails, append steps and add fallbacks and bubbles as needed.

  Example: In Outlook, it is possible to remove the **Categorize** button from the toolbar. If the Leo Wizard requires Leo to click on the **Categorize** button, play the wizard with this button missing from the screen, and check if Leo still plays it correctly. If the Leo Wizard fails, record a fallback flow that adds the **Categorize** button to the toolbar, or add shortcut keys that access the **Categorize** button without clicking it.
- **Application Windows**: Some Leo Wizards start playing from specific types of windows, such as an outgoing message window. Play the Leo Wizard from another type of window in the application to check if Leo plays it correctly. If the Leo Wizard fails, append steps and add fallbacks and bubbles as needed.

  Example: To insert an object into an outgoing message, an outgoing message must be open on your desktop. Play the Leo Wizard with no open outgoing message to check if Leo plays it successfully. If the Leo Wizard fails, record a fallback flow that opens an outgoing message for the user, or add bubbles that instruct the user to open an outgoing message.
- **Operating System**: For desktop applications, some window classes, buttons, fonts and other objects differ from one Microsoft Windows operating system to another. Play the Leo Wizard in a different operating system to check if Leo detects all windows and objects correctly. If the Leo Wizard fails, append steps and add fallbacks and bubbles as needed.

  Example: In Microsoft Windows XP, the **Save As** window class is **#32770**, while in Windows 7 it is **bosa_sdm_msword**. To ensure that Leo detects the window in both operating systems, do one of the following:

  - Append a fallback step containing the other operating system's window class.
  - Add a window class to the existing step.
- **Web Browser**: For Web applications, some window classes, buttons and fonts vary from one Web browser to another. Play the Leo Wizard in different Web browser to check if Leo detects all windows and objects correctly. If the Leo Wizard fails, append steps and add fallbacks and bubbles as needed.

  Example: In Google Calendar, the Save button font differs between Microsoft Internet Explorer, Google Chrome and Mozilla Firefox. To ensure that Leo detects the button in all three operating systems, do one of the following:

  - Append a step in which you click a stable text near the button, and then edit the step's click offset so that Leo clicks the button.
  - Append a fallback step that contains the button for each Web browser.

3.7.3 Optimizing Performance

When testing a Leo Wizard, some steps must be edited to optimize performance. The following sections provide you with options and tips for optimizing performance.

If you need assistance optimizing wizard performance, contact the Kryon Systems team.

3.7.3.1 Window Detection Optimization

If Leo fails to detect the step window, i.e. pauses (Figure 145), perform the optimization checks listed in Table 9.

Figure 141: Window Detection Pause

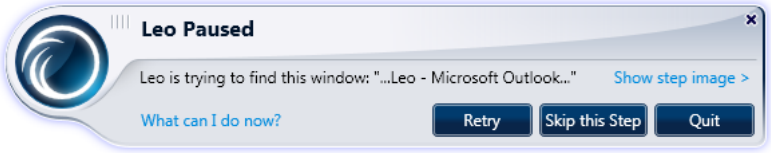


Table 9: Window Detection Optimization

Cause	Description	How to Optimize
Incorrect window caption	The window caption for the step is user-specific. It contains information relevant to a specific window.	Remove any user-specific information from the window caption, and make sure it is set to **Contains** (see Section 3.6.2.2.2.1).
Closed window	The user closes a window before Leo can detect it.	Add fallbacks to the step's Window Detection before Step phase, to enable Leo to either reopen or skip the closed window and continue the Leo Wizard (see Section 3.6.2.1.4.9.7).

3.7.3.2 Object Detection Optimization

If Leo does fails to detect the object, i.e. pauses (Figure 146), perform the optimization checks listed in Table 10.

Figure 142: Object Detection Pause

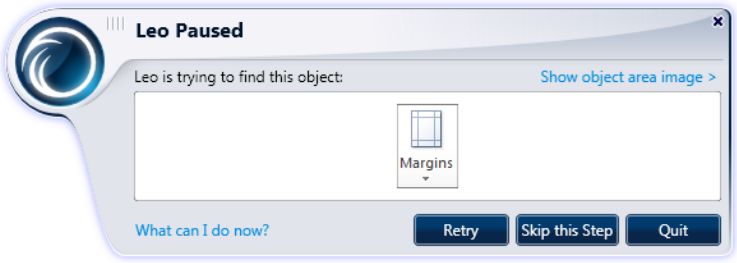


Table 10: Visual Detection Optimization

Cause	Description	How to Optimize
Customizable object	The object Leo is trying to detect is a customizable GUI element, whose appearance or location changes.	Add a fallback to the **Window Detection before Action** phase in the step (see Section 3.6.2.1.4.9.7).
Wrong core action	The wizard flow is incorrect because an unneeded action or step is played.	Edit the step's core action to fix the wizard flow and disable unnecessary clicks or keystrokes (see Section 0).

Cause	Description	How to Optimize
Incorrect object detected (1)	The visual detection settings are incorrect, or the image that was selected in the **Position** tab's **Object Image** area does not match the image that appears on the user's screen.	Choose a different image from the **Object Image** area, and/or change the settings in the **Position** dropdown list. For more information about editing the object position, see Section 3.6.2.1.4.9.1. For more information about editing the object image, see Section 3.6.2.1.4.9.2.
Incorrect object detected (2)	Leo does not detect the object and selecting a different image variation image did not solve the issue.	Check the visual detection match percentages and decrease them.

If Leo detects the wrong object, perform the optimization checks listed in Table 11.

Table 11: Detection Issues

Cause	Description	How to Optimize
Incorrect detection settings	Too many objects match the detection criteria, and Leo clicks the wrong image because the visual detection match percentages are too low.	Check the visual detection match percentages and increase them.
Similar objects in the click position area	There is a similar object in the area surrounding the click position, and Leo clicks an incorrect object.	Customize the detected object image so that it is unique, or use click offset as needed. For more information about customizing the detected object, see Section 3.6.2.1.4.9.2. For information about using click offset, see Section 3.6.2.1.4.9.6.

3.7.4 Setting the Leo Studio Runtime Options

You can set the runtime options for how a Leo Wizard is played in Leo Studio, such as sound, animations, and mouse speed.

This is useful, for example, if you want to test faster in Do It mode by selecting a faster cursor speed.

Any changes to the **Leo Studio** play options do not affect the play options in **Leo**, and vice versa.

The Leo Studio runtime options are:

- Mouse Cursor: Defines the cursor speed, cursor highlighting, and whether to highlight the click position before it is clicked.
- Sound: Defines the sound volume, bubble narration and mouse and keyboard sounds
- Click Indication: Defines the click animation.

Set the Leo Studio runtime options by doing the following:

1. On the Wizard Editor menu bar, select **Tools > Options**.
2. Select the **Playback** tab, and set the relevant options.
3. Click **OK**.

3.7.5 Saving a Wizard

Leo Studio provides the following Leo Wizard saving options:

- Saving to the Database: Saves your Leo Wizard on the company server.
- Saving Locally: Saves your Leo Wizard as a file in a location that you specify.

In addition, Leo Studio includes two mechanisms that protect wizards against loss of changes and wizard versions. These mechanisms are:

- Wizard Check-In Mechanism: Notifies you when a wizard is currently open for editing on another user's machine
- Wizard Backups: For each wizard that you open in the Wizard Editor, a backup is automatically created.

3.7.5.1 Saving to the Database

Save your Leo Wizard to the company database by doing the following:

1. Do one of the following:
   - On the toolbar, click the **Save** icon.
   - On the Wizard Editor menu bar, select **Wizard > Save**.
   - Press **CTRL+S**.

   When the confirmation appears, the Leo Wizard is saved to the company database.

You can also save the wizard locally, as described in Section 3.7.5.1, and later import the wizard to the database, as described in Section 3.7.5.3.

3.7.5.2 Saving Locally

Saving a Leo Wizard locally stores the wizard as an .lwiz file on your computer in a location that you specify. This is useful for backing up the Leo Wizards that you create. The file can then be imported into another Leo Wizard and edited. For more information about the uses of imported Leo Wizards, see Section 3.7.5.3.

To save a Leo Wizard locally:

1. Do one of the following:
   - On the Wizard Editor menu bar, select **Wizard > Save Locally**.
   - Press **CTRL+SHIFT+S**.

   If this is the first time the Leo Wizard is saved locally, the **Save As** dialog box appears.
2. Specify a filename and location for the Leo Wizard.
3. Click **Save**.

To cancel the saving process, click the X on the wizard progress bar.

The Leo Wizard is saved locally as an .lwiz file to the location that you specified.

To save the locally saved Leo Wizard under a different filename or in a different location, select **Wizard > Save Locally As** on the Wizard Editor menu bar.

3.7.5.3 Wizard Check-In Mechanism

Leo Studio has a check-in/check-out mechanism that notifies you when a wizard is currently open for editing on another user's machine. When a wizard is open for editing in the user's Wizard Editor, the wizard is automatically checked out and available as a read-only copy to all other users.

While the wizard's content, that is, recorded steps, is checked out, the wizard properties and location in the catalog remain editable by all users.

If you attempt to open a wizard that is checked out by another user, a notification message appears, providing the following options (Figure 147):

Figure 143: Checked-Out Wizard Options

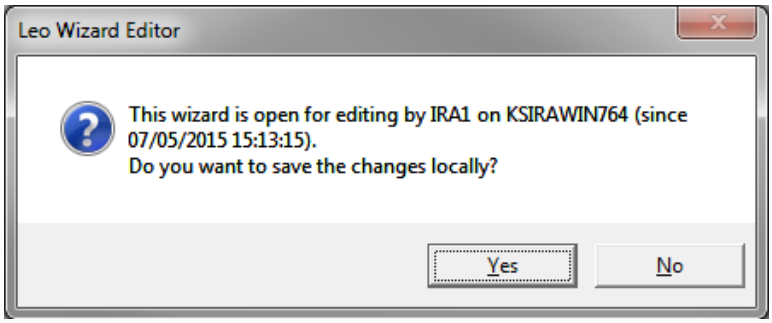
- **Take Over**: Dismisses the current checkout by the other user, and the following occurs:
  - You become the owner of the wizard, and it is checked out for you only.
  - The other user's checkout copy turns into a read-only copy (Figure 148).

    Figure 144: Read-Only Wizard

  - If the other user attempts to save the wizard to the database, they receive a notification that the wizard was taken over (Figure 149).

    Figure 145: Wizard Taken Over by Another User

  - When the wizard is saved to the database, the takeover action is added to the changes history list (Figure 150).

Figure 146: Wizard Takeover in Changes History Tab

5/7/2015 3:13:20 PM	Ira1	Wizard take over	Wizard was taken over by user Ira1 on KSIRAWIN764 (on 07/05/2015 15:13:15)	3.10.0.42

- **Open Read-Only**: The wizard opens in a read-only copy (Figure 148). Attempting to save the copy to the database results in a notification that it cannot be saved to the database (Figure 149).
- **Cancel**: Opening the wizard is canceled and the wizard is not opened in the Wizard Editor.

3.7.5.4 Wizard Backups

For each wizard that you open in the Wizard Editor, a backup is automatically created. The wizard backup is periodically updated with recent changes while the wizard is open.

All backup files are locally saved in the computer's local **My Documents > Leo > Auto Save**, in the wizard's **<Wizard Name and ID>** folder.

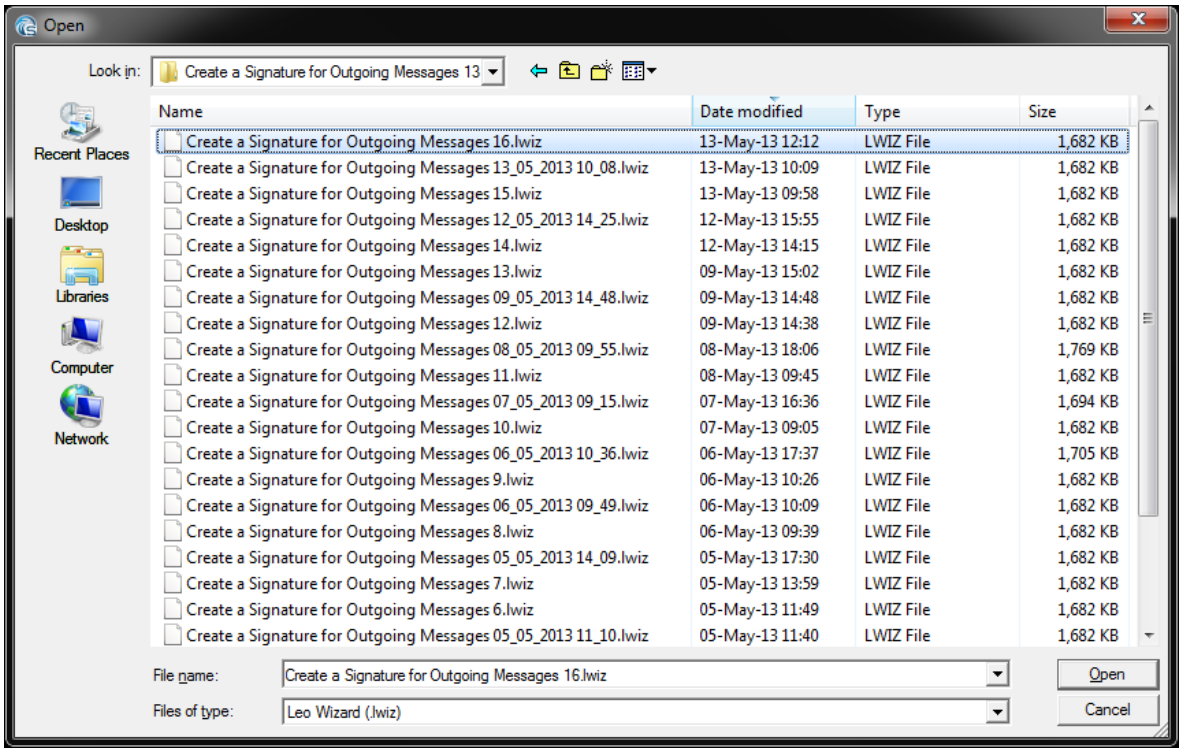
For each wizard opened in the Wizard Editor, two backup files are automatically created:

- Wizard Opened: Created when the wizard is opened in the Wizard Editor. The filename format is:
  <Wizard Name> <Unique Serial Number>.lwiz
- Latest Changes: The filename format is:
  <Wizard Name> <Date and Time of First Auto-Save>.lwiz

The file is created a predefined amount of time (configurable) after first opening the wizard, and gets updated with any changes every X minutes. The default auto-save interval time is 10 minutes.

Figure 151 shows the backup files for the Create a Signature wizard in the **Auto-Save** folder.

Figure 147: Backup Wizard Files in an Auto-Save Folder



For information about how to set the default Auto-Save intervals and other Auto-Save options, see Section 3.7.5.4.2.

3.7.5.4.1 Auto-Save Storage Capacity

The capacity limit of the backup **Auto-Save** folder is predefined in the Leo Studio Options menu. The default storage capacity is 100 MB. When the folder exceeds its capacity, the oldest files in **Auto-Save** are automatically deleted to meet the predefined capacity.

Auto-save is enabled automatically when at least 1 GB of RAM is available. If this RAM requirement is not met, Leo Studio notifies you that auto-save is automatically disabled.

For information about how to set the default Auto-Save options, see Section 3.7.5.4.2.

3.7.5.4.2 Setting the Default Auto-Save Options

The Auto-Save folder options, i.e. the time interval and storage capacity, can be set by doing the following:

1 On the Wizard Editor menu bar, select **Tools > Options > Advanced**.
2 In the **Auto Save** area, set the auto-save time interval and folder capacity.

To open an automatically saved wizard in the Wizard Editor, see Section 3.7.6.

3.7.6 Opening a Locally Saved Wizard in the Wizard Editor

This section describes how to open a locally saved Leo Wizard (.lwiz file) in the Leo Wizard Editor.

Opening locally saved Leo Wizards can be useful for the following purposes:

- Copying a Leo Wizard to a different location in the Catalog library
- Creating a new Leo Wizard based on an existing wizard
- Retrieving an earlier version of a Leo Wizard for backup, rollback, or reference purposes
- You can open locally saved wizards into an either blank or recorded wizard.

When you open a locally saved wizard from within an existing recorded wizard, all steps in the existing wizard are overwritten. To add steps to an existing Leo Wizard without overwriting it, see Section 3.6.1.3.

Example: In an outgoing message, the procedure for inserting an equation is almost identical to the procedure for inserting a table. You can save one of these wizards locally and open the local file in order to create the other Leo Wizard. You can then edit the opened local file by appending the required step and editing the bubbles. This prevents you from having to record a new Leo Wizard from scratch.

For information about saving Leo Wizards locally, see Section 3.7.5.1.

Open a locally saved wizard in the Wizard Editor by doing the following:

1 Do one of the following:
   - In a new or existing Leo Wizard, on the Wizard Editor menu bar, select **Wizard > Open Local File**.
   - Press **CTRL+O**.

   The **Open** dialog box appears.

When you open a Leo Wizard from within an existing wizard, all existing steps are overridden. To add steps to an existing wizard without overriding it, see Section 3.6.1.3.

2 Select the .lwiz wizard file to open.
3 Click **Open** to open the file into the Leo Wizard.
4 The wizard file is opened in the Wizard Editor, overriding any existing steps.

3.7.7 Handling Leo Performance Issues

When a Leo performance issue occurs and you are unable to resolve it by editing the wizard/sensor, it is important to notify the Kryon Systems Support team and work with them to resolve the issue.

The resolution process involves sending Leo logs to Kryon Systems Support. When Leo Player or Leo Studio is run, every Leo action is written to Leo log files. Log files contain performance details that enable Kryon Systems Support to troubleshoot Leo Studio or Leo Player issues. For some issues, resolution will depend on the information in these logs.

Handle a Leo performance issue by doing the following procedures:

1 Notifying Kryon Systems Support
2 Enabling Detailed Logs
3 Sending Log Files to Kryon Systems Support

3.7.7.1 Notifying Kryon Systems Support

1 Using the support credentials provided to you, open a support ticket at http://kryon.h2desk.com.

If you cannot find your support credentials, contact Support@kryonsystems.com.

2 In the support ticket, describe the Leo issue, providing specific details of:
- The Leo and/or application behavior during the issue
- The time in which the issue occurred
- Repeatability of the issue: can it be recreated?
- The Leo client (Studio and/or Player)
- The name of the user or computer for which the issue occurred

3 Do one of the following:
- If the issue **can** be recreated from the user's computer, continue with 3.7.7.2 Enabling Detailed Logs.
- If the issue **cannot** be recreated from the user's computer, continue with 3.7.7.3 Sending Log Files to Kryon Systems Support.

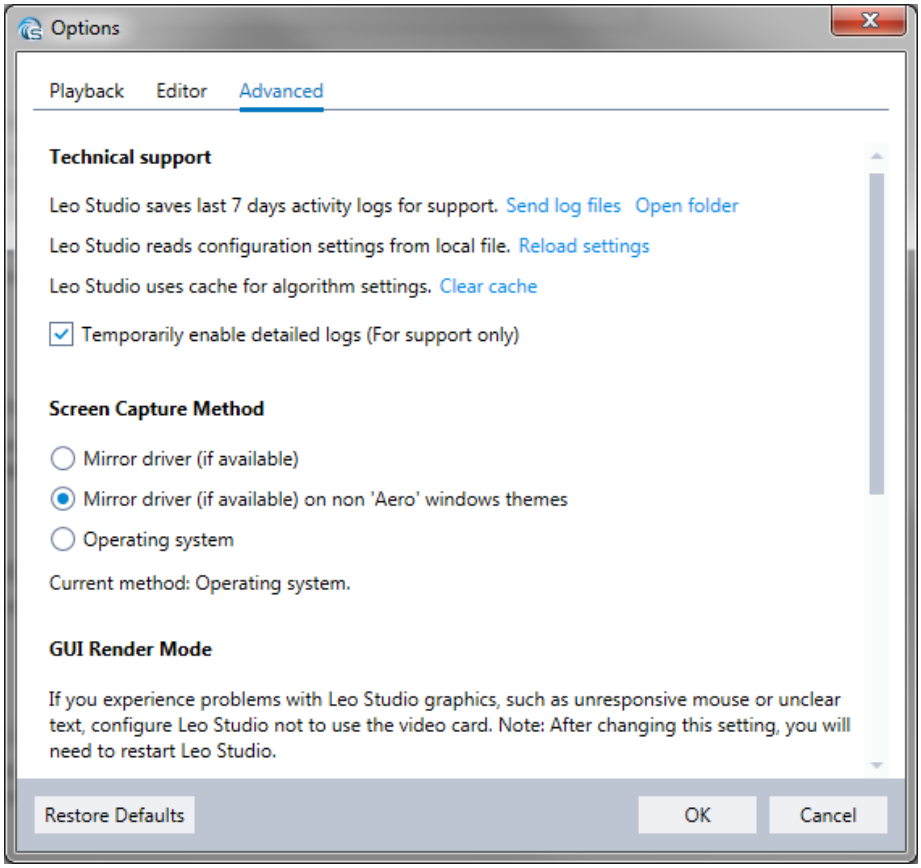
3.7.7.2 Enabling Detailed Logs

If the issue **cannot** be recreated from the user's computer, skip this procedure and continue with 3.7.7.3 Sending Log Files to Kryon Systems Support.

1 On the computer where the issue occurred, identify whether it occurred in the Leo Player or Leo Studio client.
2 From the Leo client you identified in step 1, click **Tools > Options > Advanced**.
3 Under the **Technical Support** section, select the checkbox titled **Temporarily enable detailed logs for support** ().

Detailed logs are enabled.

Figure 148: Enabled Detailed Logs in Advanced Options



4 Recreate the issue on the computer where the issue occurred, noting to yourself the exact time at which it occurred (e.g. 4:23 pm).

5 Continue with 3.7.7.3 Sending Log Files to Kryon Systems Support.

3.7.7.3 Sending Log Files to Kryon Systems Support

1 On the computer where the issue occurred, identify whether it occurred in the Leo Player or Leo Studio client.

2 From the Leo client you identified in step 1, click **Tools > Options > Advanced**.

3 Click **Send Log Files**.
The **Leo Log Sender** dialog box appears, displaying **Step 1: Select files to send** (Figure 154).
You can also launch the Log Sender from the Leo Splash screen (Figure 153)

Figure 149: Launch Log Sender from Splash Screen

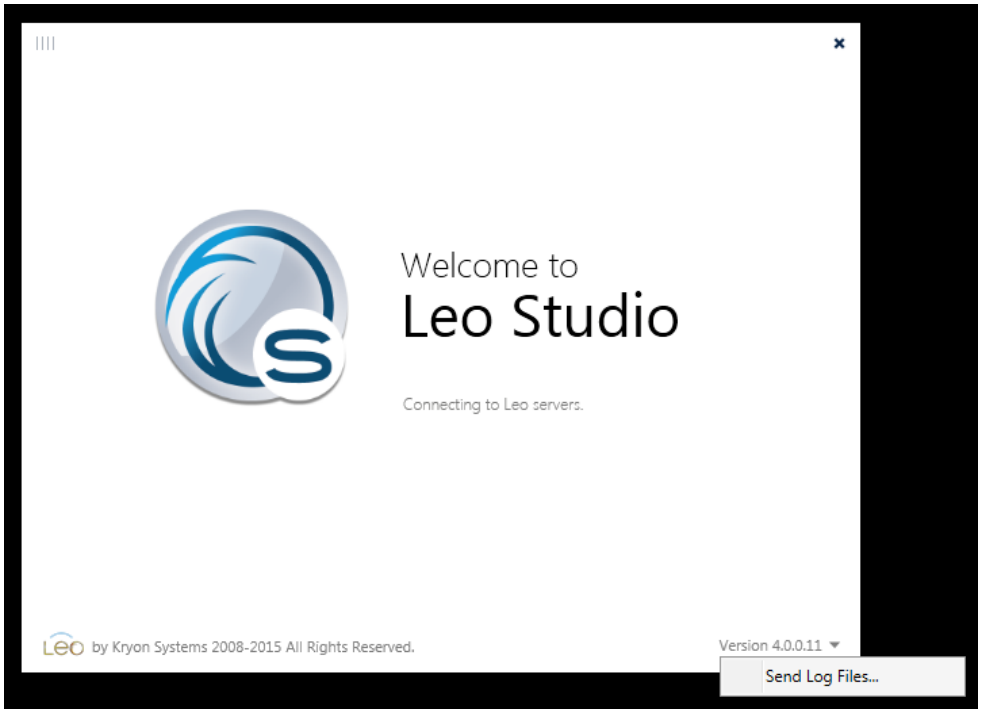
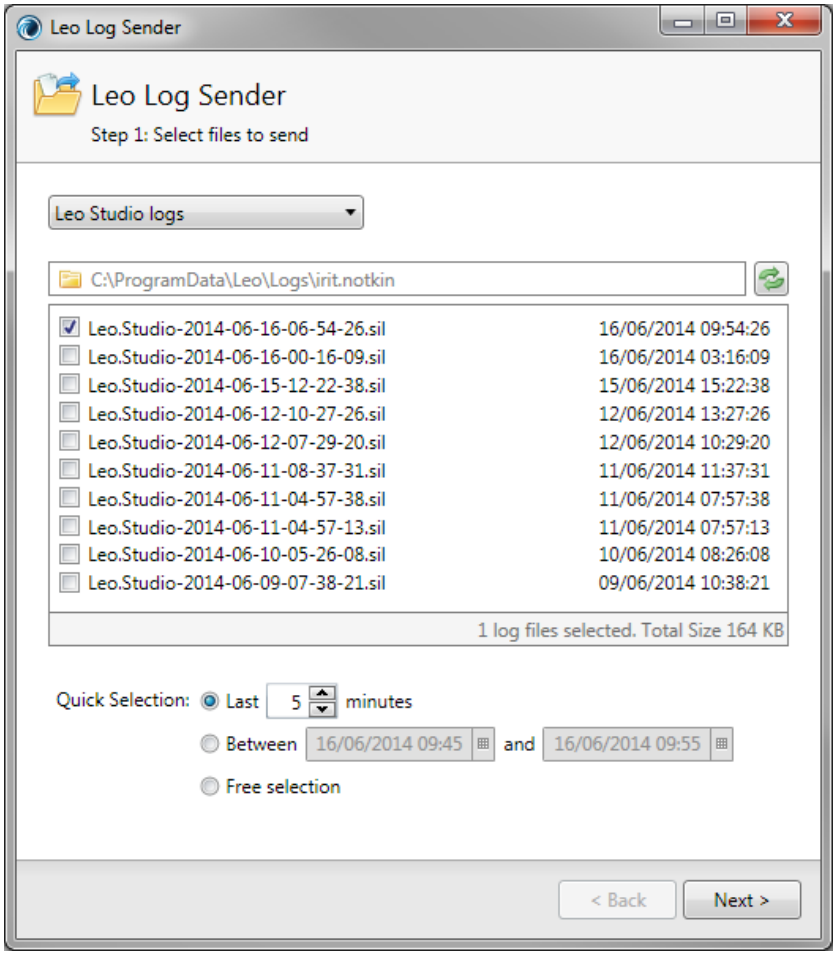


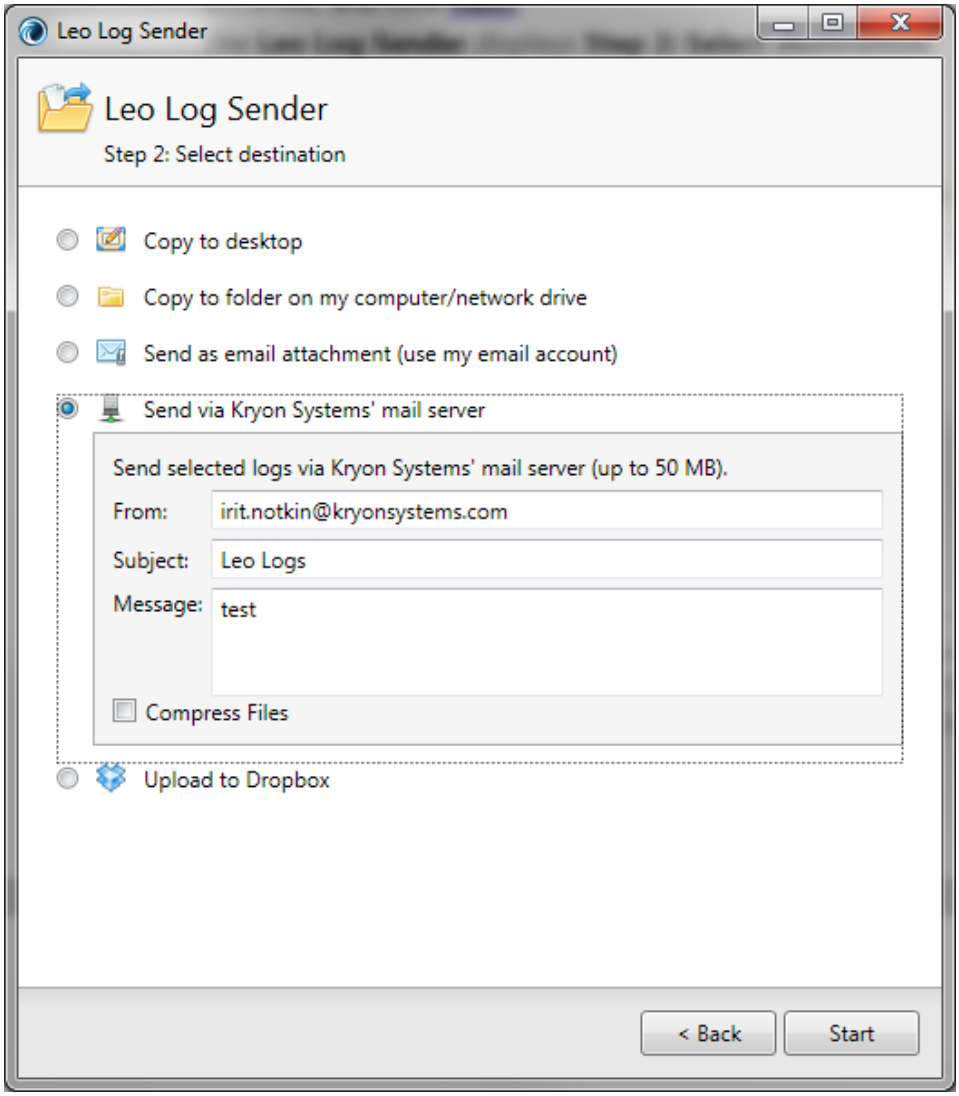
Figure 150: Leo Log Sender Dialog Box



4 From the client selection dropdown list, select the client you identified in step 1.

5 From the **Quick Selection** options, select the time or time range at which the issue occurred, and click **Next**.
The **Leo Log Sender** displays **Step 2: Select destination** (Figure 155).

Figure 151: Log Sender Step 2: Select Destination

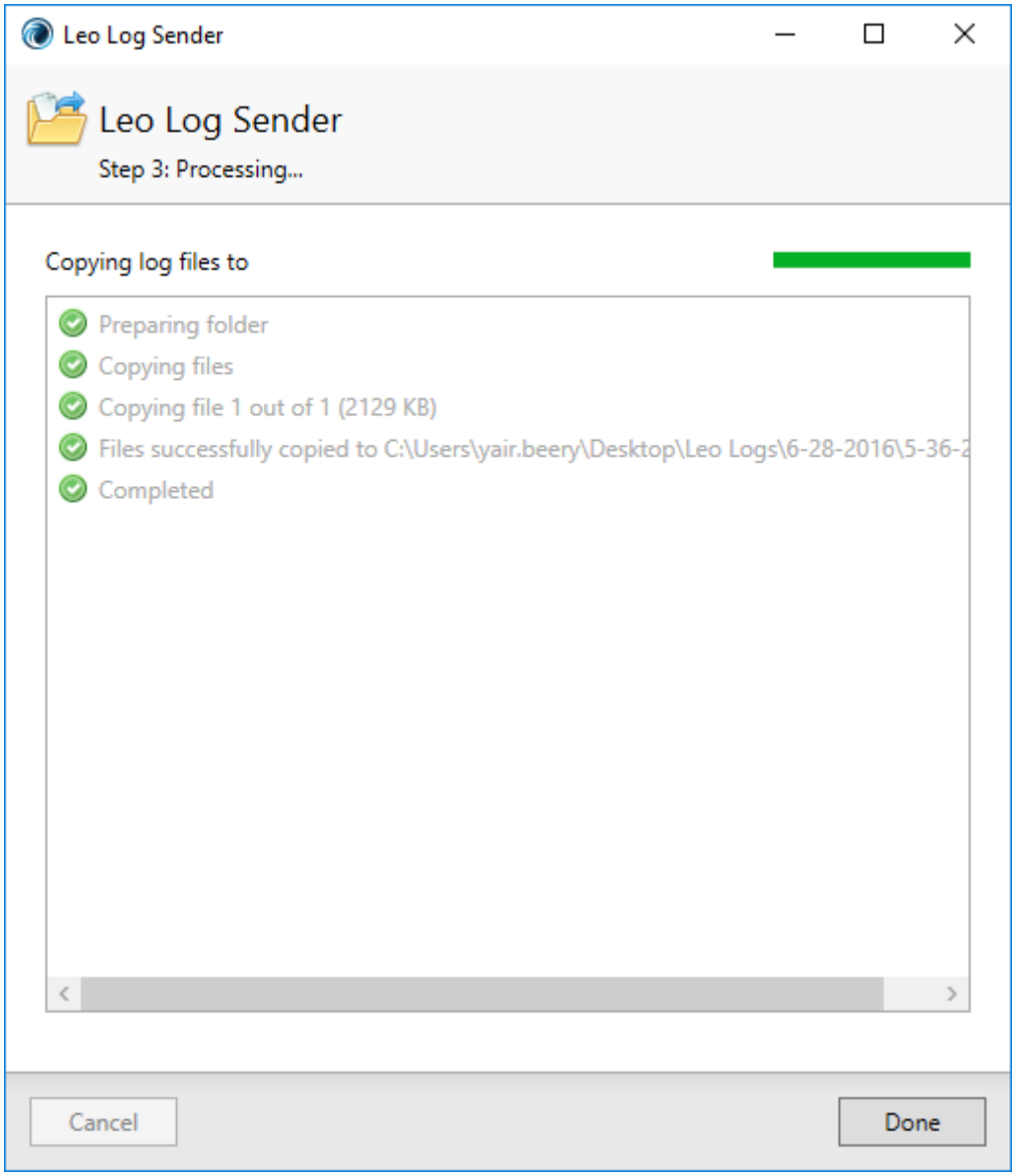


6 Choose the method for sending the log files.

Click an option to view its description and any additional steps.

7 Click **Start**.
The Leo Log Sender displays **Step 3: Processing...** (Figure 156).

Figure 152: Log Sender Step 3: Processing



8 When the processing is completed, click **Done**.
The logs are sent or saved, depending on the option you selected in step 7.

3.8 Managing Leo Content

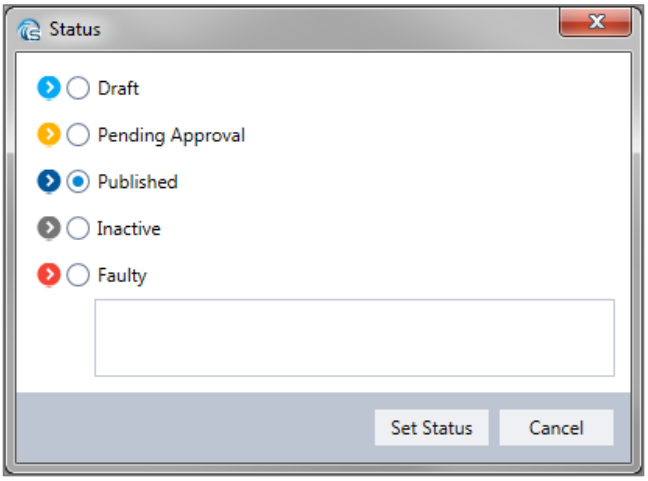
Leo Studio can store many categories and Leo Wizards that you can publish to end users, reorganize, count, and collaborate on as needed. The following sections describe how to manage Leo content in the catalog.

3.8.1 Publishing a Wizard or Changing its Status

Publish a Leo Wizard to end users by doing the following:

1 From Leo Studio, in the **Catalog** pane, select the Leo Wizard you want to publish.
2 In the **General** tab, the **Browse** button next to the **Status** field.
3 The **Wizard Status** dialog box appears (Figure 157).

Figure 153: Wizard Status Dialog Box



4 To publish the wizard, select **Published**. Otherwise select the relevant status.
5 Click **Set Status**.
6 Save your changes by clicking **Save Changes**.
The wizard becomes available to all end users who have Leo Player installed on their machines, and who have permissions to access this wizard.

3.8.2 Organizing the Leo Catalog

Organize the Leo Catalog to suit your organization's needs by moving Leo Wizards and categories around the hierarchy tree.

Organize the Leo Catalog by doing the following:

1 From Leo Studio, in the **Catalog** pane, select the Leo Wizard or category that you want to move.
2 Drag the wizard or category and drop it in the tree location that you want.
The Leo Wizard or the category and its entire contents are moved to the new location.

3.8.3 Importing and Exporting Leo Content

The Leo Studio Content Assistant enables you to import and export Leo content from the Leo Studio catalog. The Leo content elements that can be imported and exported are:

- Leo Wizards/Sensors
- Wizard/Sensor Categories
- Leo Libraries

Content is exported and imported in CTP format. All properties of the content are included such as the catalog path, related and embedded wizards, links and supported applications.
The following sections describe how to import and export Leo Studio content using the Content Assistant.

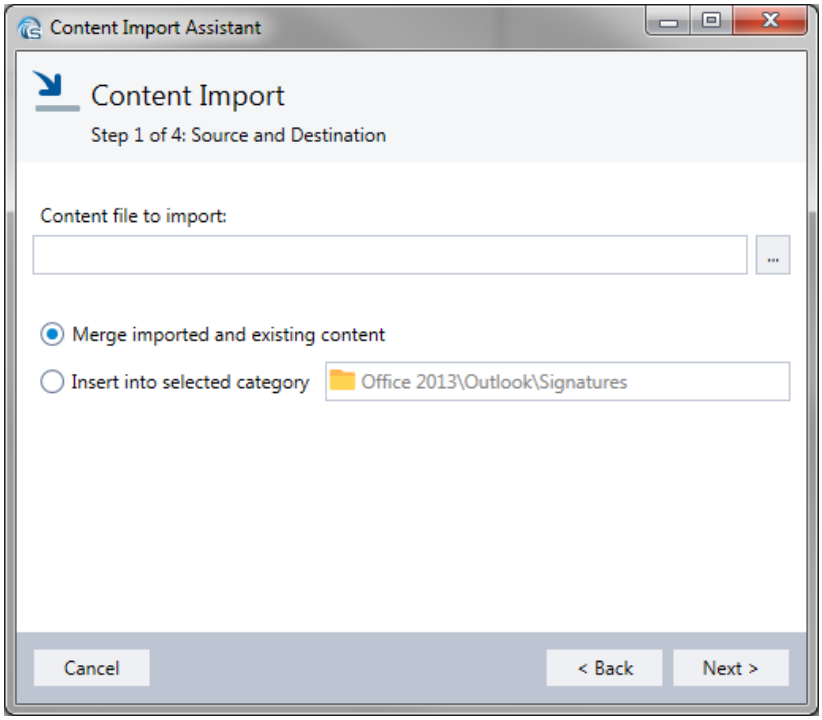
3.8.3.1 Importing Leo Content into the Catalog

Leo content can be imported into the catalog in two ways (Figure 158):

- **Merge Imported and Existing Content**: Places imported content in its matching path.

- If the existing path fully matches the imported path, matching wizards are overwritten by imported wizards. Existing wizards without a matching imported wizard are not deleted. Non-existing wizards are created under the imported path.
- If the existing path does not fully match the imported path, existing wizards are updated under the imported category path. Non-existing wizards are created under the imported path.

Insert into Selected Category: Places imported content under path selected by the user. Matching wizards are either duplicated or moved under the imported category, based on your selection. Wizards that do not currently exist in the selected path are created under the selected path. Figure 154: Import Methods



Import wizards in Content Transfer Package (**CTP**) format, as follows:

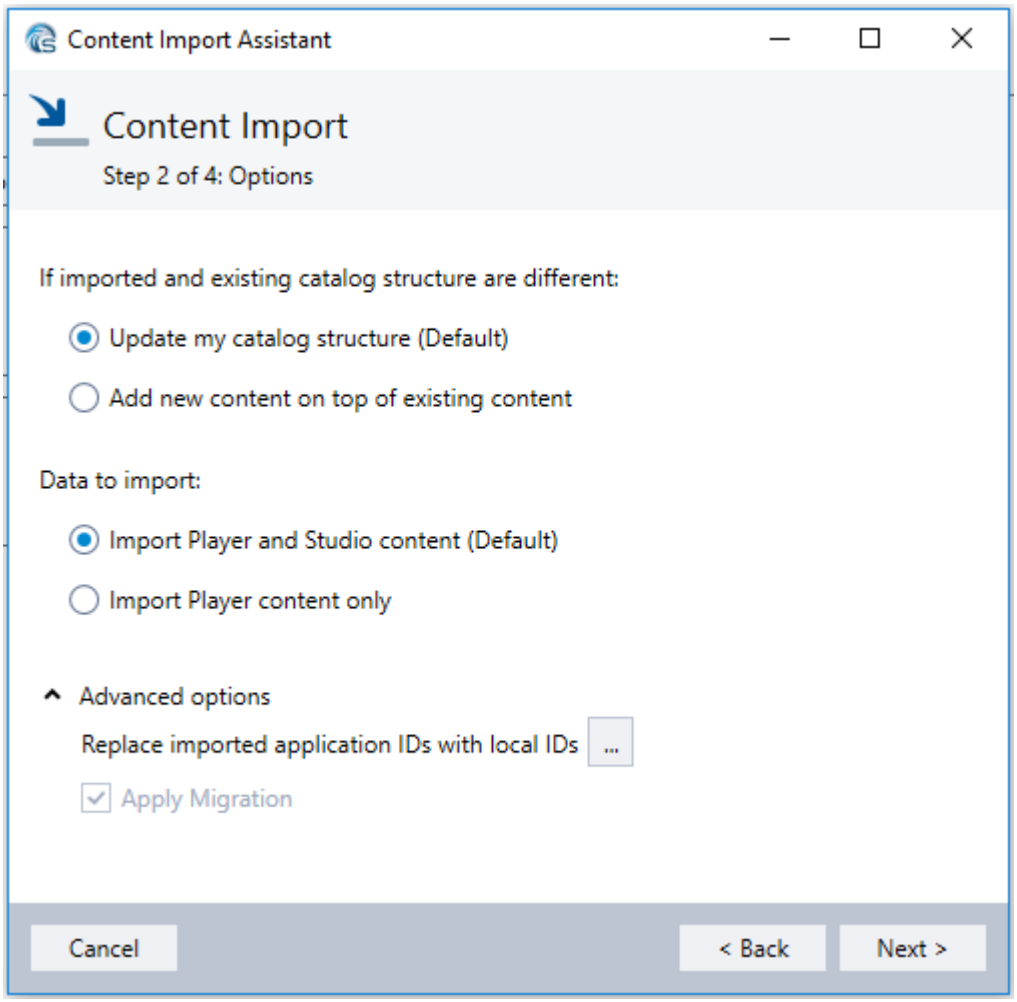
1. From the main Studio window, select the category into which to import the wizard.
2. Right-click the selected category and select **Import/Export > Import from file...**.
3. From the **Content Export Assistant** dialog box, browse for the wizard to import.
4. Choose the method for importing the wizards, as described in Section 3.8.3.1.
5. Click **Next**.
6. Choose how to import the content and which data to import, and click **Next**.
   The wizard data is analyzed.
7. Review and confirm the list of wizards to import, and click **Import**.
8. When the import process is completed, click **Finish**.
   The catalog is updated with the imported wizard in the category you selected.

3.8.3.1.1 Content Version Upgrade using Import

In case that the imported content was created in an older Leo version, the Import process will enable the user to select if to upgrade it into the existing version. See Figure 159 - Apply Migration option.

In case the older content version is significantly different that the existing version, the Import process will automatically upgrade the content to the existing version without giving the user the option not to upgrade it.

Figure 155 - Apply Migration option



3.8.3.2 Exporting Leo Content from the Catalog

Export wizards in DWIZ format and sensors in DSEN format, as follows:

1. From the main Studio window, select the wizard/sensor to export.
2. Right-click the selected wizard/sensor, and select **Import/Export > Export wizard/sensor to file...**.
3. From the **Content Export Assistant** dialog box, click **Next**, and browse for the location to export the wizard/sensor to.
4. Click **Export**.
   The wizard/sensor structure is validated and processed.
5. When the export process is completed, click **Finish** or **Open file location** to view the location of the exported file.
   The content is exported to the selected location in DWIZ format for wizards and DSEN format for sensors.

Empty categories and empty wizards/sensors are not exported.

3.8.4 Generating Leo Outputs

You can generate a number of output types from a Leo wizard. These outputs include Wizard videos, Word documents and PowerPoint presentations. The following sections describe how to generate each of these outputs and more.

3.8.4.1 Exporting a Wizard to Word or PowerPoint

Leo Studio enables you to export Leo Wizards to Microsoft Word and PowerPoint. When you export a Leo Wizard, a DOC/DOCX or PPT/PPTX file is created containing images and details of the wizard and its steps. These wizard elements are organized and formatted in the export file according to a predefined template.

Each Leo installation comes with a number of sample templates for your convenience. These templates are located in the Leo Studio installation folder. For example:

C: > Program Files > Leo Studio > Templates

You can create your own custom templates for wizard export in Microsoft Word or PowerPoint, as described in Sections 3.8.4.2 and 3.8.4.3.

You are **not** required to have Microsoft Word or PowerPoint installed in order to successfully export wizards to a Word or PowerPoint file.

Depending on the template you select, the export file will include some or all of the following wizard elements:

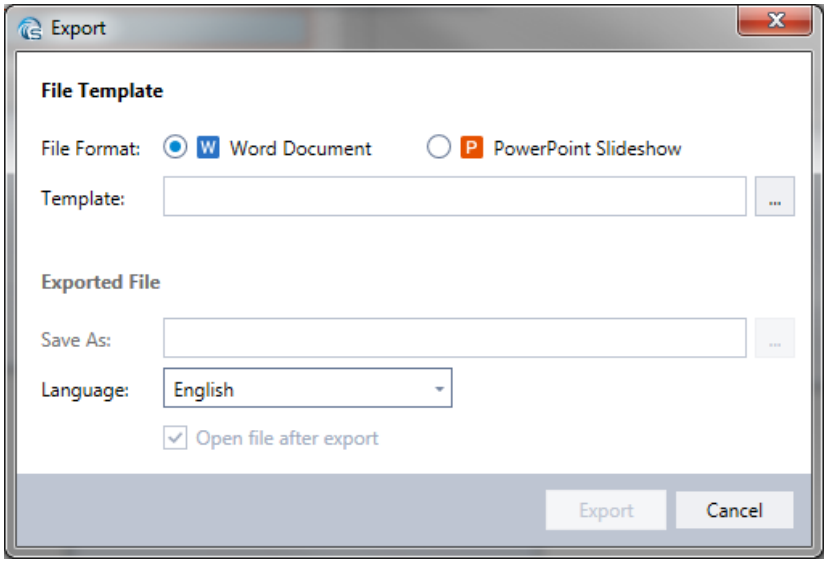
- Wizard name
- Wizard description
- Wizard path
- Wizard hyperlink
- Wizard notes
- Application icon
- Step name
- Step number
- Click position area image
- Full step image
- Core action description (mouse clicks, key strokes, and bubbles)
- Step notes

- Click position image
- Action type icon

Export a Leo Wizard to Microsoft Word or PowerPoint by doing the following:

1 In Leo Studio, open the Leo Wizard that you want to export, as described in Section 2.6.2.3.
2 In the Wizard Editor menu bar, click **Tools > Export to**.
The **Export to** dialog box appears (Figure 160).

Figure 156: Export to Dialog Box



3 In the **Export to** dialog box, select the file format to which you want to export the Leo Wizard.
4 In the **Template** field, browse for the template that you want to use for the exported file.
5 In the **Save as** field, select a location for the export file.
6 Select or clear the **Open file after export** checkbox to determine whether to open the file automatically when the export is finished.

If you do not have Microsoft Word or PowerPoint installed on your computer, clear the **Open file after export** checkbox.

The exported Leo Wizard is saved to the file format you selected, containing wizard and step elements that are structured and formatted according to the template you used.

3.8.4.2 Microsoft Word Export Template Creation

Microsoft Word export templates are DOC or DOCX files used to generate, structure and format exported wizard documents.

In the export process, wizard properties and content elements are inserted into the export file by using predefined fields in the template.

You can use the templates provided with your Leo installation, or you can create your own custom templates for wizard export in Microsoft Word. The following sections describe how to create a Word export template for wizards.

3.8.4.2.1 Microsoft Word Template Fields, Structure and Formatting

In Word, wizard elements are inserted into predefined MergeField fields in the template. All MergeField fields are described in Table 12.

In order to successfully export the wizard to a document, these MergeField fields must be structured according to the hierarchy described in Figure 161.

The formatting used for each field in the template is reflected in the export file.

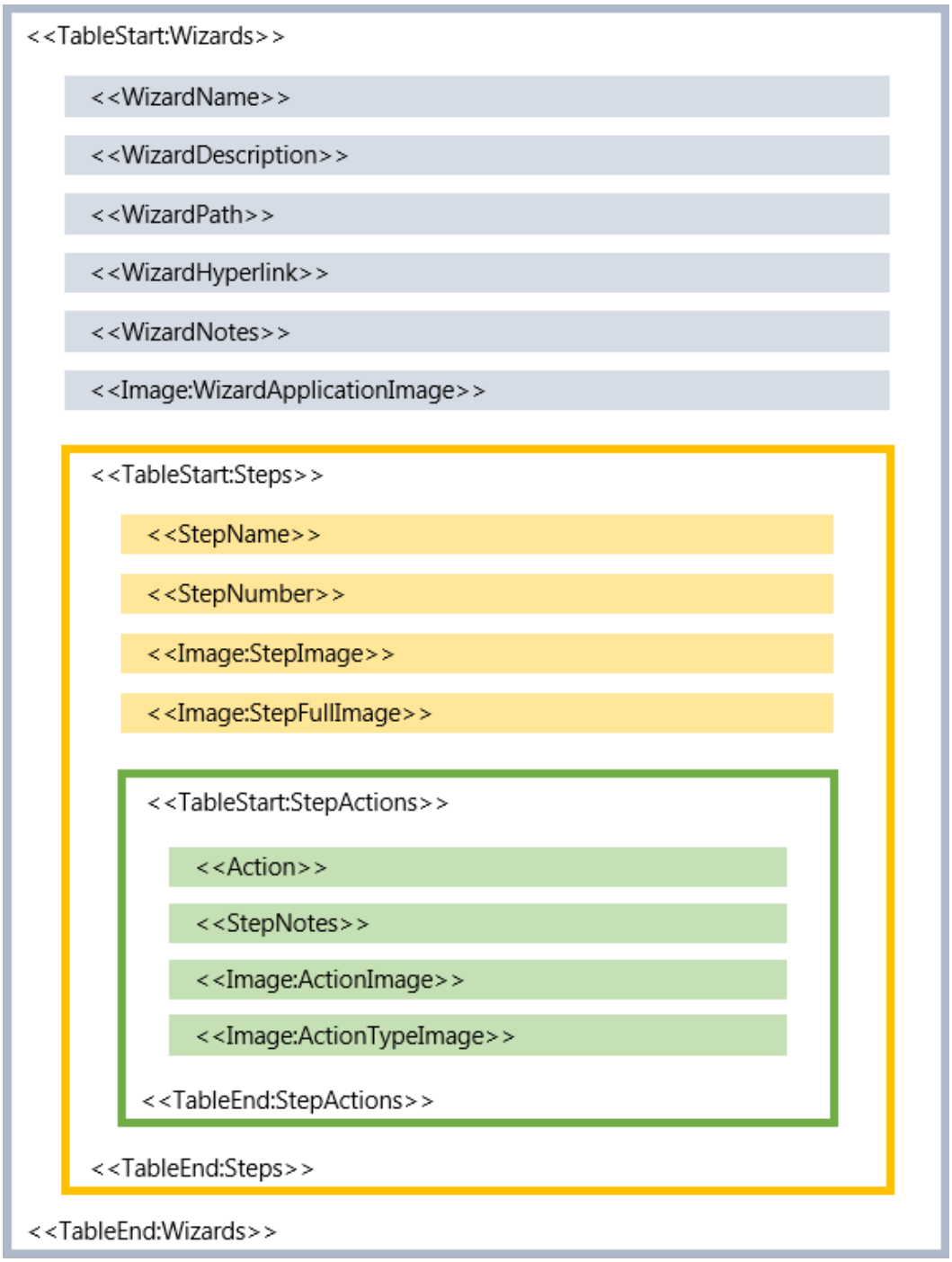
Table 12: Word Export Template – Field Descriptions

Field Name	Description	Related Information
TableStart:Wizards	An opening tag that causes wizard properties to appear in the exported file. This field, along with the TableEnd:Wizards closing field, **must** wrap any other wizard, step and action fields.	
WizardName	The wizard title	
WizardDescription	A description of what the wizard does	
WizardPath	A full path to the library, category and subcategories that contain the Leo Wizard in Leo Studio	Section 2.6.1
WizardHyperlink	A unique, direct link to the Leo Wizard that can be pasted or embedded in an email message, Web browser address bar, or the Windows **Run** box	
WizardNotes	Any notes or comments typed in the wizard's **Notes** tab in Leo Studio	
Image:WizardApplicationImage	An icon of the application on which the Leo Wizard is played	
TableStart:Steps	An opening tag that causes all wizard steps to appear in the exported file in their original sequence. This field along with the TableEnd:Steps closing field, **must** wrap any step and action fields.	
StepName	The step title assigned by the content author	Section 3.6.2.1.1.1.1
StepNumber	The step number assigned automatically in the Wizard Editor	
Image:StepImage	An image of the area that surround the step's click position	
Image:StepFullImage	The full step image, as it appears in the Wizard Editor **Display** pane.	
TableStart:StepActions	An opening tag that causes all the actions of a wizard step to appear in the exported file in their original sequence. This field, along with the TableEnd:StepActions closing field, **must** wrap any other action fields.	

Field Name	Description	Related Information
Action	A description of the mouse click, any key strokes and bubble text (including buttons) in the order in which they appear in a step	
StepNotes	Any notes or comments typed in the step's **Notes** tab	
Image:ActionImage	An image of the object that Leo detects and clicks in the step	Section 3.6.2.1. 4.9.2
Image:ActionTypeImage	An icon indicating the type of action that Leo performs, such as a bubble or mouse click	
TableEnd:StepActions	A closing tag that causes all the actions of a wizard step to appear in the exported file in their original sequence. This field, along with the TableStart:StepActions opening field, **must** wrap any other action fields.	
TableEnd:Steps	A closing tag that causes all wizard steps to appear in the exported file in their original sequence. This field, along with the TableStart:Steps opening field, **must** wrap any step and action fields.	
TableEnd:Wizards	A closing tag that causes wizard properties to appear in the exported file. This field, along with the TableStart:Wizards opening field, **must** wrap any other wizard, step and action fields.	

Figure 161 shows the MergeField hierarchy you must use in your Word export template:

Figure 157: Word Export Template - MergeField Hierarchy

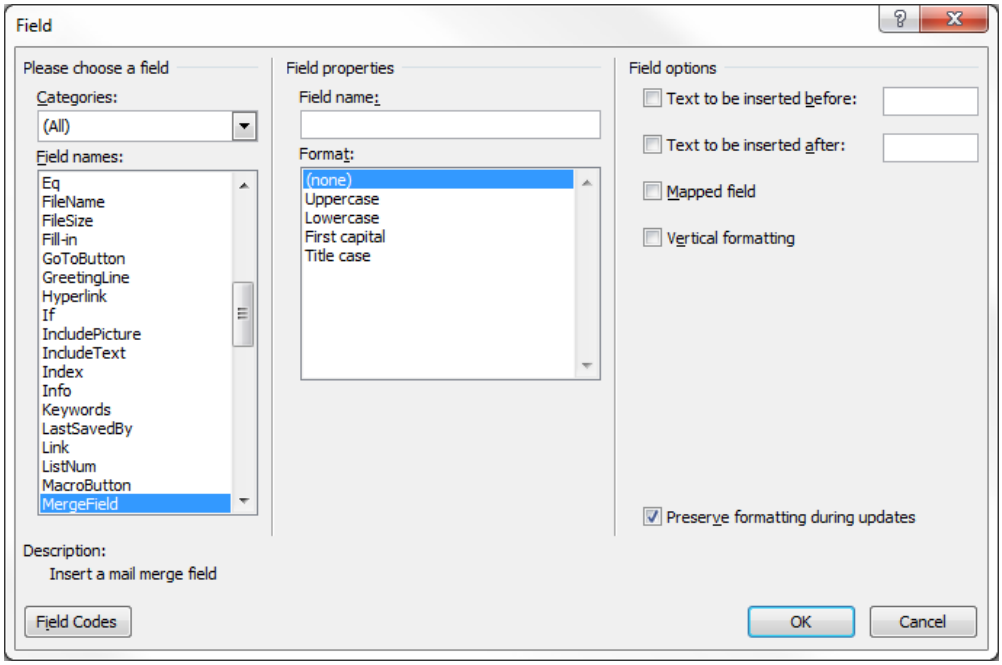


3.8.4.2.2 Creating a Microsoft Word Export Template

Create a Microsoft Word export file template by doing the following:

1. In a blank Microsoft Word document, click **Insert > Quick Parts > Field...**.
2. In the **Field** dialog box, from the **Field names** list, select **MergeField** (Figure 162).

Figure 158: MergeField



3 In the **Field name** field, type:

TableStart:Wizards

For the field names and their descriptions, see Table 12.

4 Repeat steps 1-2, and in the **Field name** field, type the field name for the wizard item that you want to insert, according to the field names in Table 12.

Repeat this step for all wizard items that you want to insert.

5 To insert step items, do the following:

a Repeat steps 1-2, and in the **Field name** field, type:
TableStart:Steps

b Repeat steps 1-2, and in the **Field name** field, type the field name for the step item that you want to insert, according to the field names in Table 12.

Repeat this step for all step items that you want to insert.

c To insert core action items, do the following:

i Repeat steps 1-2, and in the **Field name** field, type:

TableStart:StepActions

ii Repeat steps 1-2, and in the **Field name** field, type the field name for the core action item that you want to insert, according to the field names in Table 12.

Repeat this step for all core action items that you want to insert.

iii When you are done inserting core action fields, repeat steps 1-2, and in the **Field name** field, type:

TableEnd:StepActions

d When you are done inserting step fields, repeat steps 1-2, and in the **Field name** field, type:

TableEnd:Steps

6 When you are done inserting wizard fields, repeat steps 1-2, and in the **Field name** field, type:

TableEnd:Wizards

7 Save the template to any location in DOC or DOCX file format.

Save the template in a shared folder that can be accessed by anyone who needs to use it.

8 Customize the Word document's header and footer, page numbers, logo, and any other customizations of your choice.

3.8.4.3 Microsoft PowerPoint Export Template Creation

Microsoft PowerPoint export templates are PPT or PPTX files used to generate, structure and format exported wizard slideshows.

You can use the templates provided with your Leo installation, or you can create your own custom templates for wizard export in Microsoft PowerPoint. The following sections describe how to create a PowerPoint export template for wizards.

3.8.4.3.1 Microsoft PowerPoint Template Fields, Structure and Formatting

In PowerPoint, wizard elements are inserted into predefined text box fields in the template. All text box fields are described in Table 13.

In order to successfully export the wizard to a slideshow, these text box fields must be structured according to the hierarchy described in Figure 163.

The formatting used for each field in the template is reflected in the export file.

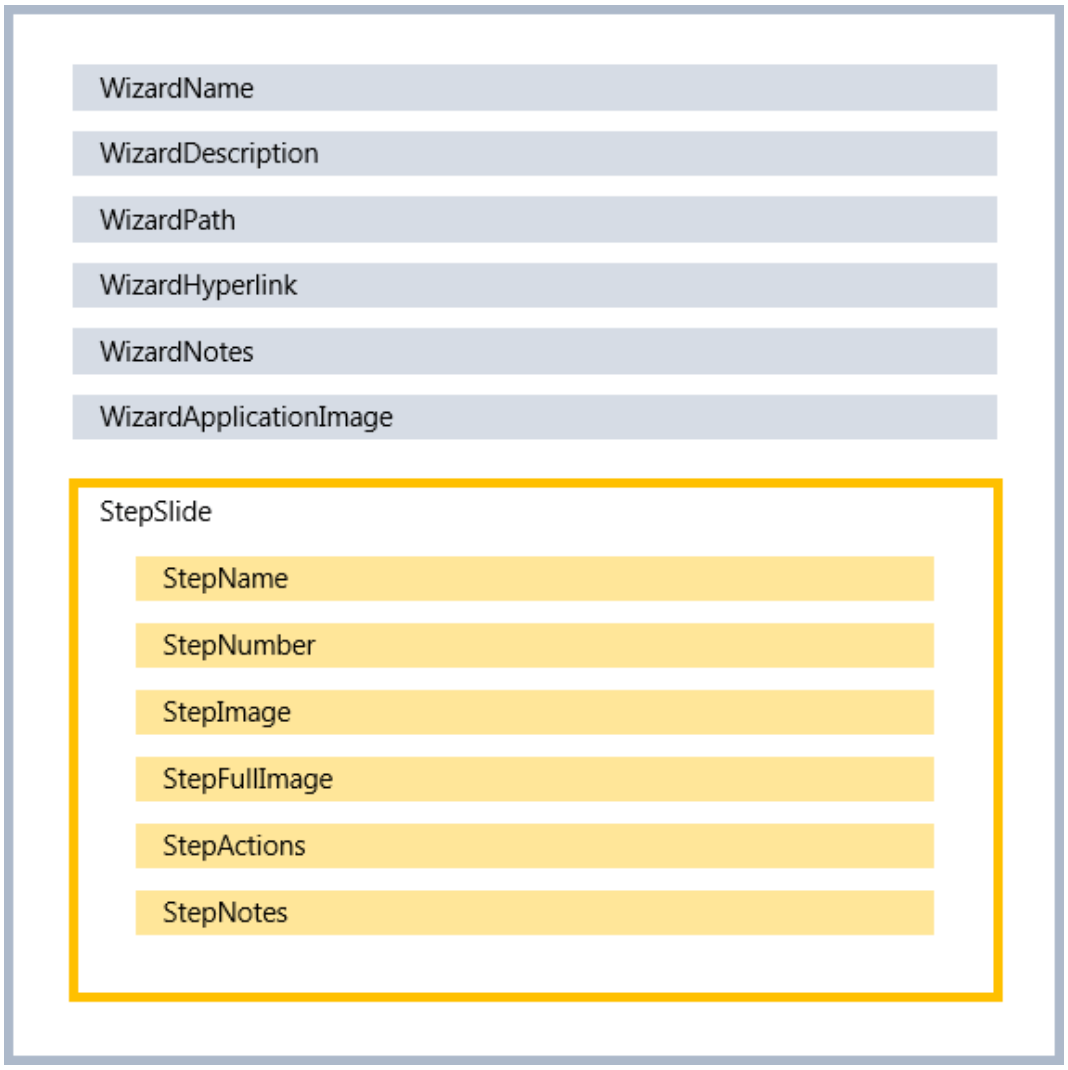
Table 13: PowerPoint Export Template – Field Descriptions

Field Name	Description	Related Information
WizardName	The wizard title	
WizardDescription	A description of what the Leo Wizard does	
WizardPath	A full path to the library, category and subcategories that contain the Leo Wizard in Leo Studio	Section 2.6.1
WizardHyperlink	A unique, direct link to the Leo Wizard that can be pasted or embedded in an email message, Web browser address bar, or the Windows **Run** box	
WizardNotes	Any notes or comments typed in the wizard's **Notes** tab in Leo Studio	
WizardApplicationImage	An icon of the application on which the Leo Wizard is played	
StepSlide	A tag that declares the slide from which all wizard steps appear in the exported file in their original sequence. This field only appears in the template. It does not appear in the exported file.	
StepName	The step title assigned by the content author	Section 3.6.2.1.1.1

Field Name	Description	Related Information
StepNumber	The step number assigned automatically in the Wizard Editor	
StepImage	An image of the area that surround the step's click position. The image retains its original proportions regardless of its placeholder size in the slide.	
StepFullImage	The full step image, as it appears in the Wizard Editor **Display** pane. The image retains its original proportions regardless of its placeholder size in the slide.	
StepActions	A description of the mouse click, any key strokes and bubble text (including buttons) in the order in which they appear in a step	Section 0
StepNotes	Any notes or comments typed in the step's **Notes** tab	

Figure 163 shows the text box hierarchy you must use in your PowerPoint export template:

Figure 159: PowerPoint Export Template – Text Box Hierarchy



3.8.4.3.2 Creating a Microsoft PowerPoint Export Template

Create a Microsoft PowerPoint export file template by doing the following:

1. In the PowerPoint presentation you are using for your template, select the slide that will be the first to contain wizard steps.
2. In the slide you selected in in step 1, insert a text box and type:

   StepSlide

Because the StepSlide text box will not appear in the exported file, you can locate it anywhere in the slide that you selected. However, it must be located in the slide in which you want the first wizard step to appear.

3. Insert a text box and type the field name for the wizard item that you want to insert, according to the field names in Table 13.

Each field name must be created in its own text box, with nothing else in the text box other than that field name.

4. Repeat step 3 for every wizard property, step and action item that you want to insert.

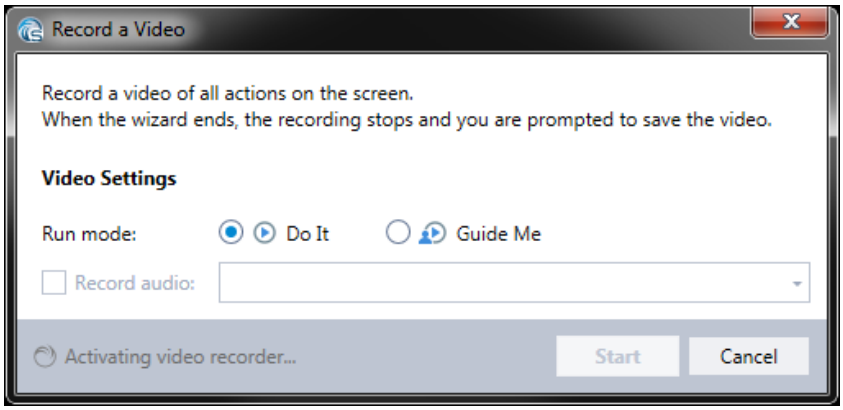
3.8.4.4 Recording a Wizard Video

Leo Studio enables you to record a video of a Leo Wizard as it plays. You can then use this video for training and demonstration purposes. Leo enables you to crop your video and add a watermark image.

Record a video of a Leo Wizard by doing the following:

1 In the Wizard Editor, click **Tools > Record a Video**.
2 The **Record a Video** dialog box appears (Figure 164), including the **Activating video recorder** progress notification.

Figure 160: Record a Video Dialog Box



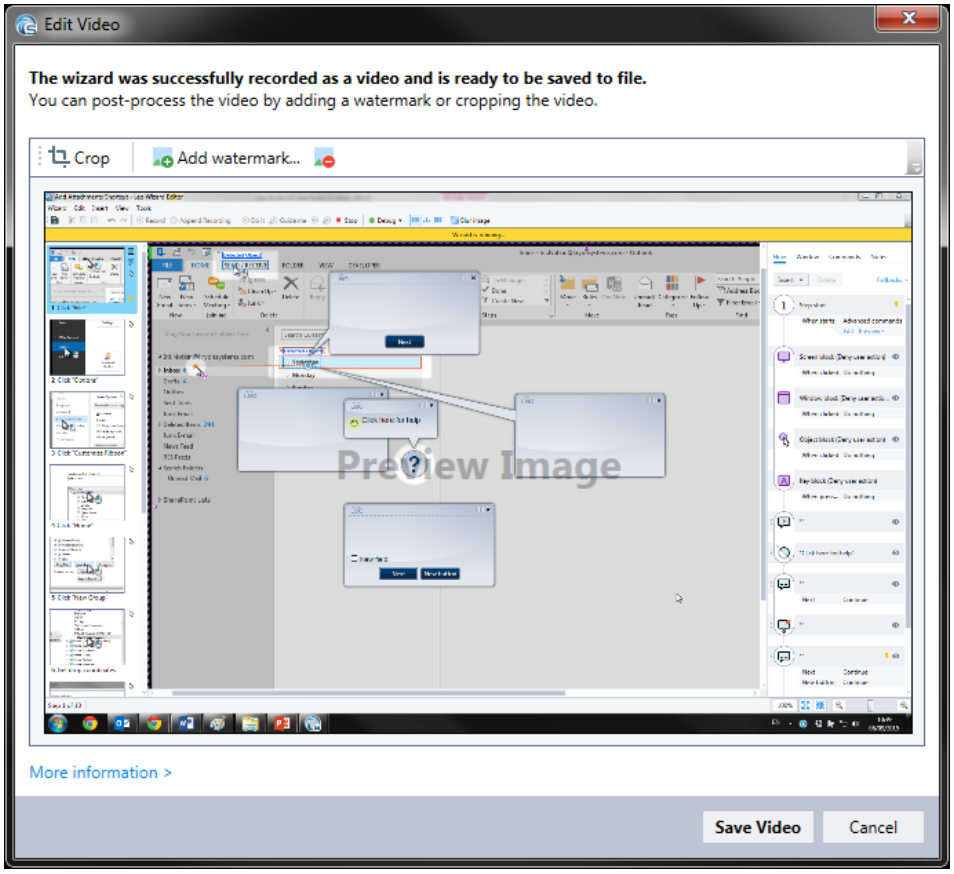
3 In the **Record a Video** dialog box, select the run mode in which you want to record the Leo Wizard.
4 To record audio for the wizard video, select the **Record audio** checkbox and from the dropdown list, select the input audio device that you want to use.
5 Click **Start** to begin recording the video.
The Leo Wizard begins to play and the video recording begins.
6 To stop the recording, do one of the following:

- Wait for the Leo Wizard to end and for the **Edit Video** dialog box to appear (Figure 165).
- Stop the recording before the Leo Wizard ends by clicking the **Leo** icon on the taskbar and then clicking **Quit**.

The **Edit Video** dialog box appears, containing a preview image of the recorded wizard video and additional information about the recording (Figure 165).

The encoding information becomes available after you save the video (see step 9).

Figure 161: Edit Video Dialog Box



7 In the **Record a Video** dialog box (Figure 164), do one or all of the following:

- To crop the video so that only a specific frame appears when the video is played, do the following:
  - i Click **Crop** and drag the crop handles over the preview image to crop the video.
  - ii When you are done, click **Apply Crop**.

To cancel the cropping, click **Cancel Crop**.

- To add a watermark image to the wizard video, do the following:
  - i Click **Add Watermark**.
  - ii From the **Open** dialog box that appears, browse for the image that you want to use as a watermark, and click **Open**.
  - iii In the **Edit Video** dialog box, the image you selected appears at the bottom right corner of the video's preview image.

To reposition the watermark image in the video, drag and drop it around the preview image. To remove the watermark, click the **Remove Watermark** icon.

8 Click **Save Video** to save the wizard video**.**

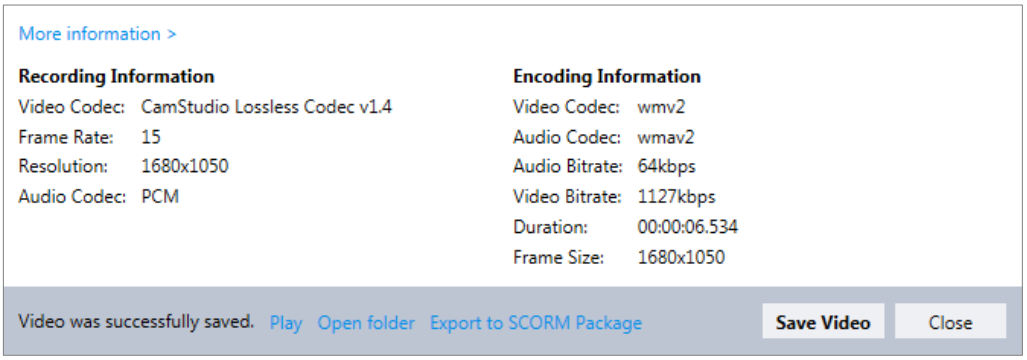
9 In the **Save As** dialog box, select the format in which to save your video:

- ASF
- Flash
- MP4

10 Select the location for your saved video and click **Save**.
The **Creating Video** progress notification appears. When the video is ready, the following information appears (Figure 166):

- A confirmation message including links to playing your video or opening its containing folder.
- The video's encoding information, in the **Edit Video** dialog box's **More Information** area.

Figure 162: Saved Video Information



The wizard video is saved to the location you selected.

Repeat steps 8-9 every time you want to save any additional cropping or watermark changes to your video.

3.8.4.5 Exporting Wizard Images

Leo Studio enables you to export all the images of a Leo Wizard into a destination folder. You can use these images in your existing training or demo materials, such as presentations. Images can be exported either as the thumbnails that appear in the wizard's **Navigation** pane, or as the full images that appear in the **Display** pane. The images are exported in JPEG format and saved to a folder named after the wizard whose images you are exporting.

Export the images of a wizard by doing the following:

1. In the Wizard Editor, click **Tools > Export wizard images...**.
2. In the **Export Wizard Images** dialog box, select the image type to export.
3. From the **Folder** field, browse to a destination folder in which to save the exported images.
4. Click **Export**.
5. The images are exported to an automatically generated folder named after the wizard. The folder is saved in the destination you selected.

3.8.5 Duplicating a Wizard

Duplicate any wizard by doing the following:

1. From the Leo Catalog, right-click the wizard and select **Duplicate**.
   The wizard is duplicated in the same folder, along with all its properties and recorded data.

3.8.6 Viewing Wizard Count by Status

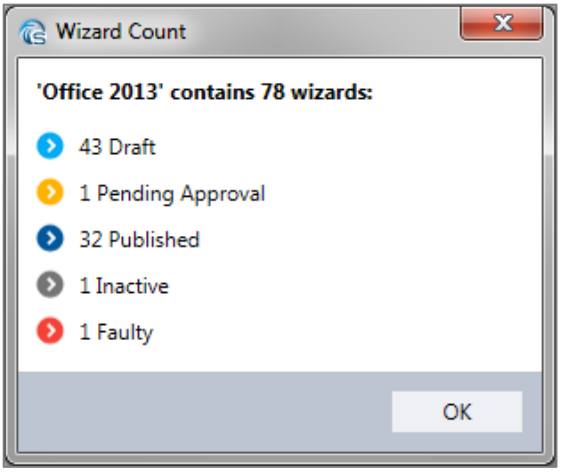
Leo enables you to count the number of Leo Wizards in a library or category, displaying the total number of wizards and the number of wizards per status.

Count Leo Wizards by doing the following:

1. From the **Catalog** pane, select the library or category in which you want to count Leo Wizards.
2. Do one of the following:

- From the Leo Studio menu bar, select **Edit > Wizard Count**.
- Right-click the library or category and select **Wizard Count**.
- The **Wizard Count** dialog box appears (Figure 167), displaying the number of wizards for the selected library or category, as well as the number of wizards per status.

Figure 163: Wizard Count Dialog Box



3.8.7 Searching Leo Wizards

You can perform two types of Leo Studio search

- **Simple**: Search wizards by title.
- **Advanced**: Search wizards by any of the following parameters:
  - ID
  - Name
  - Description
  - Keywords
  - Status
  - Run Mode
  - Is Blank
  - Date Create
  - Last Modified

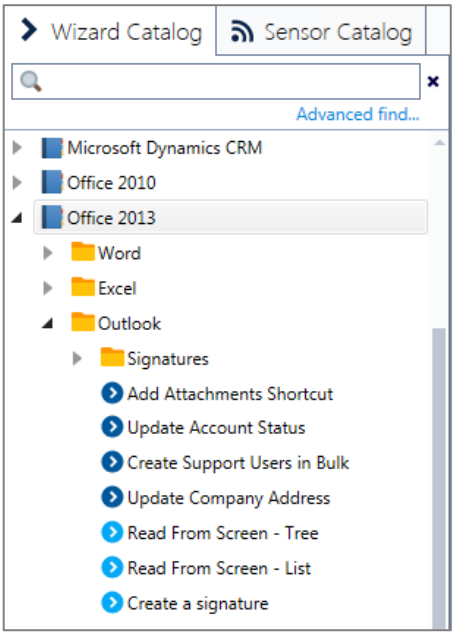
Search for a wizard by doing the following:

1 Do one of the following:

- From the Leo Studio menu bar, select **Edit > Find**.
- Press **CTRL+F**.

A search box appears at the top of the **Catalog** pane (Figure 169).

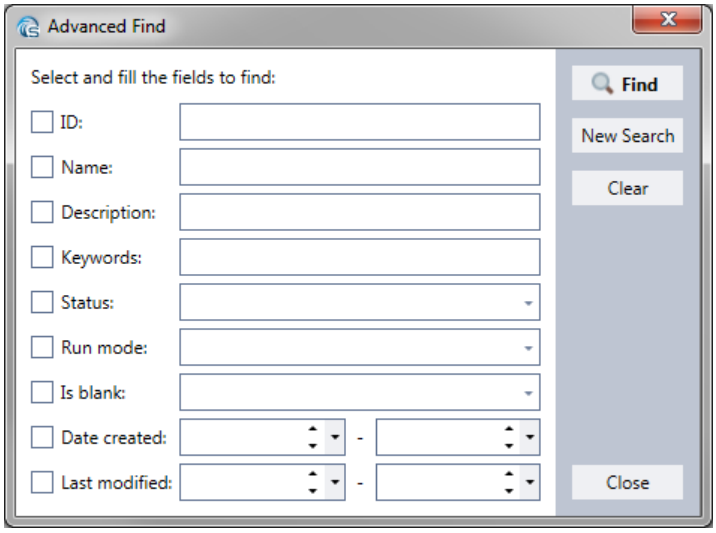
Figure 164: Leo Catalog Search Box



2 Do one of the following:

- In the search box, type a word that appears in the name of the wizard you are searching for, and press **ENTER**.
- Click **Advanced Find**. In the **Advanced Find** dialog box (Figure 169), enter parameters by which to search, and click **Find**.

  Figure 165: Advanced Find Dialog Box

  The search results are displayed in the **Catalog** pane.

3 To display the entire catalog again, close the search box by clicking the **X to its right**.

3.8.8 Refreshing the Leo Catalog

If other content authors are working in Leo Studio on the same server as you, you can reload the Leo Catalog to load any changes they make to Leo Wizards, categories or libraries.

Reload the Leo Catalog by doing the following:

1 From the Leo Studio menu bar, select **Catalog > Reload**.

2 The catalog is reloaded with the latest changes.

4 Sensor Development

The sensor capability requires purchase of the Leo Sensors System License.

Leo Sensors push content to end users just when they need it, at the right place and time. Sensors are used to push information to users or trigger a sequence of actions that users need to know, when those users are not aware that they need to know this information.

Leo Sensors work behind the scenes to detect when the user has launched a specific application or reached a specific screen, and then trigger a context-sensitive action on the user's computer. The action can be as simple as a notification bubble, or as complex as validating fields from the screen and triggering a logic of steps, conditions and loops. Sensors are useful when you need to inform users of important information, to ensure that processes are done correctly, to enhance features of your application or to prompt users to perform a specific action.

Sensors are created and managed in the Sensor Catalog in Leo Studio, and are run in Leo Player. The Leo Sensor Catalog enables you to determine what action the sensor will trigger, and when the action will be triggered.

To run and test sensors, Leo Player must be installed and running on your computer.

The following sections describe the Leo Studio purpose of recording and editing Leo Sensors in the Wizard Editor window.

For information about the Leo Sensor components and properties, see Section 4.1.

For information about managing Leo Sensors in the Sensor Catalog, see Section 4.2.

4.1 Sensor Components and Properties

A Leo Sensor consists of the following components:

- **Sensor**: The Leo mechanism that periodically scans the Leo user's active application and window, and determines when to display a message accordingly. The general sensor properties are described in Section 4.1.1.
- **Trigger**: The event on the Leo user's computer that causes a sensor to launch a sequence of steps or display a notification. The trigger properties are described in Section 4.1.2.
- **Action**: The notification message that appears on the Leo user's computer when a sensor trigger has been activated. The action properties are described in Section 4.1.2.1.

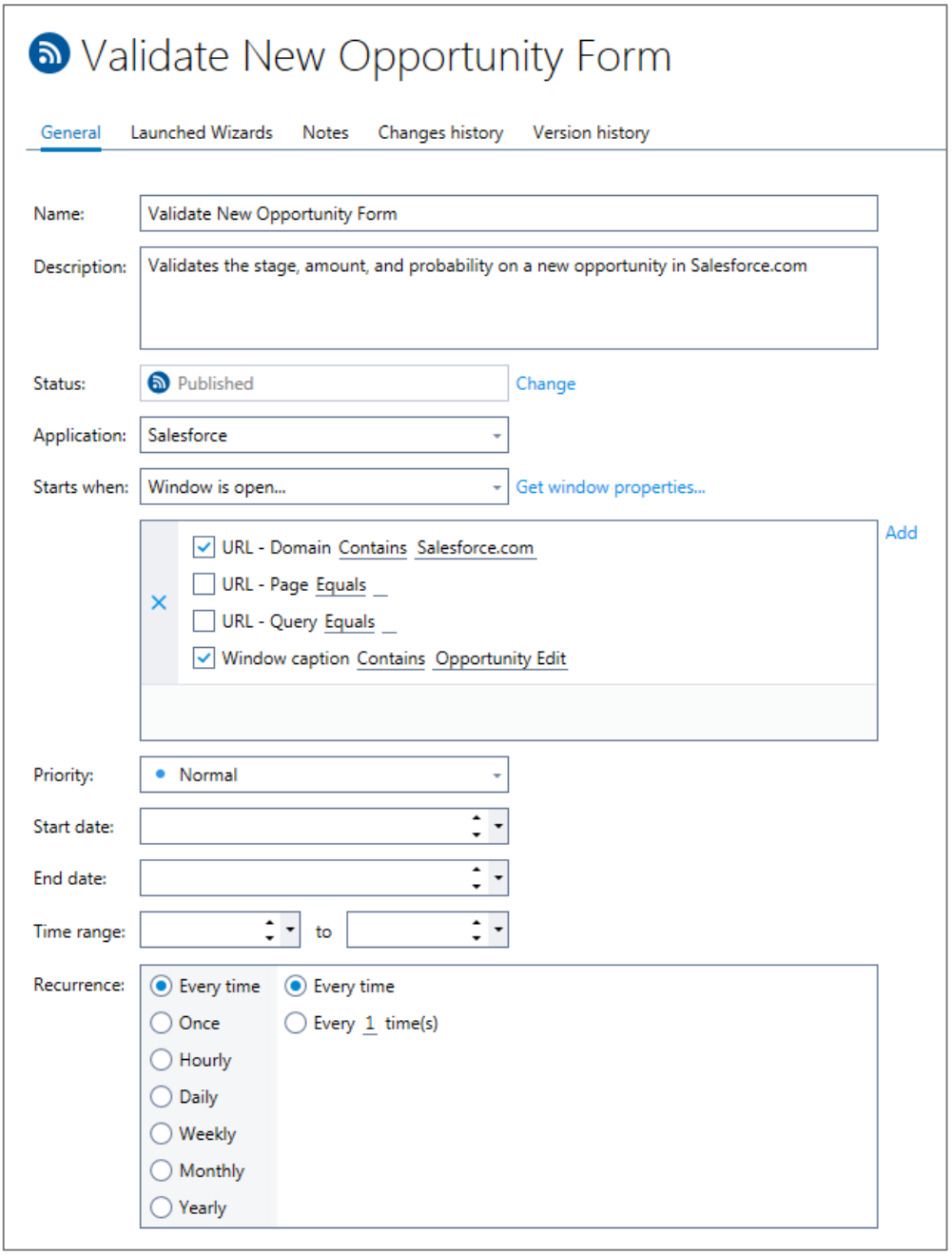
For information about how to edit the sensor components and properties, see Section 4.2.1.

4.1.1 Sensor Mechanism

The sensor mechanism periodically scans the **Leo** user's active application and window, and determines when to trigger an action accordingly. The general sensor properties determine whether the sensor is published, how it is labeled, and when it starts and ends. These are properties that characterize the sensor itself rather than its content. Properties pertaining to the content of the sensor are determined in the trigger and action properties, described in Sections 4.1.2 and 4.1.2.1, respectively.

Figure 170 shows the main sensor properties in the sensor's **General** tab.

Figure 166: Sensor General Tab (Multi-Step Sensor Example)



The general sensor properties are:

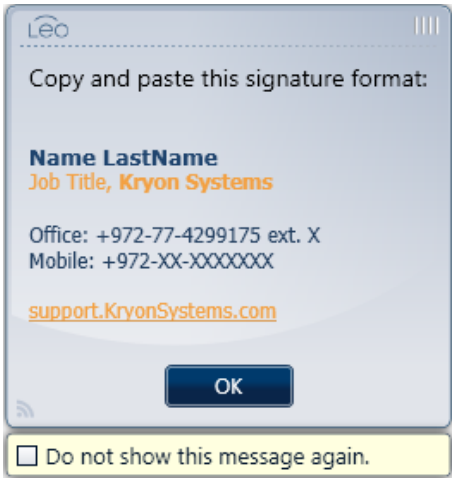
- **Name**: A descriptive title for the sensor, for reference to aid in content management.
- **Description**: A description of what the sensor does, for reference to aid in content management.
- **Status**: A stage in the sensor's publishing process. The sensor status types are:
  - **Draft**: The sensor is incomplete, and needs to be further edited and tested. It is not activated for any Leo users in your company. This is the default status of a new sensor.
  - **Pending Approval**: The sensor is edited and tested, and is pending approval before it can be published. It is not activated for any Leo users in your company.
  - **Published**: The sensor is completed and approved, and is activated for all Leo users in your company.

⚠ Only sensors in Published status can be run and tested in Leo Player.

Any changes to a published sensor take approximately 15 minutes to get updated in Leo Player. This time varies depending on your configuration. To view the changes immediately, clear the sensor cache from the Leo **Options > Advanced** tab.

  - **Inactive**: The sensor should not be published, and is not activated for any Leo users in your company.
  - **Faulty**: The sensor does not work properly, and is not activated for any Leo users in your company.
- **Priority**: The precedence of a sensor over other sensors, determined by the following logic:
  a. High: The sensor takes precedence over any sensors that share similar trigger settings but whose priority is set to Normal or Low.
  b. Normal: The sensor takes precedence over any sensors that share similar trigger settings but whose priority is set to Low.
  c. Low: The sensor takes no precedence over sensors that share similar trigger settings and whose priority is set to Normal or High.
- **Start Date**: The beginning of a date range during which the published sensor is active.
- **End Date**: The end of a date range during which the published sensor is active.
- **Time Range**: The beginning and end of a time range during which the published sensor is active.
- **Recurrence**: The frequency at which the sensor action is triggered when the trigger settings are met. The frequency types are:
  - Once: The action is performed only the first time the trigger is detected.
  - Every time: The action is performed every time an instance of the trigger, that is, an instance of the defined application or window, is detected by the sensor. For example, if two different PowerPoint presentations are open on the user's computer, each of the presentations is an instance of the PowerPoint application. When the sensor is set to act once per application instance, it triggers an action for each of the presentations separately.
  - Hourly/Daily/Weekly/Monthly/Yearly: The action is performed periodically at regular predefined time intervals.
  - End After X Occurrences: The action is performed for the first predefined number of times that the trigger is detected, and then deactivated.
  - Disable sensor when Wizard Runs: Automatically deactivates the sensor if an actively running wizard is detected.
  - Allow the User to Dismiss: Provides users with a message that allows them to dismiss the sensor (Figure 171):

Figure 167: Sensor Dismiss Message



None of the general sensor properties are visible to end users in Leo Player.

For information about how to access and edit the general sensor properties, see Section 4.2.1.

4.1.2 Sensor Trigger

A trigger is the event on the user's computer that causes a sensor to perform a predefined action. The sensor's trigger properties determine what causes the sensor to perform the action.

For information about sensor actions, see Section 4.1.2.1.

The sensor trigger properties are located in the sensor's **General** tab (see Figure 170).

The sensor trigger properties are:

- **Trigger Type (Starts when)**: Determines what event on the user's computer triggers an action. The trigger types are:
  - Open application: When a predefined type of application is launched on the user's computer, the sensor action is triggered.
  - Open window: When a predefined type of window is currently selected on the user's computer, the sensor action is triggered.
- **Trigger Settings (Window)**: Determine which window or application the sensor must detect to trigger the action. Trigger settings are enabled depending on the trigger type selected. The trigger settings are:
  - Application: The application in which the sensor must trigger the action. The **Application** dropdown list lists all the applications defined for the sensor library. This setting is relevant for both window and application triggers.
  - Window Detection Properties: Requires you to enter window properties that the sensor uses to identify the relevant window in which to trigger the action (see Figure 170). This setting is relevant for window triggers only.
  - The window detection properties can be set either by typing them manually or by retrieving them automatically from an open window.

For information about how to set the window detection properties automatically, see Section 4.1.2.1.

For information about how to set them manually, see Section 4.1.2.2.

The relevant Web or desktop properties appear automatically for the selected application.

For information about window detection properties for desktop applications, see Section 3.1.1.1.

For information about window detection properties for Web applications, see Section 3.1.1.2.

For information about how to edit the trigger sensor properties, see Section 4.2.1.

4.1.2.1 Setting the Trigger's Window Detection Properties Automatically

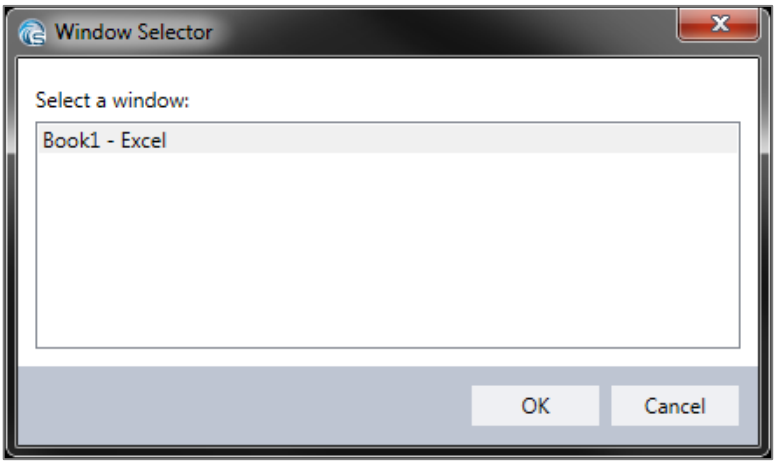
The window detection properties of a window can be set automatically by retrieving them from an open window of the relevant application.

For a description of the window detection properties of sensor triggers, see Section 4.1.2.

Retrieve the window detection properties automatically by doing the following:

1. Open the relevant sensor for editing, as described in Section 4.2.1.
2. From the **Sensor** dialog box, click the **General** tab (see Figure 170).
3. From the **Application** dialog box, select the application in which the sensor must trigger the action.
4. From the **Starts when** dialog box, select **Window is open**.
5. Make sure that the relevant application window, whose properties you want to retrieve, is open on your computer.
6. Click **Get properties from an open window**.
   The **Window Selector** dialog box appears (Figure 172).

   Figure 168: Window Selector



7. From the **Window Selector** dialog box, select the window that you want to retrieve properties from, and click **OK**.
   The full window detection properties appear in the relevant fields.
8. Edit the conditions for each property as needed, to apply the properties to the broadest possible variety of user-specific windows.
9. When you are done editing the window detection properties, click **Save Changes**.
   Your window detection changes are saved.

4.1.2.2 Setting the Trigger's Window Detection Properties Manually

Manually setting the window detection properties of a sensor trigger requires you to know these properties in advance. If you do not know these properties, you can retrieve them automatically as described in Section 4.1.2.1.

For a description of the window detection properties of sensor triggers, see Section 4.1.2.

Set window properties manually by doing the following:

1. Open the relevant sensor for editing, as described in Section 4.2.1.
2. From the **Sensor** dialog box, click the **General** tab (see Figure 170).
3. From the **Application** dialog box, select the application in which the sensor must trigger the action.
4. From the **Starts when** dialog box, select **Window is open**.
5. Make sure that the relevant application window, whose properties you want to retrieve, is open on your computer.
6. Click the **Add** icon in the detection properties area.
7. Type the properties into the relevant fields.
8. Edit the conditions for each property as needed, to apply the properties to the broadest possible variety of user-specific windows.
9. When you are done editing the window detection properties, click **Save Changes**. Your window detection changes are saved.

4.1.3 Sensor Action

A sensor action is the event or sequence of actions that takes place on the user's computer when a sensor trigger has been activated. The sensor's action properties determine what kind of action Leo must perform when the sensor detects the relevant trigger.

There are two types of sensors:

- Simple Sensor: A simple bubble notification displayed on the user's screen. For information about how to edit the sensor action and other properties, see Section 4.2.2.
- Multi-Step Sensor: A variety of customizable actions that are either behind the scenes or visible to end users in Leo Player, such as field validation, wizard launch, loops, decision points and conditions, or conditional bubbles. For information about how to edit the sensor action and other properties, see Section 4.2.3.

4.1.3.1 Simple Sensor

A simple sensor is used to provide an informative message to the user, in context at the moment of need. A bubble notification is displayed on the user's screen when a specific application or window is detected by a Leo sensor.

- **Button Caption**: The text on the bubble button.
- **Position**: Alignment of the bubble on the user's screen.
- **Color**: The bibble's background color.
- **Options**: Duration of the message display on the user's screen:
  - Auto-Click Button: A few seconds after the bubble appears on the user's screen, Leo automatically clicks the bubble button. The number of seconds is configurable.
  - Lock Button: Leo disables the bubble button for a configurable number of seconds. The user cannot click the bubble button during that time.

- Block Screen: The user's entire screen is blocked for as long as the bubble is displayed. The user cannot click anything on the screen around the bubble.

- **If Window Is Closed**: Determines whether to display the bubble when the window that triggered it is closed.

For information about how to edit the simple sensor action, see Section 4.2.2.

4.1.3.2 Multi-Step Sensor

A multi-step sensor is used to ensure that processes are done correctly, to enhance features in your application or to inform or prompt users to perform a specific action.

Multi-step sensors are a wizard-like sequence of actions on the user's computer, triggered by a variety of customizable actions that are either behind the scenes or visible to end users in Leo Player, such as field validation, wizard launch, loops, decision points and conditions, or conditional bubbles.

Multi-step sensors can be triggered by a specific GUI object in a window, or by a user click on a specific object such as a button.

Multi-step sensors are recorded and managed in a similar way to wizards.

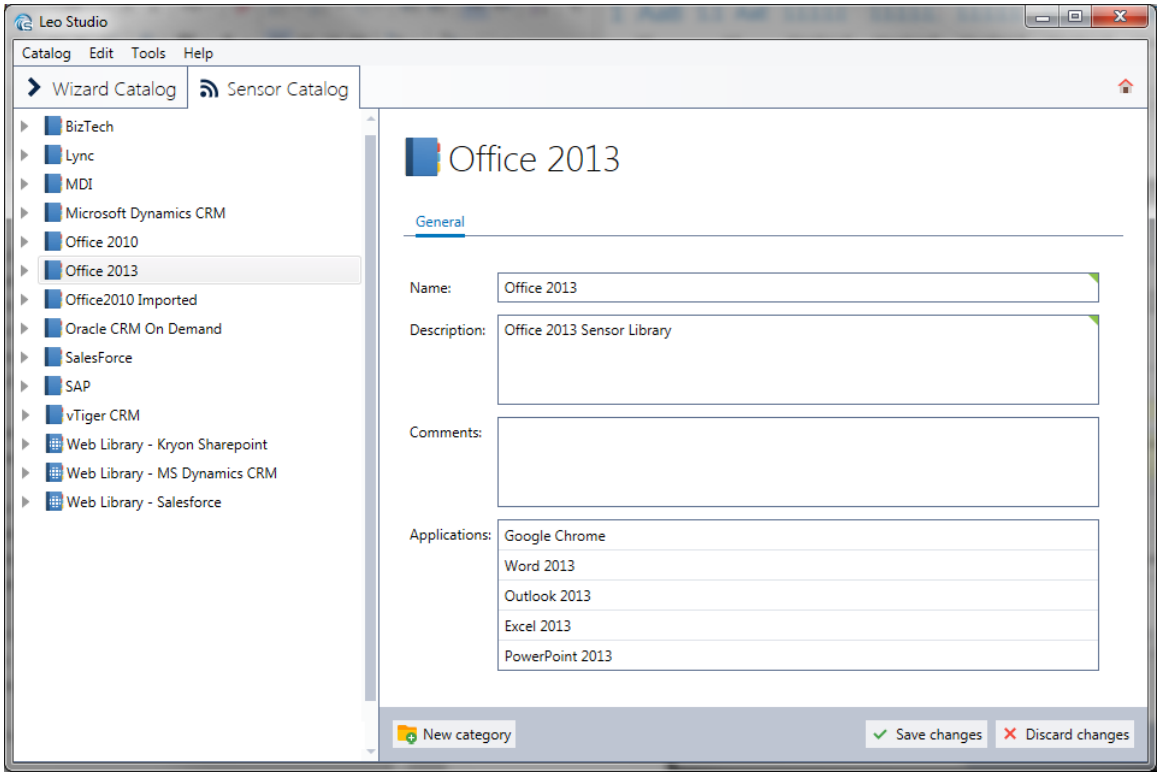
For information about how to edit the multi-step sensor action, see Section 4.2.3.

4.2 Managing Leo Sensors

All the Leo Sensors available to you and your users are listed in the Sensor Catalog. The Sensor catalog allows you to add, edit, and delete sensors.

Figure 173 shows the **Sensor** Catalog.

Figure 169: Sensor Catalog



The Sensor Catalog contains the same libraries as the Wizard Catalog, and is managed in the same way. The categories in the Sensor Catalog are managed separately from those in the Wizard Catalog. The sensor catalog displays only categories created specifically in it, and any existing sensors. This catalog does not display any wizards or wizard categories.

For information about how to add a new sensor, see Section 2.6.2.1.

For information about how to record a multi-step sensor, see Section 3.5.

For information about how to edit the general sensor properties, see Section 4.2.1.

For information about how to edit the sensor action properties, see Section 4.2.2.

For information about how to launch a wizard from a multi-step sensor, see Section 4.2.3.

4.2.1 Editing the General Sensor Properties

Edit the general sensor properties by doing the following:

1. From Leo Studio, in the **Sensor Catalog** tab, select the sensor whose properties you want to edit.
2. In the **General** tab, type a sensor name.

None of the fields in the sensor's **General** tab, including the name and description, are visible to end users in Leo Player.

3. In the **Description** field, type the sensor description.
4. Set the sensor mechanism and trigger properties in the relevant fields. For a description of these properties, see Sections 4.1.1 and 4.1.2.
5. Click **Save Changes** to save the sensor.

To cancel your changes, click **Discard**.

6. Edit the sensor action properties, as described in Section 4.2.2.

4.2.2 Editing the Action of a Simple Sensor

Edit the prompt of a single-prompt sensor by doing the following:

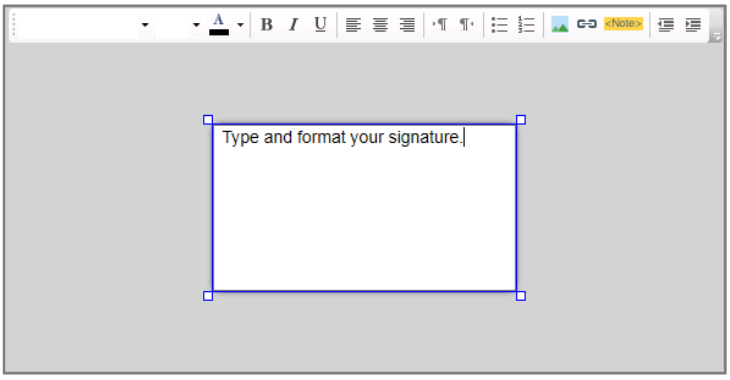
1. From Leo Studio, in the **Sensor Catalog** tab, select the simple sensor whose prompt you want to edit.
2. Click the sensor's **Bubble** tab.
3. Double-click the sensor bubble to edit its content.

   The **Edit Sensor** button is disabled.

Use the **Bubble** toolbar to format the your text and add images and hyperlinks.

To resize the prompt, click and drag its borders.

Figure 170: Editing the Sensor Prompt (with Prompt Toolbar)



i. Set the sensor's prompt properties in the relevant fields. For a description of these properties, see Section 4.1.3.1.

ii. Click **Save Changes** to save the updated sensor.

To cancel your changes, click **Discard**.

4.2.3 Editing the Action of a Multi-Step Sensor

1 Click **Edit Sensor**.
2 Record or edit the sensor steps as you would a wizard, as described in Section 3.5.

For more information about editing a sensor step, see Section 4.3.

For more information about the purpose of multi-step sensors, see Section 4.1.3.2.

4.2.4 Launching a Wizard from within a Sensor

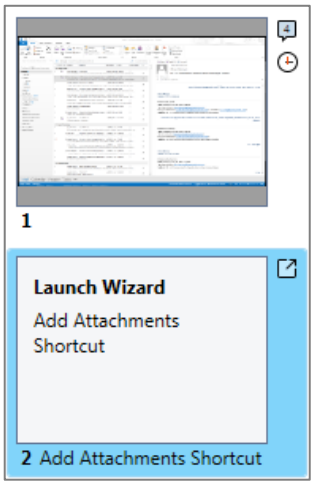
Launch a wizard from within a sensor by doing the following:

1 Open the Leo Sensor from which you want to launch a wizard.
2 In the sensor's **Navigation** pane, select the step after which you want to launch the wizard.
3 On the Wizard Editor menu bar, select **Insert > Launch Wizard**.
4 The **Catalog** dialog box appears (Figure 27).
5 From the **Catalog** dialog box, select the Leo Wizard that you want to embed.

To search for a specific Leo Wizard, click **Find** or press **CTRL+F** and then type your search query.

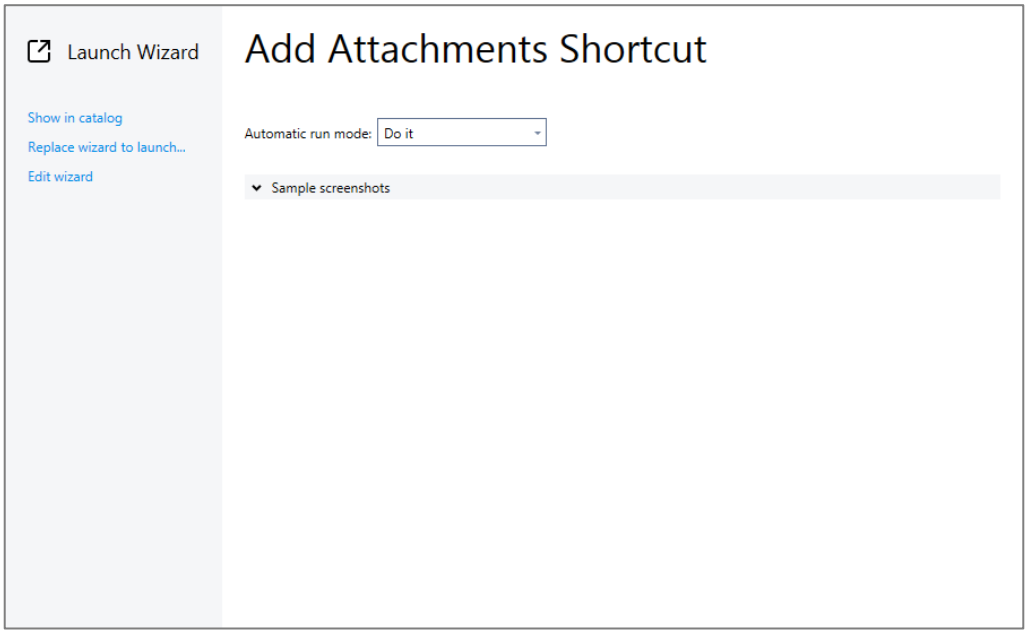
The Leo Wizard you selected is inserted as a step into the sensor flow. The embedded wizard's thumbnail appears in the Wizard Editor's **Navigation** pane (Figure 175).

Figure 171: Launched Wizard Thumbnail in the Navigation Pane



6 In the Launched Wizard step's **Display** pane, from the **Automatic Run Mode** dropdown list, select the run mode in which to run the launched wizard.
7 The launched wizard's control pane appears in the Sensor Editor's **Display** pane (Figure 176).

Figure 172: Launched Wizard Control Pane in the Display Pane



To preview the steps of the embedded wizard, click the **Sample screenshots** dropdown list.

4.2.5 Resetting Sensor Usage Statistics

The sensor reset feature allows you to reset the following sensor statistics per user in the database:

- Number of times the sensor was triggered
- Last sensor run time
- Number of times the trigger window/application was detected.
- Did the user select the sensor's "Do not show this again" checkbox.
- Last usage reset time

This is useful if you change, for example, the sensor's recurrence or time range, because it resets the usage statistics so that they can be tracked from scratch according to the updated settings.

For more information about sensor settings, see Section 4.1.

Reset sensor usage statistics by doing the following:

1 From the Sensor catalog, right-click the sensor for which to reset usage statistics.
2 Click **Reset Sensor Usage**.
The sensor usage statistics are reset in the database.

4.3 Editing a Sensor Step

When you record a Leo Sensor, Leo captures the clicked objects and keystrokes you recorded in application windows. Review the settings in each step and edit them as needed to ensure that this information is generic so that the step can be run on any computer, and to ensure that Leo knows what to do if it cannot run the step.

Editing sensor steps is identical to editing wizard steps. For more information about editing sensor steps, see Section 3.6.2.

4.4 Debugging Sensors

Debug mode runs the sensor while visually indicating its ending point, for testing purposes. This helps you identify incorrect sensor behavior and to understand whether it ended as it should. When you run a sensor from Studio in debug mode, it indicates the ending point of the sensor, and which type of ending it is:

- Sensor failed
- Sensor ended successfully
- Sensor ended unsuccessfully

Run a wizard in debug mode by doing the following:

1. In the **Sensor Editor** window, select the step from which to start running the wizard in debug mode.
2. On the Sensor Editor menu bar, click **Debug**.
3. Go to the window that triggers the sensor.
   The sensor begins to run from the step you selected, showing the following actions:
   - Sensor failed
   - Sensor ended successfully
   - Sensor ended unsuccessfully

4.5 Testing Sensors

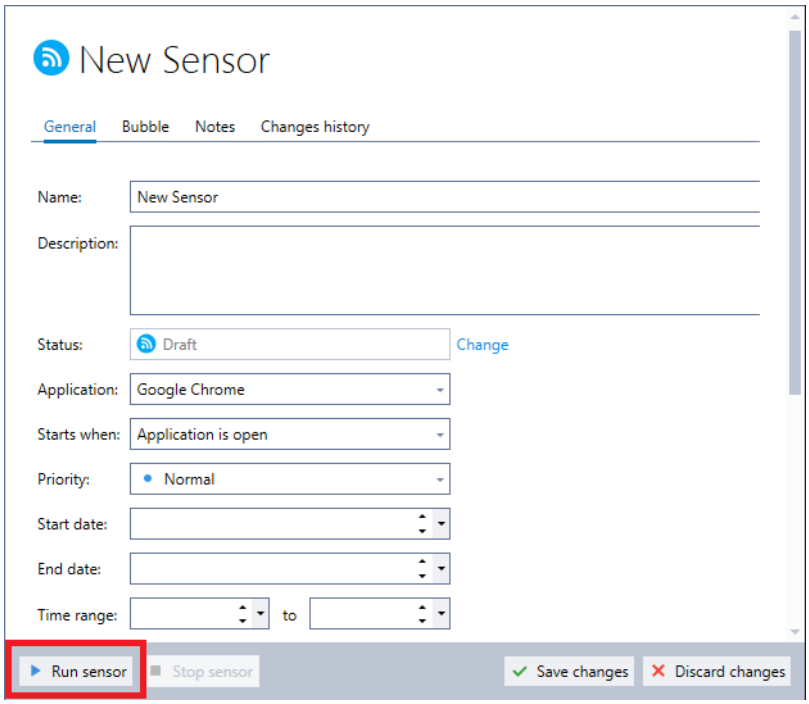
Both simple and multi-step sensors can be run and tested from Leo Studio and Leo Player.s

4.5.1 Testing a Simple Sensor in Leo Studio

Test a simple sensor from Leo Studio by doing the following:

1. Open the application window that launches the sensor.
2. From the Leo Studio catalog, select the simple sensor to test, and click **Run Sensor** (Figure 177).

Figure 173: Run Sensor (Single Prompt)



To stop testing the sensor, click **Stop Sensor**.

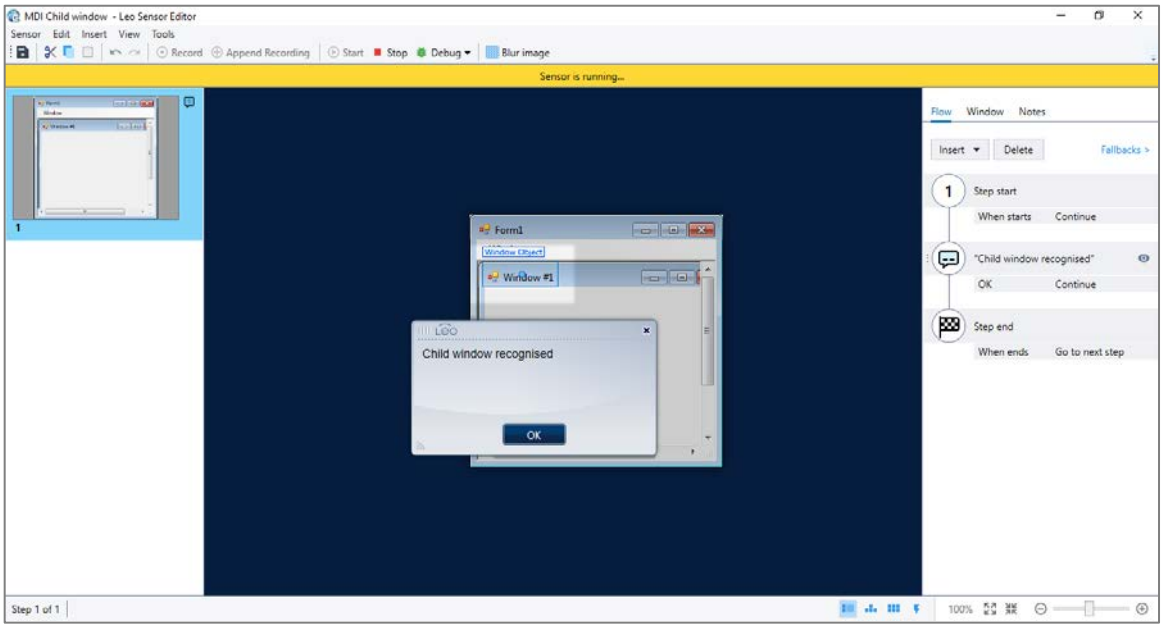
4.5.2 Testing a Multi-Step Sensor

When both Studio and Player are open, avoid testing from Studio a sensor that is already published in Player. To test a published sensor from Studio, make sure Player is closed.

Test a multi-step sensor from Leo Studio by doing the following:

1. Open the application window that launches the sensor.
2. From the Leo Studio catalog, select the multi-step sensor to test, and click **Edit Sensor**.
   The sensor opens up for editing in the Sensor Editor.
3. Run the sensor by clicking **Start**.
   A **Sensor is running** indication appears at the top of the Sensor Editor window (Figure 178).

Figure 174: Sensor Is Running Indication



4. Switch to the window that launches the sensor.
   The multi-step sensor is launched.

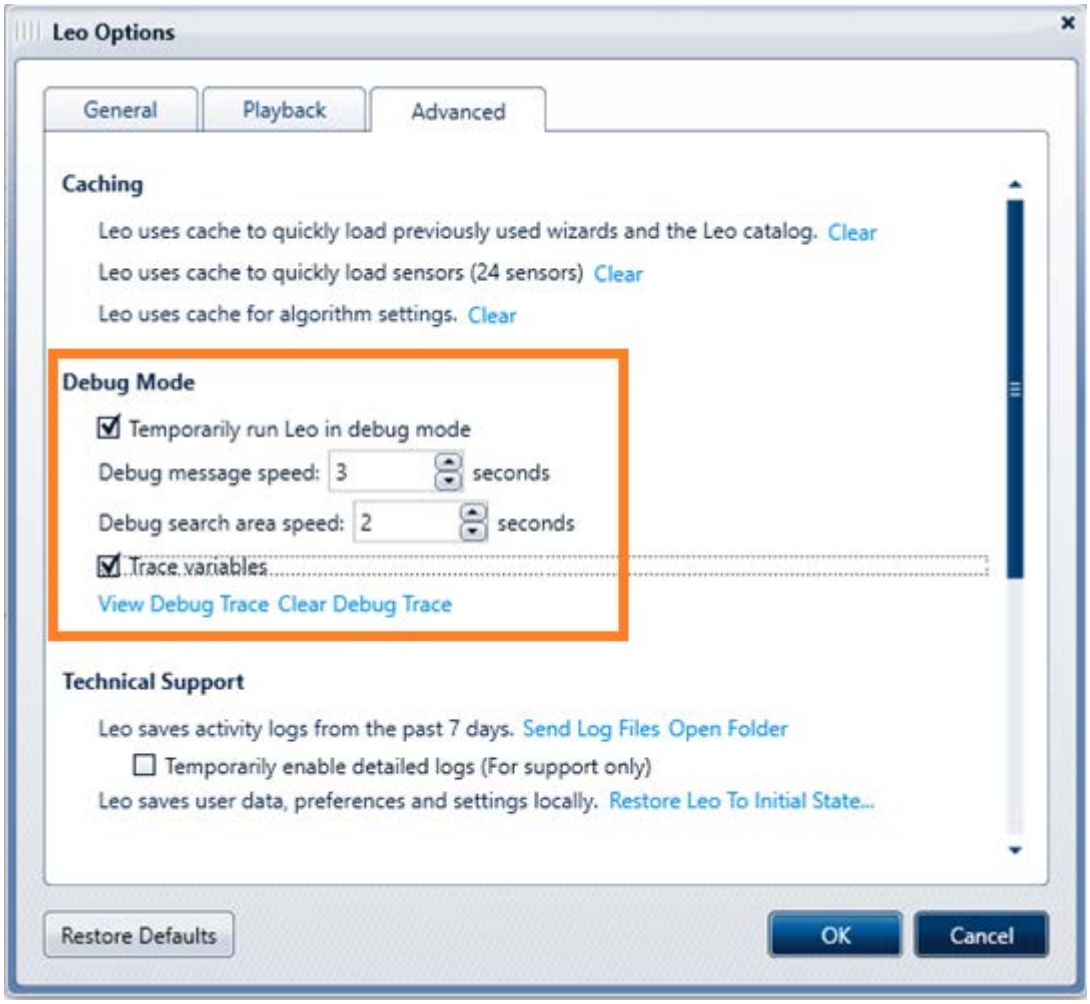
To stop testing the sensor, click **Stop**.

4.5.3 Testing a Sensor in Leo Player

To test a sensor from Leo Player by doing the following:

1. Open Leo Player
2. Open the Options menu
3. For running the current wizard in debug mode, select the option “Temporarily run Leo in debug mode” (see Figure 179 - Leo Player Debug Mode).
4. For creating debug trace for the current wizard execution, select the option “Trace variable” (see Figure 179 - Leo Player Debug Mode).

Figure 175 - Leo Player Debug Mode



5 Advanced Leo Features

5.1 Advanced Commands

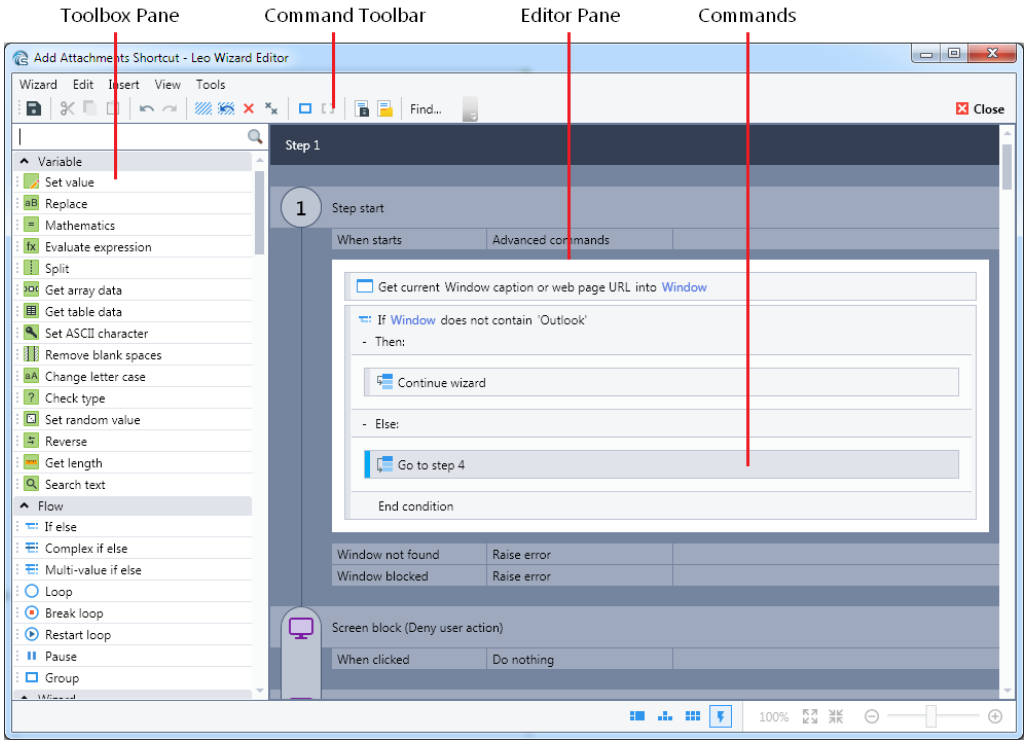
Advanced commands are a Leo programming feature that can be used by content authors with no programming experience. Advanced commands enable you to retrieve information of the system, user and currently running wizard, as well as inserting advanced actions such as reading files, executing various scripts, and using logical flows such as loops and conditions.

Advanced commands can be triggered from specific locations in a Leo Wizard. For more information about the locations from which you can define advanced commands, see Section 5.1.1.

Some of the advanced commands are made up of a variable (or multiple variables) and a logical action performed on that variable. For more information about variables, see Section 5.1.2.

Each command is an instruction to the Leo Wizard. These commands affect the flow of the wizard, the flow of other commands, and more. For more information about the types of advanced commands, see Section 5.1.3.

Figure 176: Advanced Command View



5.1.1 Triggers for Advanced Commands

Advanced commands can be triggered from the following places in a Leo Wizard:

- **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. For more information about editing fallbacks, see Section 3.3.2.3.2.
- **Step Start**: Triggers a predefined set of advanced commands when the step begins. The advanced commands are triggered as early as before Leo brings the window to front and activates it.
- **Step End**: Triggers a predefined set of advanced commands when the step is completed.

For information about how to access the Advanced Command Editor from each of these locations, see Section 5.1.4.1.

For information about how to preview advanced commands from each location, see Section 1.1.1.1.

5.1.2 Variables

A variable is a named location that stores a textual or numeric value.

The variable characteristics are:

- **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 !@#$%^&*()/<>-=|\.
- **Value**:
  - Variables do not have to contain a value in order to be used in commands. An undefined variable will simply contain an empty string.
  - Variable values can be changed at any time.
  - The variable value is always treated by Leo as a string.
- **Referencing**: In wizard bubbles and commands, variables can be referred to by wrapping them with dollar ($) 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# = '2'.

- **Variable Comparisons**: 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 select the variable type.

Variables can be used in commands in order to execute a sequence of actions in a wizard, and also in the following places:

- **Bubble Text**: Enables you to insert a variable into the text, so that the text is customized based on the variable's current value. For more information about editing the bubble text, see Section 3.6.2.1.3.15.2.
- **Bubble Button Text**: Enables you to insert a variable into the text, so that the text is customized based on the variable's current value.
- **Bubble Show/Hide Settings**: Displays the bubble **only** if a specific variable, which enforces a condition, returns a "TRUE" value. If the variable returns a "FALSE" value, the bubble is not displayed to the user and the wizard continues as designed without it. For more information about editing bubbles' Show/Hide settings, see Section 3.6.2.1.3.9.

5.1.3 Advanced Command Types

For information about specific advanced commands and how to use them see the *Advanced Commands Reference Guide* available from the Advanced Command View toolbar.

5.1.4 Managing Advanced Commands

Commands are managed from the **Advanced Command Editor**. The Command Editor displays the set of commands defined for every wizard location that uses advanced commands.

5.1.4.1 Accessing the Advanced Command View

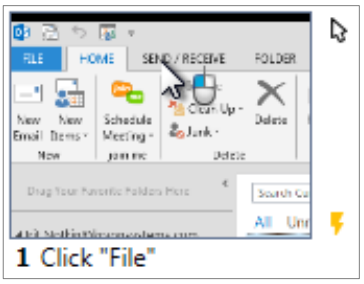
Access the Advanced Command Editor by doing the following:

1 Select the wizard location from which to access the Advanced Commands Editor.

For the list of locations from which you can access the Advanced Commands Editor, see Section 5.1.1.

2 From the wizard location you selected, select **Execute Advanced Commands**.
3 Under the **Execute Advanced Commands** dropdown list, click **Edit**.
4 The **Advanced Command View** appears, and the **Advanced Commands** icon appears in the step's thumbnail (Figure 195).

Figure 177: Advanced Commands Indication in Step Thumbnail



To preview the advanced commands you defined for this location, click **Preview**.

Alternatively, you can click the AC view icon on the wizard's status bar (Figure 196)

Figure 178: View buttons

5.1.4.2 Adding Advanced Commands

Add an advanced command to a wizard location by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 Locate the step, action and event where you want to edit the Advanced Command.
3 Do one of the following:

- Drag and Drop: From the **Command Toolbox** pane, select the command and then drag and drop it to a chosen location in the **Editor** pane.

You can select a command from the command toolbox by either clicking it or navigating to it with the **UP**/**DOWN** arrow keys.

- Double-Click (or Select and **ENTER)**:
  i. In the **Editor** pane, select a command after which you want to add your command.
  ii. From the **Command Toolbox** pane, select the command that you want to add and press **ENTER**.
  iii. The command is added after the selected command in the **Editor** pane.
- Auto-Complete (Type and Select):
  i. In the **Editor** pane, click either a blank area or the command after which you want to add a command.
  ii. Start typing the command name.
  iii. A dropdown list appears, containing command names that match the letters you typed.
  iv. Do one of the following:
    - Press **ENTER** to select the matching command that is displayed.
    - Press the **UP**/**DOWN** arrow keys to navigate to the relevant command, and then press **ENTER** to add it.

For information about the types of commands you can add, see Section 5.1.3.

5.1.4.3 Editing Advanced Commands

Edit the contents of an advanced command by doing the following:

1. Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2. In the Advanced Commands Editor's **Editor** pane, select the command that you want to edit.
3. Double-click the selected command.
4. If the command contains editable content, the command dialog box appears in which you can make your changes.

5.1.4.4 Moving Commands

Leo enables you to move commands in the Advanced Commands Editor to reorder the sequence of commands within a wizard location.

Move commands by doing the following:

1. In the **Editor**, select the command that you want to move.
2. Do one of the following:
   - Drag and drop the command onto its new location in the **Editor** pane.
   - Cut and paste the command as follows:
     i. Cut the command by doing one of the following:
       - On the **Advanced Commands** toolbar, click the **Cut** icon.
       - Press **CTRL+X**.
     ii. In the **Editor** pane, select the command after which you want to paste your command.
     iii. Paste the command by doing one of the following:
       - On the **Advanced Commands** toolbar, click the **Paste** icon.
       - Press **CTRL+V**.

5.1.4.5 Copying Commands

Leo enables you to copy commands within a step or from one step to another, within a wizard or from one wizard to another.

Copy a command by doing the following:

1. In the **Editor**, select the command that you want copy.
2. Do **one** of the following:
   a. Press CTRL+D. The step will be immediately duplicated and the new step will appear just after the original one.
   b. User Copy -> Paste:
      i. On the **Advanced Commands** toolbar, click the **Copy** icon.
      ii. Press **CTRL+C**.
      iii. In the **Editor** pane, select the command after which you want to paste your command.
      iv. Paste the command by either click the Paste icon in the Advanced Commands toolbar or press CTRL+V.

5.1.4.6 Exporting Commands

Advanced Leo commands can be exported by locally saving them to command files in CLEO format.

For information about importing commands from CLEO command files, see Section 5.1.4.7.

Export a command by doing the following:

1. Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2. In the **Editor** pane, select the command pane that you want to export.
3. On the **Advanced Commands** toolbar, click the **Save to File** icon.
   The **Save As** dialog box appears.
4. Select a location for the exported command file.
5. Click **Save** to save the command file in the selected location.
   The exported command is saved to file in CLEO format.

5.1.4.7 Importing Commands

Advanced Leo commands can be imported from locally saved command files in CLEO format.

For information about exporting commands to CLEO command files, see Section 5.1.4.6.

Importing advanced commands can be useful for reusing them in a different location of the wizard or for use in another wizard.

Import a command by doing the following:

6. Access the Advanced Commands Editor, as described in Section 5.1.4.1.
7. In the **Editor** pane, select the location to which you want to insert the imported command.
8. On the Advanced Commands toolbar, click the **Open from File** icon.
   The **Open** dialog box appears.
9. Select the CLEO command file that you want to import.
10. Click **Open** to import the command file into the Advanced Command Editor.
    The command file is imported into the sequence of commands.

5.1.4.8 Disabling and Enabling Commands

Commands in the **Advanced Commands Editor** can be disabled from the command sequence without being deleted from the **Editor** pane. This is useful when you do not need to use a command in a Leo Wizard but want to keep it for future use or reference. Disabled commands can be enabled as needed.

Disable and enable commands by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 In the **Editor** pane select the command that you want to disable or enable.
3 Do one of the following:

- To disable the command, click the **Disable** icon on the **Advanced Commands** toolbar, or press **CTRL+D**.
- The command is grayed out and deactivated in the Advanced Commands Editor.
- To enable the command, click the **Enable** icon on the **Advanced Commands** toolbar, or press **CTRL+E**.
- The command is recolored and reactivated in the Advanced Commands Editor.

5.1.4.9 Removing Advanced Commands

Remove advanced commands by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 In the **Editor** pane, select the commands that you want to remove, and do one of the following:

- Press **DELETE**.
- On the **Advanced Commands** toolbar, click **Delete** .

Removing a command block, such as a Condition command, removes the entire command block after displaying a warning message.

3 To remove all commands currently defined in the Command Editor, on the **Advanced Commands** toolbar, click **Delete All** .

5.1.4.10 Grouping Advanced Commands

Make a logical group from a sequence of commands.

Group command by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 In the **Editor** pane, select the commands you want to group.
3 On the Advanced Commands toolbar, click the **Group** icon.

5.1.4.11 Ungrouping Advanced Commands

Break a group of advanced commands.

Ungroup command by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 In the **Editor** pane, select the group you want to ungroup.
3 On the Advanced Commands toolbar, click the **Ungroup** icon.

5.1.4.12 Advanced Commands Global Groups

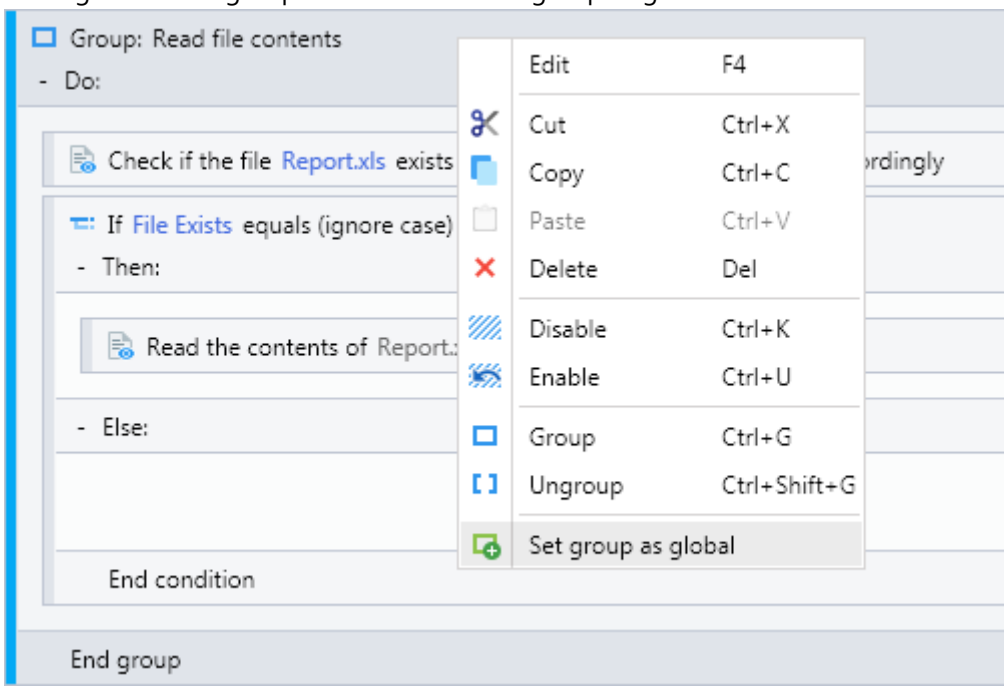
Reuse a predefined set of commands in various places within a wizard by converting a group into a Global Group.

Create a global group by doing the following:

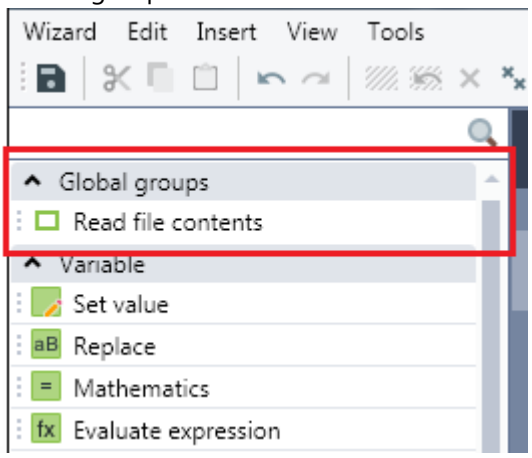
1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.

2 In the **Editor** pane, select the group you want to set as global.

3 Right click the group icon and select Set group as global.



4 The new global group will be added to the **Command Toolbox** pane, under the "Global groups" section.



5 You can now use this group as a command. On wizard runtime the global group will execute its sequence of commands automatically.

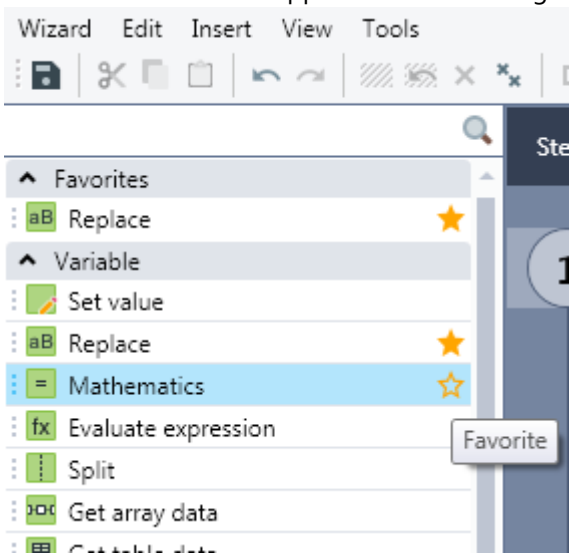
5.1.4.13 Favorite Advanced Commands

Mark your favorite commands on the **Command Toolbox** with a star to easily find them.

Create a global group by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.

2 In the **Command Toolbox**, select the command you want to set as favorite.
3 Click the star icon that appears when hovering over the command name



4 The command will be marked as favorite and will also appear under "Favorites" section in the **Command Toolbox**.
5 To remove the command from the list of favorites, click the star icon again.

5.1.4.14 Undoing and Redoing an Advanced Command Action

Most actions you perform when managing advanced commands can be undone and redone. However, actions can only be undone before you close the Advanced Command Editor. Once you close the Editor, the action can no longer be undone.

To undo or redo an action when editing an advanced command, do one of the following:

- On the **Advanced Commands** toolbar, click either the **Undo** or **Redo** icon.
- Press **CTRL+Z** to undo or **CTRL+Y** to redo an action.

5.2 Find

Leo wizard editor features a find mechanism that allows finding actions and commands by the following properties:

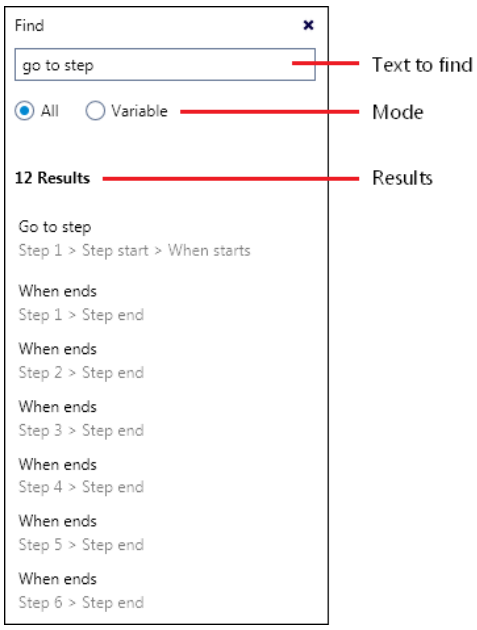
- **Flow action name**: e.g. "Step start", "Step end", "Bubble"
- **Flow action text**: applicable only for bubbles, who have editable text.
- **Flow event name**: e.g. "When starts", "Object not found"
- **Flow event text**: applicable for bubble buttons, who have editable text.
- **Flow command**: e.g. "Go to step", "End wizard"
- **Advanced command**:
  - Name
  - Textual properties
  - Variable usage

Find actions, commands and variables by doing the following:

1 Access the Advanced Commands Editor, as described in Section 5.1.4.1.
2 On the Advanced Commands toolbar, click the **Find...** button.
3 The find pane will open on the right side of the window (see Figure 197)
4 Type the text to find
5 Select the find mode: choose whether you're searching for a variable or free text.

6 Press **Enter** to start searching the wizard.
7 The Results list will be populated with all the commands that were found.
8 Click a result for Leo Studio to focus it on the Advanced Commands Editor.

Figure 179: Find panel



5.3 Global Actions

The Global Actions feature (Figure 198) enables you to globally handle fallback errors, and to determine global start/end actions for all steps.

5.3.1 Error Handling

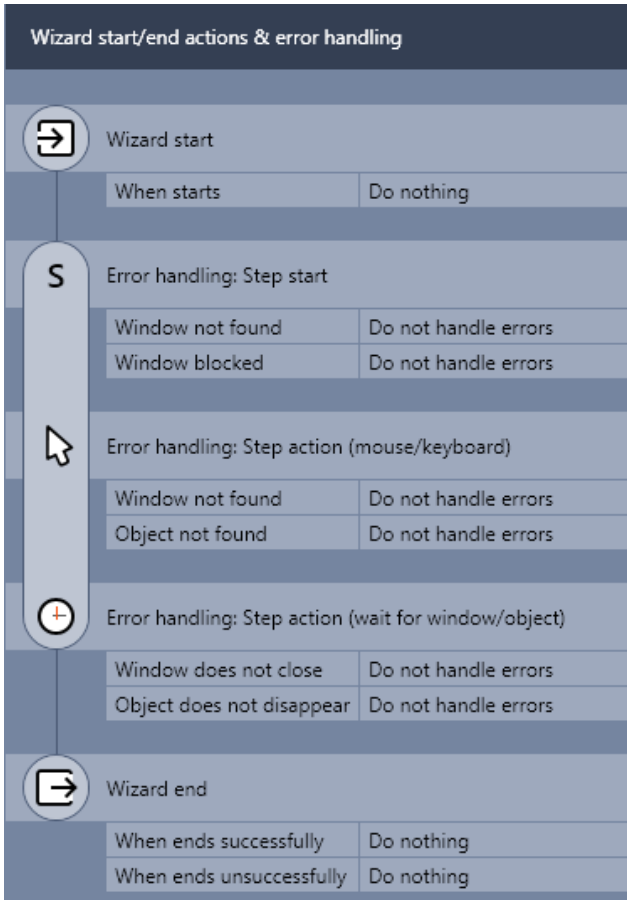
Global error handling lets you handle all fallbacks in a wizard that belong to a specific action type.

Global error handling is available for the following action fallbacks:

- Step Start Fallbacks
- Step Action (mouse/keyboard) Fallbacks
- Wait Action (wait for window/object) Fallbacks

For information about fallbacks, see Section 3.3.2.3.

Figure 180: Global Error Handling



5.3.2 Wizard Start / Wizard End Actions

Start/End Action handling (Figure 198) enables you to set a command for all step start/end actions in the wizard.

The available actions that you can handle globally are:

- Wizard starts: when starts
- Wizard end: when ends successfully
- Wizard end: when ends unsuccessfully

5.4 Credentials Vault

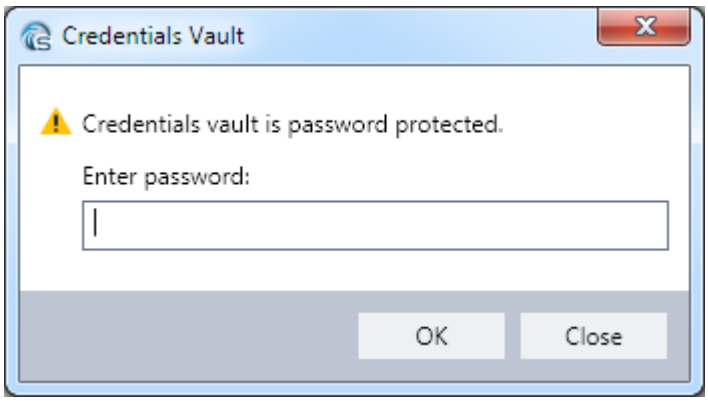
The Credentials Vault feature enables you to securely save usernames and passwords in Leo's server for later use within wizards.

Storing and using passwords can be used for desktop automation wizards running by end-users or robotic process automation wizards running by Leo Robots.

Passwords are saved in DB with a 2-phased encryption mechanism, allowing only the Leo components to be able to decrypt the information and enter the passwords when needed.

Access the Credentials Vault by doing the following:

1. From **Leo Studio** main menu, click Tools > Credentials Vault
2. If the vault is password protected by your system administrator, the following window will appear. Enter the password given to you by the system administrator.



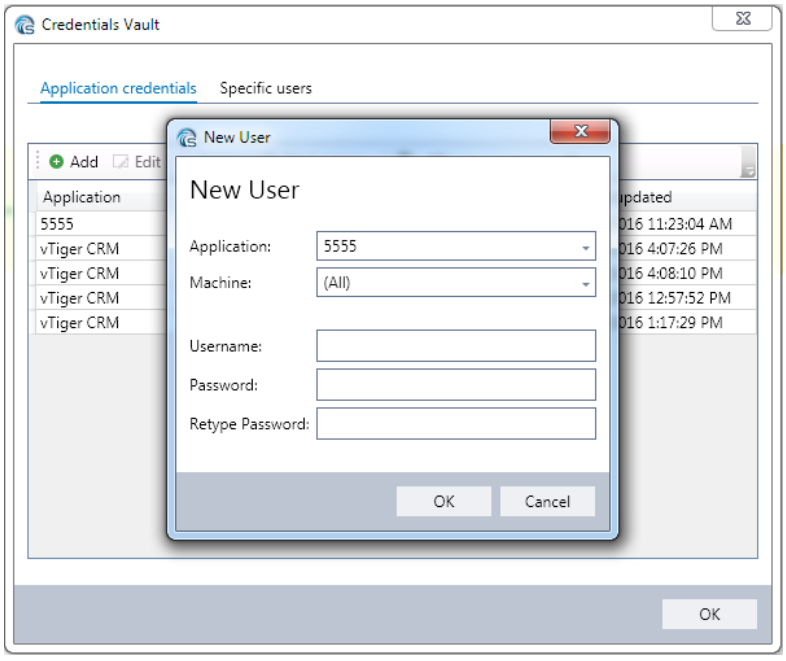
5.4.1 Application Credentials

Credentials vault can store username-password combination per machine and application. Use if your application has different username-password combination for any user or machine. When using the credentials, Leo will fetch the correct username-password combination according to the machine the wizard is currently running on.

Add Specific User credentials to the Credentials Vault by doing the following:

1. From **Credentials Vault** main window, select the "Application Credentials" tab
2. Click **Add**
3. On the "New User" window, select the application whose credentials you are entering, type the username and password.
4. Select the machine that will own these credentials, or select (All) if these are default credentials that every machine can use.

Figure 181: Application Credentials



5.4.2 Specific Users

Credentials Vault can store any username-password combination with a certain ID. Use this option if your application has a specific username-password combination that will work from any user or machine trying to use it.

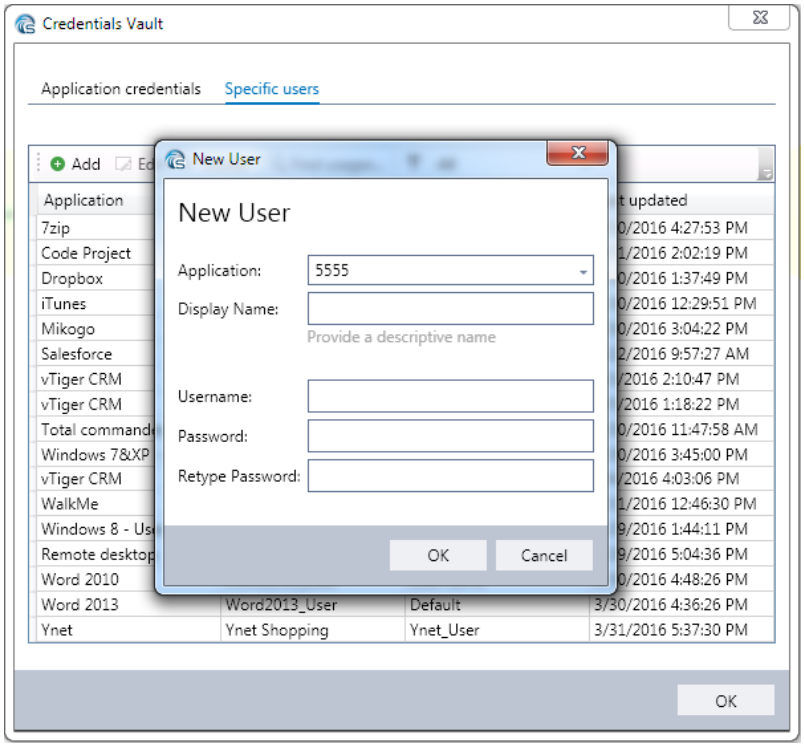
Add Specific User credentials to the Credentials Vault by doing the following:

5 From **Credentials Vault** main window, select the "Specific User" tab

6 Click **Add**

7 On the "New User" window, select the application whose credentials you are entering, type the username and password, and provide a friendly name to later access these credentials.

Figure 182: Specific user credentials



6 Appendix A: Leo Report Breakdown

The following section describes the parameters and components of all Leo reports generated from the Leo Studio Report Generator.

For more information about generating Leo reports, see Section 2.5.

6.1 Wizards

6.1.1 Wizard Use

6.1.1.1 Wizard Run Data

- **Per Wizard/User**: all played wizards and all usernames who played them, grouped by user/wizard, with these statistics:
  - Date and time of wizard launch
  - Wizard run mode
  - Source for launching the wizard: Search/catalog/hyperlink/sensor/**Run Again** button
  - How it ended – ended successfully, stopped, failed, ended unsuccessfully
  - ID of last step played
  - Wizard duration
- **Full Catalog (List)**: each wizard is presented with its full path, with these statistics:
  - Total runs per wizard
  - Success percentage rates (how it ended - successfully, stopped, failed, unsuccessfully)
  - Total no. of feedbacks received
  - Feedback percentage rates
- **Full Catalog (Detailed)**: Each wizard is presented with its full path, description, keywords, with these statistics:
  - Number of times it ended a certain way (successfully, stopped, failed, unsuccessfully)
  - User feedback (total no. each star rating was given)

6.1.1.2 Wizard Failure Data

- **Per Failure Type**: For each wizard:
  - Wizard ID
  - Total no. of failures per wizard
  - For each wizard: Total no. of failures per step
  - For each step: Total no. of failures per failure type, divided into run modes.
- **Detailed View**: For each wizard:
  - Wizard ID
  - For each step: List of failures by type, incl. username and time of failure

6.1.1.3 Wizard Duration Statistics

For each library:

- Wizard with full path
- Total no. of successful runs
- Duration: average, median, minimum, minimum.

6.1.1.4 Wizard Success Rates

For each wizard:

- Wizards listed by total no. of runs
- Wizard ID
- Success percentage rates (how it ended - successfully, stopped, failed, unsuccessfully)

6.1.1.5 Answered Queries

For each wizard:

- Wizard ID
- Total no. of queries
- The search query typed by the user
- The username
- Time of query

6.1.1.6 Library Usage Summary

- Top 10 wizards (with total no. of runs), colored by success (how it ended - successfully, stopped, failed, unsuccessfully)
- Usage graph: Total no of runs by date, colored by how the wizards ended.
- Success rates in numbers and percentages, with totals
- Total user feedback in numbers and percentages, with totals

6.1.2 Unanswered Queries

Lists each search query by user and time of query, and whether the query generated search results.

6.1.3 Full Catalog (List)

Lists published wizards by category, with description and wizard ID.

6.1.4 Runtime User Actions

Displays clicks on bubble buttons, variable values or other actions you chose to report on when developing the wizard, by enabling the **Report User Action** advanced command.

6.1.4.1 Runtime Actions per Wizard

For each wizard, displays:

- Catalog Path
- Name:
  - Bubble: Name/Bubble Text
  - Advanced Command: Action Name
- Username
- Value: Button Caption
- Date and Time
- Run ID

6.1.4.2 Runtime Actions per User

For each user, displays:

- Catalog Path
- Action Type: Bubble or advanced command
- Name:
  - Bubble: Name/Bubble Text
  - Advanced Command: Action Name
- Value:
  - Bubble: Button Caption
  - Advanced Command: Value (can be variable name, etc.)
- Date and Time
- Run ID

6.1.4.3 Runtime Action Summary

For each wizard, displays:

- Catalog Path
- Action Type: bubble or advanced command
- Name:
  - Bubble: Name/Bubble Text
  - Advanced Command: Action Name
- Value:
  - Bubble: Button Caption
  - Advanced Command: Value (can be variable name, etc.)
- Count: Total number of times action was performed
- Last User: Last user to click the reported button

6.2 Sensors

6.2.1 Sensor Activity

Displays a read receipt for each sensor, listed by username including date and time.

6.2.2 Runtime User Actions

6.2.2.1 Runtime Actions per Sensor

For each sensor, displays:

- Catalog Path
- Step Number
- Bubble ID
- Name:
  - Bubble: Name/Bubble Text
  - Advanced Command: Action Name
- Command: Value (can be variable name, etc.)
- Username
- Value:
  - Bubble: Button Caption
  - Advanced Command: Value (can be variable name, etc.)
- Date and Time
- Run ID

6.2.2.2 Runtime Actions per User

For each user, displays:

- Catalog Path
- Action Type: Bubble or advanced command
- Step Number
- Bubble ID
- Name:
  - Bubble: Name/Bubble Text
  - Advanced Command: Action Name
- Value:
  - Bubble: Button Caption
  - Command: Value (can be variable name, etc.)
- Date and Time
- Run ID

6.2.2.3 Runtime Action Summary

For each sensor, displays:

- Catalog Path
- Action Type: Bubble or advanced command
- Sensor Name
- Value:
  - Bubble: Button Caption
  - Advanced Command: Value (can be variable name, etc.)
- Count: Total number of times action was performed
- Last User: Last user to click the reported button

6.3 Administration

6.3.1 Active Users

For each company, the no. of times each active user played wizards, with date and time of run. Lists the total number of users and runs for the selected time range.

6.3.2 Login History

6.3.2.1 First Login

The number of first time logins by users over a specific time range:

- Filtered by Leo client (Player/Studio)
- Graph with total number of first-time logins by date.
- For each date, displays:
  - Time of login
  - User ID
  - Leo username and full name (if applicable)
  - Company Name

6.3.2.2 Logged-In Users

Total number of logged in Leo users, over a specific time range:

- Filtered by Leo client (Player/Studio)
- Graph with total number of logins by date.

- For each date, displays:
  - User ID
  - Leo username and full name (if applicable)
  - Company Name

6.3.3 User Management Audit

Lists actions performed in Leo Admin on user accounts:

- Date and Time
- Admin User
- Action:
  - Reset Password
  - User deleted
  - User created
  - User updated
  - User unlocked
  - Login
- Leo User: The user account on which the action was performed
- Company: The admin company the Leo user pertains to.
- More details:
  - User type: Studio\Player
  - Updated user details (e.g. email address)

6.3.4 User Login Audit

Lists login-related user actions:

- Date and Time
- Leo Product: Player, Studio
- Leo User: The user account on which the action was performed
- Company: The admin company the Leo user pertains to.
- Action:
  - Login
  - Failed Login
  - User created
  - Password changed
- More details: Number of login attempts

6.3.5 Registered Users

Lists user registration and other activities in the Leo platform:

- For Leo Player users:
  - Registered username
  - Date created in Admin
  - Date of last Leo login
  - Date of last wizard run
  - Date of last sensor run
  - Name of the user's machine
- For Leo Studio users:
  - Registered username
  - Date created in Admin
  - Date of last Leo login

- Date of last wizard save
- Date of last sensor save
- Name of the user's machine

6.3.6 Active Robotic Clients

Lists all active robotic clients per company:

- Machine name
- User group
- Client state
- Creation date
- Name of last task performed
- Start time of last task performed

7 Appendix B: Leo Plugin Development

This section is intended for computer programmers who need to know how to develop Leo plugins, for use in the advanced command Run Plugin Method.

For more information about the Run Plugin Method command, see Section 5.1.3.7.

7.1.1 Plugin Framework

Plugin framework is the most common design pattern used for application extension.

Plugin interfaces enable third-party developers to extend the Leo application, 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, plugins must sometimes be unloaded for the following reasons:

- The plugin consumes excessive resources.
- The plugin has not been used for a certain amount of time.

Loading an assembly in .NET is very easy and for the most part does not require high plugin design complexity. However, unloading a .NET assembly increases the design complexity.

Basically, there is an interface that you must implement 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 **ILeoPlugIn** 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**.

7.1.1.1 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 Leo expression. The attribute also applies the Leo policy regarding method signature, validates return type and input arguments permitted by Leo expression .NET types:

- Blittable
- Some non-blittable types: string, object, Boolean

7.1.1.2 Plugin Usage

Leo plugins can be defined and used in advanced commands in two ways:

- By Location (Defining a path)
- Embedding

7.1.1.2.1 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 potential plugin into 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 Leo Studio, unless there are changes in the method signature.

Distribution of the external plugin files is to be done by the organization's IT department. When playing a wizard that uses an external plugin, Leo will assume that the required DLL files are already located in the client machine and will try to run them.

7.1.1.2.2 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 Leo'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 Leo Studio interface and by redefining its in/out parameters.

Distribution of the external plugin files is done automatically by Leo. 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.

7.1.2 Developing a Leo Plugin

Create a Leo plugin by doing the following:

1 Add a Reference to Leo.Plugin.Library.dll

2 Mark Assembly as Leo Plugin
3 Derive from LeoPlugIn
4 Implement LeoPlugIn
5 Add a Plugin to the Leo Plugin Library
6 Apply Plugin to Individual Methods

7.1.2.1 Add a Reference to Leo.Plugin.Library.dll

Use the **Add Reference** dialog box to make LeoPlugIn a reference in your project.

7.1.2.2 Mark Assembly as Leo 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 tells the Leo 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.

7.1.2.3 Derive from LeoPlugIn

To make this class a Leo 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
       {
       }
    }
}
```

7.1.2.4 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.

To create your own logic:

- To initialize your own objects use the initialize method, this is invoked before the execution process.
- To clean up/dispose your own objects, use the shutdown method, which is invoked prior to the Leo 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

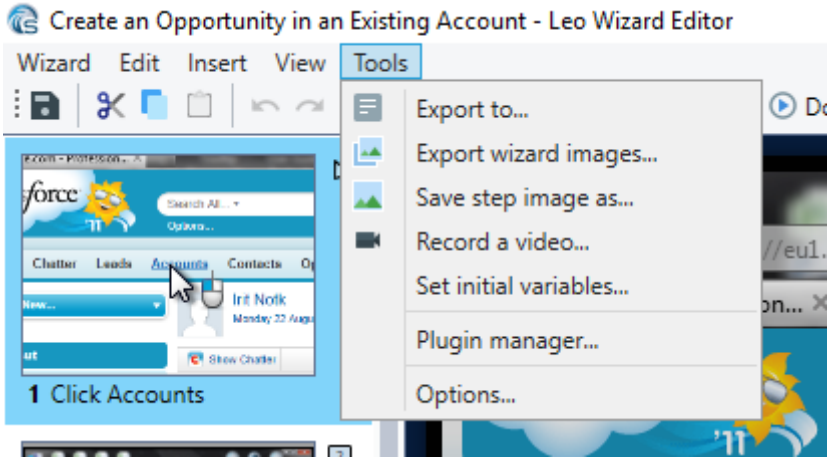
```
using Kryon.Leo.Infrastructure.Plugin.Library;
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;

namespace SamplePlugIns
{
    public class PlugInA : LeoPlugIn
    {
        public override void Initialize()
        {
            Console.WriteLine(Name + " Initializing...");
        }
    }
}
```

7.1.2.5 Add a Plugin to the Leo Plugin Library

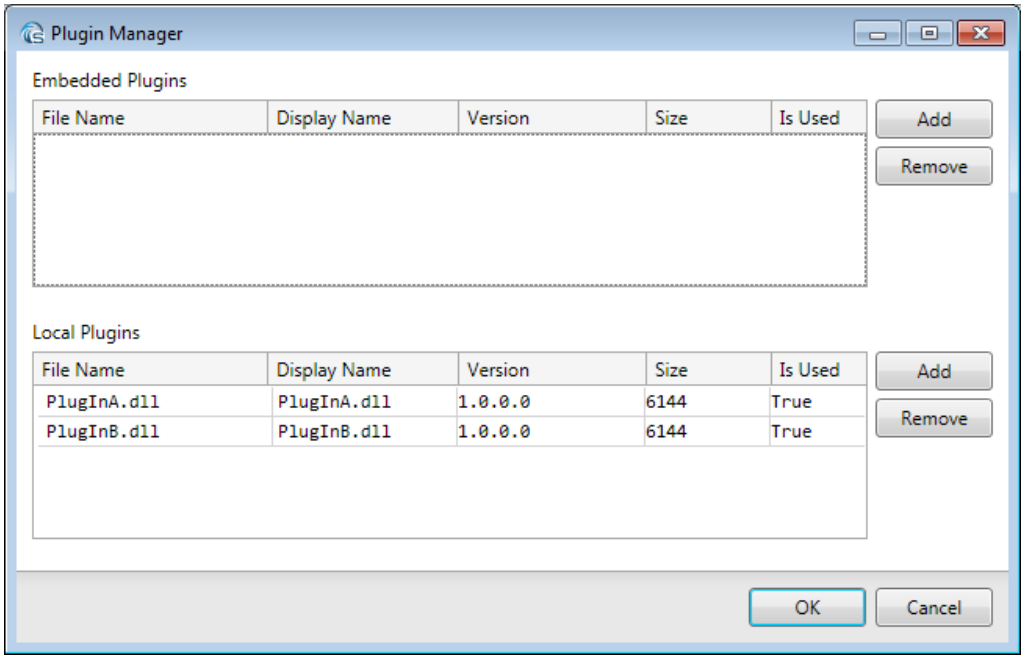
1 From the Leo Studio Editor window, go to **Tools > Plugin Manager** (Figure 203).

Figure 183: Accessing the Plugin Manager



The **Plugin Manager** dialog box appears (Figure 204).

Figure 184: Plugin Manager Dialog Box



2 Depending on whether your plugin is embedded or local, click the corresponding pane's **Add** button.

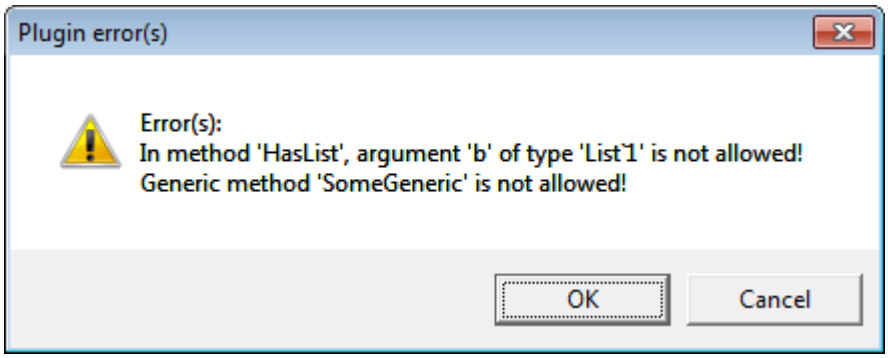
3 Navigate to the folder in which your DLL plugin is saved, and select it.

4 Your plugin is added and can be used from the Run Plugin Method advanced command, as described in section 7.1.2.5.

If the following error appears (Figure 205), it indicates that the plugin you are trying to add uses methods that are unsupported by the Leo Plugin Library framework. To resolve this error, use a different plugin or make sure that the plugin is updated to use methods that are supported by the Leo framework.

For a description of the supported methods, see section 7.1.1.1.

Figure 185: Plugin Error



7.1.2.6 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 Leo. Simply apply it to individual methods.

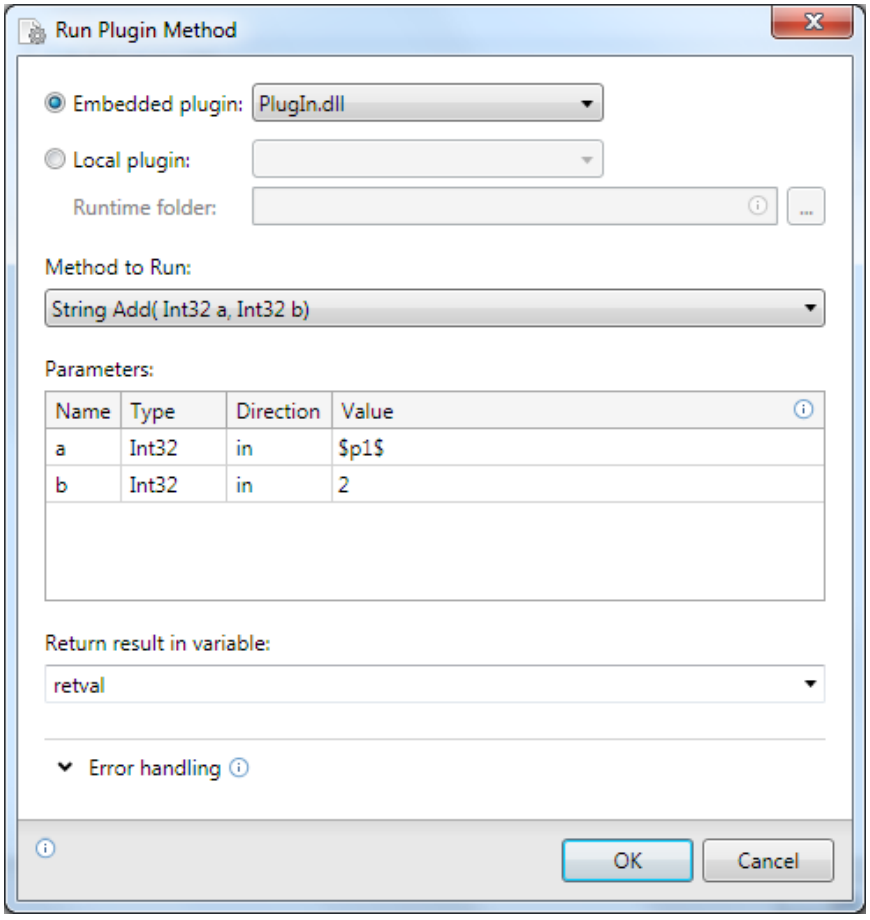
```
using Kryon.Leo.Infrastructure.Plugin.Library;
using Kryon.Leo.Infrastructure.Plugin.Library.Attributes;


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 (

Figure 180).

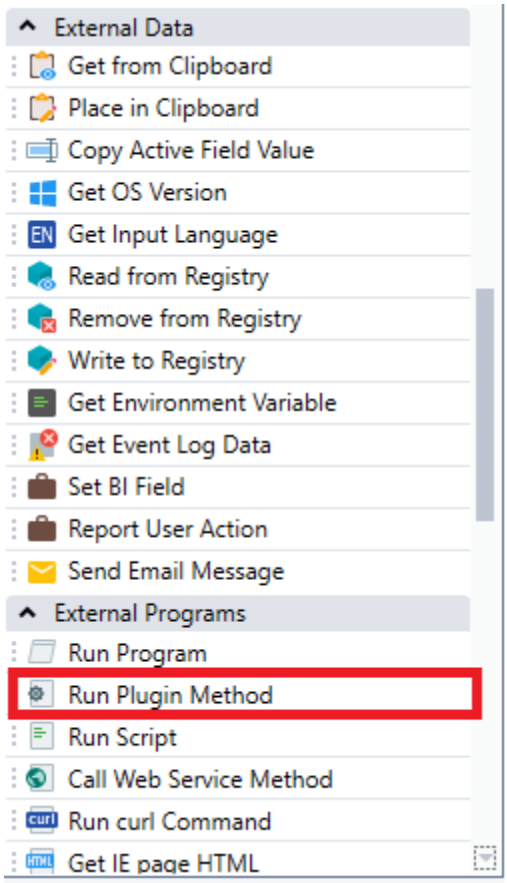
Figure 186: Plugin Method in Leo Studio



To launch the method, Leo Studio users must click the Run Plugin Method command in the Studio's Advanced Commands Editor (

Figure 180).

Figure 187: Plugin Command in Leo Studio



8 Appendix C: BI Fields in the Leo Database

This section is intended for Business Intelligence (BI) personnel who need to know how to retrieve Leo statistics from the Leo database.

The Leo statistics that can retrieved from the Leo database include usage statistics about the wizard, such as who ran it, the run duration, the last step run in the wizard and more. In addition, the content developer can use the Set BI Field advanced command to report additional information into the Leo database.

Access Leo BI statistics from the Leo database by doing the following:

1 In the Leo database, run the view named **View_LeoStatisticsForBI**.
2 Filter the results as needed, using the columns described in Table 14.

The BI fields recorded in the database for each wizard run are:

Table 14: BI Fields in the Leo Database

Field Name	Field Value
StatisticID	A primary key, which is a unique number.
Time	The date and time when the wizard run data was saved to the database. Note that this is not the time of the wizard run in Leo Player. It is the time logged just after the wizard run ends, when the data is saved to the database.
CompanyID	The ID of the company as defined in Leo Admin. Usually one company is defined.
Company Name	The name of the company as defined in Leo Admin. Usually one company is defined.
LibraryID	The ID of the Leo Catalog library that contains the wizard.
LibraryName	The name of the Leo Catalog library that contains the wizard.
WizardID	The wizard ID
WizardName	The wizard name
UserID	The Leo ID of the user that ran the wizard.
UserName	The name of the user that ran the wizard.
ReportType	The wizard run result, identified by the following IDs: 1 = Ended Successfully 2 = Ended Unsuccessfully 3 = Stopped by the User 4 = Failed
ReportSubType	For failed wizards, an ID for each failure reason. For more information, please contact the Kryon Systems team.
RunMode	The ID of the wizard run mode: 1 = Do It 2 = Guide Me

Field Name	Field Value
Source	The ID of how the user launched the wizard: 1 = Search Bar 2 = Catalog 3 = Hyperlink 4 = **Run Again** button
LastStep	The last step number that was run.
RunSteps	The number of steps run in this wizard.
Duration	The duration of the wizard run in milliseconds.
NetoDuration	Measurements for internal Kryon Systems usage.
CustomData1	The variable value set by the content developer into BI field 1
CustomData2	The variable value set by the content developer into BI field 2
CustomData3	The variable value set by the content developer into BI field 3
CustomData4	The variable value set by the content developer into BI field 4
CustomData5	The variable value set by the content developer into BI field 5