

# Advanced Commands Reference Guide

Leo Studio v5.3.56



INTELLIGENT ROBOTIC PROCESS AUTOMATION



This document contains Kryon Systems proprietary information. The information contained herein is confidential and cannot be distributed without the prior written approval of Kryon Systems Ltd.

© 2008-2018 Kryon Systems Ltd. All rights reserved.

# Contents

## CHAPTER 1: Variable Commands

Set Value .....	10
Find .....	12
Replace .....	19
Mathematics .....	21
Evaluate Expression .....	22
Split .....	27
Get Array Data .....	31
Get Table Data .....	35
Get ASCII Character .....	39
Remove Blank Spaces .....	41
Change Letter Case .....	42
Check Type .....	43
Get Random Number .....	44
Reverse .....	45
Get Length .....	46
Extract Numeric Values .....	47

## CHAPTER 2: Flow Commands

If Else .....	51
Complex If Else .....	55
Multi-Value If Else .....	58
Loop .....	60
Loop: Break .....	62
Loop: Restart .....	63
Loop Items .....	64
Pause .....	66
Group .....	67

## CHAPTER 3: Wizard Commands

Continue Wizard .....	71
End Wizard .....	72
Go To Step .....	73

Get Step Data .....	74
Get Wizard Data .....	75
Check Leo Application .....	76
Get Leo User Data .....	77
Check Run Mode .....	78
Check Video Recording Mode .....	79
Set User Interrupt Mode .....	80
Show Message .....	81
Raise Wizard Error .....	83
Get Last Failure Type .....	84
Resume Error .....	85
Report Wizard Output .....	86

#### **CHAPTER 4: Mouse and Keyboard Commands**

Use Keyboard Shortcut .....	88
Input Text .....	89
Get Mouse Position .....	90
Drag & Drop .....	92
Mouse Click .....	93
Wait for Busy Cursor .....	94
Set Caps Lock State .....	95
Get Caps Lock State .....	96

#### **CHAPTER 5: Block Commands**

Allow last stored action .....	97
Deny last stored action .....	97
Remove all blocks .....	98

#### **CHAPTER 6: Date and Time Commands**

Get Current Date .....	101
Validate Date .....	102
Get Day of Month .....	103
Compare Dates .....	105
Add/Subtract Date .....	107
Calculate Date Range .....	108

Check Day of Week .....	110
Format Date .....	112
Get Current Time .....	114
Compare Time .....	115
Add/Subtract Time .....	117
Calculate Time Range .....	119
Convert Between Time Zones .....	122

## CHAPTER 7: Window Commands

Get Step Window Handle .....	125
Find Matching Window Handles .....	126
Get Active Application .....	128
Control Window State .....	129
Check Window State .....	130
Get Active Window/Web Page .....	131
Capture Step Window Image .....	132

## CHAPTER 8: File Commands

Create a Text File .....	134
Read From Text File .....	136
Write to Text File .....	137
Does File Exist .....	138
Copy a File .....	139
Move a File .....	140
Rename a File .....	141
Delete a File .....	142
Delete File(s) .....	143
Monitor File Changes .....	145

## CHAPTER 9: Folder Commands

Create a Folder .....	148
Get Folder Location .....	149
Get Files .....	150
Does Folder Exist .....	152
Is Folder Empty .....	153

Copy a Folder .....	154
Move a Folder .....	155
Rename a Folder .....	156
Delete a Folder .....	157
Monitor Folder Changes .....	158

## **CHAPTER 10: Scanned Document Analysis (OCR) Commands**

Document Analysis .....	161
Document: Get Text .....	165
Document: Get Value .....	166
Document: Get Checkbox State .....	168
Document: Does Word Exist .....	170
Document: Get Date/Time .....	171
Document: Save as Image .....	173
Document: Save to Excel .....	174
Convert to Text (SmartScan+) .....	176
Convert to Text (Tesseract) .....	177

## **CHAPTER 11: Digital PDF Analysis Commands**

Analyze Digital PDF File .....	179
Page: Get Text .....	182

## **CHAPTER 12: External Data Commands**

Get From Clipboard .....	184
Place in Clipboard .....	185
Copy Active Field Value .....	186
Get OS Version .....	187
Get Input Language .....	188
Read From Registry .....	189
Remove From Registry .....	191
Write to Registry .....	192
Get Environment Variable .....	193
Get Windows Event Log Data .....	194
Set BI Field .....	196
Report User Action .....	197

Query XML .....	198
<b>CHAPTER 13: Email Commands</b>	
Send Email Message .....	201
Get Email Messages .....	206
Email: Get Data .....	211
Email: Move to Folder .....	212
Email: Forward .....	213
Email: Reply .....	215
Email: Delete .....	216
Email: Mark as Read/Unread .....	217
Email: Save Attachments .....	218
<b>CHAPTER 14: External Program Commands</b>	
Run Program .....	221
Run .NET Plugin Method .....	223
Call Web Service Method .....	224
Run Script .....	225
Run curl Command .....	227
Get IE Page HTML .....	228
Run JavaScript on Page .....	229
<b>CHAPTER 15: Database Commands</b>	
Execute SQL Query .....	232
Monitor Database Changes .....	234
<b>CHAPTER 16: Credentials Vault Commands</b>	
Insert Username Into Active Field .....	237
Insert Password Into Active Field .....	238
Generate New Password .....	239
Revert Password .....	240
<b>CHAPTER 17: Robotic Process Automation Commands</b>	
Add Automation Task to Queue .....	243
Get Automation Task Status .....	246
Get File Trigger Input .....	248
Get Folder Trigger Input .....	249

Get Email Trigger Input .....	250
Get Database Trigger Input .....	251
<b>CHAPTER 18: Excel Commands</b>	
Get Excel Range Values .....	254
Set Excel Range Values .....	256
Excel Worksheet Actions .....	259
Excel Row Actions .....	260
Excel Column Actions .....	262
Convert Excel to CSV .....	264
Create New Excel File .....	265
Query From Excel .....	266
Run Macro .....	268
<b>CHAPTER 19: SAP Commands</b>	
Get SAP Object Text .....	271
Get SAP Object Value .....	272
Get SAP Object Location .....	273
Set SAP Object Value .....	275
<b>CHAPTER 20: UI Automation Commands</b>	
Get UI Object Text .....	277
Get UI Object Value .....	278
Get UI Object Location .....	279
Set UI Object Value .....	281
<b>CHAPTER 21: HTML Commands</b>	
Get HTML Table .....	284
Get HTML Object Text .....	287
Get HTML Object Value .....	289
Get HTML Object .....	291
Set HTML Object Value .....	293
Does HTML Object Exist .....	295
Click on HTML Object .....	297
<b>CHAPTER 22: .NET Automation Commands</b>	
Get .NET Object Text .....	300

Get .NET Object Value .....	301
Get .NET Object Location .....	302
Set .NET Object Value .....	304
<b>CHAPTER 23: Java Automation Commands</b>	
Get Java Object Text .....	307
Get Java Object Value .....	308
Get Java Object Location .....	310
Set Java Object Value .....	312
<b>CHAPTER 24: Global Variable Commands</b>	
Set Global Variable .....	314
Get Global Variable .....	315
Delete Global Variable .....	316
<b>CHAPTER 25: Scripting Commands</b>	
Note .....	318
Notes for Mobile/Web Access .....	319
View Variable List .....	320
Show Debug Message .....	322
<b>APPENDIX A: Error Handling</b>	
Using ERROR HANDLING options .....	323



# CHAPTER 1: Variable Commands

In this chapter:

Set Value .....	10
Find .....	12
Replace .....	19
Mathematics .....	21
Evaluate Expression .....	22
Split .....	27
Get Array Data .....	31
Get Table Data .....	35
Get ASCII Character .....	39
Remove Blank Spaces .....	41
Change Letter Case .....	42
Check Type .....	43
Get Random Number .....	44
Reverse .....	45
Get Length .....	46
Extract Numeric Values .....	47

## Set Value

- Create a new variable and set its value; or
- Set the value of an existing variable

### Using the SET VALUE command

- 1 Enter the name of the variable (new or existing) to which you want to assign a value
  - If you want to create a new variable, type the name of the new variable
  - If the variable already exists, choose the name of the variable from the drop-down list
- 2 Set the value of the variable you have specified
  - You can include free text and/or values copied from different variables
    - <Enter> <Space> and/or <Tab> can be used
  - To include the value of a different variable, indicate its name by typing it between dollar signs (e.g., \$MyVar\$)



#### TIP

##### Create "special character" variables to use later in other Advanced Commands

To use <Enter> <Space> or <Tab> in other Advanced Commands, it's a great idea to create variables for these special characters up front:

- Create a variable named **Enter**, and set its value to <Enter>
- Create a variable named **Space**, and set its value to <Space>
- Create a variable named **Tab**, and set its value to <Tab>
- Create a variable named **Empty**, and set its value to <Nothing> (i.e., no character)

You'll find these special variables especially useful when using delimiters (such as in the **SPLIT** command) or when replacing text (such as in the **REPLACE** command).



### EXAMPLE

Build a date string by combining month, day, and year

1. Set a variable to define the separator character

The screenshot shows a 'Set value' dialog box with a title bar containing a close button (X). Inside the dialog, there are two main sections: 'In the variable:' with a dropdown menu currently showing 'separator', and 'Set the value:' with a text input field containing a single forward slash character '/'. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel', with an information icon (i) to the left of the 'OK' button.

**Result:** ⚡ separator = /

2. Set a variable to store the date using the predefined separator character

The screenshot shows a 'Set value' dialog box with a title bar containing a close button (X). Inside the dialog, there are two main sections: 'In the variable:' with a dropdown menu currently showing 'date', and 'Set the value:' with a text input field containing the string '08\$separator\$10\$separator\$1967'. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel', with an information icon (i) to the left of the 'OK' button.

**Result:** ⚡ date = 08/10/1967

## Find

- Find the location of a [specific text](#) string within a variable; or
- Find all text matching a [regular expression](#) within a variable



### NOTE

#### What is a *regular expression*?

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

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

## Using the FIND command

### To find specific text

When you search for the location of a specific text, Leo will return a number indicating the character position of the **first instance** of the text (based on the search direction you select).

- Character position is counted from the top left corner of a variable and includes spaces.
- If the text you are searching for contains multiple characters, Leo will return the location of the first character in the string.
- If the text you are searching for is not found, Leo will return the value 0.

- 1 Enter the name of the variable in which you would like to search
- 2 Select **FIND TEXT**
- 3
  - Enter the text for which you would like to search (free text and/or values copied from different variables);
  - Enter the character position at which you would like to begin searching; **and**
  - Select whether to search **START TO END** or **END TO START**
- 4 Indicate whether you wish to allow **close matching** of the specified text; **and** the level of accuracy required for the close match to be accepted
- 5 Indicate whether Leo should ignore letter case when identifying matching text
  - If left unchecked, only text in the same case as that entered will be considered a match
  - When **close match** is allowed, Leo will always ignore letter case
- 6 Enter the name of the variable into which you'd like Leo to place the search result



## NOTE

### What is a close match?

Close match (sometimes referred to as *fuzzy match*) allows a certain level of flexibility in the matching of visually similar characters – such as the number 1 and a lowercase L – which can be very useful when working with scanned documents.

### How close does the match need to be?

By using the **REQUIRED ACCURACY** slider, you determine how visually similar the characters need to be in order for the match to be accepted.

For example, with a high required accuracy setting:

- Leo would likely accept the word `c1ose` as matching the word `close` because the number 1 is highly visually similar to a lowercase L.
- Leo would **NOT** likely accept the word `ad ress` as matching the word `address` because a blank space is not highly similar to a lowercase D.

In order for the word `ad ress` to be accepted as matching the word `address`, you would need to specify a lower required accuracy setting.



### EXAMPLE

#### Finding the location of specific text

`⚡ quote` = I think, therefore I am.

**Result:** `⚡ find result` = 3

## To find text matching a regular expression

When you search for text matching a regular expression, Leo will return all instances of text matching the pattern, separated by a delimiter you specify.

- 1 Enter the name of the variable in which you would like to search
- 2 Select **FIND TEXT MATCHING A REGULAR EXPRESSION**
- 3 Enter the regular expression for which you would like to find matching text; *and* Enter the delimiter to use to separate each matching text found
  - **Try it out:** (Optional) To ensure that your regular expression will provide the expected results, try some test data. Simply click the **TEST** link from within the **FIND** command.
- 4 Indicate whether Leo should ignore letter case when identifying matching text
  - If left unchecked, only text with the same case as that entered will be considered a match





Enter the name of the variable into which you'd like Leo to place the search result

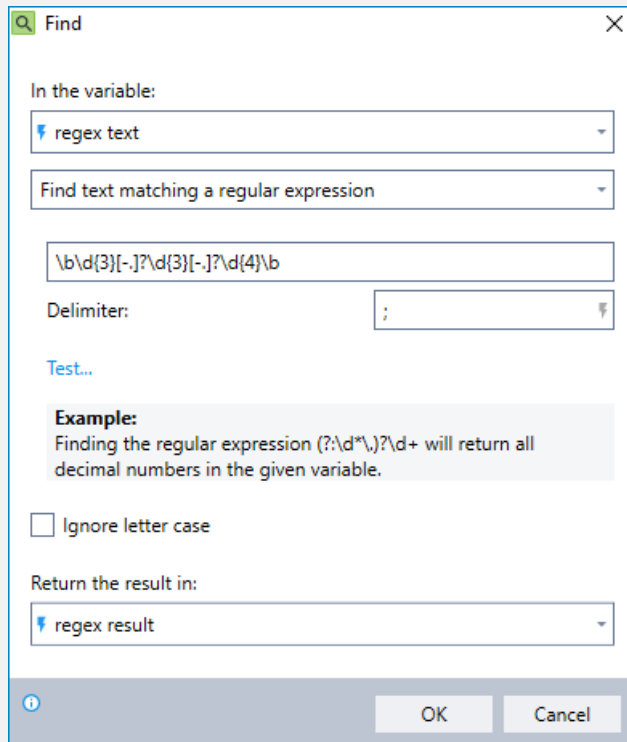


## EXAMPLE

### Extracting phone numbers using a regular expression

Let's say you have copied a large block of text from a web page, and you want Leo to extract all telephone numbers from this text. *(Note: The regular expression used in this example will find all phone numbers in the following formats: 444-555-1234; 444.555.1234; 4445551234.)*

```
⚡ regex text = abcdefghijklmnopqrstuvwxyz
                ABCDEFGHIJKLMNOPQRSTUVWXYZ
                0123456789 _+-.,!@#$$%^&* ();\|/|<>"'
                12345 -98.7 3.141 .6180 9,000 +42
                555.123.4567      800-555-2468
                foo@demo.net     bar.ba@test.co.uk
                www.demo.com     http://foo.co.uk/
                http://regexr.com/foo.html?q=bar
                https://mediatemple.net
```



**Result:** ⚡ regex result = 0123456789;555.123.4567;800-555-2468

## Replace

- Replace a specific text string within a variable; or
- Replace text matching a regular expression within a variable. (To learn more, see [What is a regular expression?](#))

### Using the REPLACE command

- 1 Enter the name of the variable in which you want to replace text
- 2 Enter the specific text you want to replace (free text and/or values copied from different variables); **or**  
Enter the regular expression that represents the text you want to replace
- 3 Enter the replacement text (free text and/or values copied from different variables); **and**  
Choose which instance(s) of the matching text you want to replace



### EXAMPLE

`⚡ quote` = I think, therefore I am.

**aB Replace** [X]

In the variable:  
quote

Replace the text  
think

Replace the text matching the regular expression: [Test...](#)

Ignore letter case

With:  
breathe

Replace all  
 Replace first  
 Replace last

[i] [OK] [Cancel]

**Result:** `⚡ quote` = I breathe, therefore I am.

## Mathematics

Perform simple mathematical calculations (using constants and/or values copied from other variables) and place the result into a new or existing variable.

### Using the MATHEMATICS command

**1** Enter the 2 values on which you want to perform the calculation and select the operation you wish to perform. The available operations are:

- Addition
- Subtraction
- Multiplication
- Division
- Modulus (returns the remainder obtained when dividing the first value by the second)

**2** Enter the name of the variable into which you'd like Leo to place the result



#### NOTE

If one or both of the values entered is non-numeric, Leo will return an empty result in the variable you have specified. (Note that numbers including currency symbols are treated as **non**-numeric.)

You can use the **CHECK TYPE** command to validate that variable values are numeric prior to using them in mathematical calculations.

## Evaluate Expression

- Perform complex mathematical or textual operations, for example:
  - Evaluate the expression:  $15 * 16 * 2$
  - Result = 480
- Evaluate the validity of complex mathematical or logical expressions to obtain a result of **True** or **False**, for example:
  - Evaluate the expression:  $15 * 16 * 2 = 500$
  - Result = False

## Using the EVALUATE EXPRESSION command

**fx Evaluate expression**

Evaluate the expression:

Return the result in variable:

**Note:**

When using a constant value (number/text), add apostrophes around the value (e.g. \$VarName\$ = 'A').

To use a variable as a numeric value, add number (#) signs around the variable name (e.g. #VarName# = '2').

OK Cancel

- 1 Enter the operation you want to perform or the expression you want to evaluate for validity
- 2 Enter the name of the variable in which you want Leo to place the result

### Constants:

- To utilize a text string as a constant, place it within single quotes, for example:
  - 'Franklin D. Roosevelt'
  - If the text string includes a single quote, precede it with another single quote, for example: 'Franklin D. Roosevelt''s New Deal'

- To utilize a numeric constant, simply enter the number without any additional characters, for example: 10000
  - **Exception:** To treat a number as a text string (as opposed to a number), place it within single quotes
  - Decimals **are** supported within numeric values, for example: 3.141519
  - Commas and currency characters are **not** supported within numeric values

### Variables:

- To utilize a variable as text string, place the variable name within dollar signs, for example: \$TextVariable\$
- To utilize a variable as a numeric value, place the variable name within number signs, for example: #NumericVariable#

### Operators:

Supported arithmetic operators:

- + (addition)
- - (subtraction)
- \* (multiplication)
- / (division)
- % (modulus)  
*returns the remainder obtained when dividing the first value by the second*



#### NOTE

Standard mathematical order of operations applies. Use parentheses to force order of precedence. For example:

- $120/10 + 2 = 14$
- $120/(10+2) = 10$

Supported Boolean operators:

- AND
- OR

## Supported comparison operators:

- = (equals)
- <> (does not equal)
- < (is less than)
- > (is greater than)
- <= (is less than or equal to)
- >= (is greater than or equal to)
- LIKE (similar to *equals*, but permits the use of wildcard characters)
  - Valid wildcard characters are \* and % (and can be used interchangeably)
    - If the string in a LIKE clause contains a \* or %, those characters should be enclosed in brackets, for example: 25 [%]
    - If a bracket is in the clause, each bracket character should be enclosed in brackets, for example: [ [ ] or [ ] ]
  - A wildcard is allowed at the start of a pattern, the end of a pattern, or both. For example:
    - \$name\$ LIKE 'Franklin\*' returns **True** when ⚡ name = Franklin D. Roosevelt
    - \$name\$ LIKE '\*Franklin' returns **True** when ⚡ name = Benjamin Franklin
    - \$name\$ LIKE '\*Franklin\*' returns **True** when ⚡ name = John Adams & Benjamin Franklin signed the Declaration of Independence
  - A wildcard is **not** allowed in the middle of a pattern, for example: \$name\$ LIKE 'Frank\*lin'

**NOTE**

Letter case is not considered when expressions are evaluated for validity. For example: \$name\$ = 'franklin' returns **True** when ⚡ name = Franklin

## String operator:

- Use the + character to concatenate a text string. For example, 'Benjamin' + ' ' + \$lastname\$ returns **Benjamin Franklin** when ⚡ lastname = Franklin



## Concatenation &amp; order of precedence:

- You can create complex expressions by concatenating clauses using the **AND** and **OR** operators
  - The AND operator has precedence over other operators
  - You can use parentheses to group clauses and force precedence, for example:  
`($firstname$ = 'Theodore' OR $firstname$ = 'Franklin')  
AND $lastname$ = 'Roosevelt'`

## Functions:

**LEN**

Description	Gets the length of a string (including spaces)
Syntax	LEN( <i>expression</i> )
Arguments	<i>expression</i> = the string to be evaluated
Example	LEN(\$name\$) returns <b>21</b> when <code>name</code> = Franklin D. Roosevelt

**IIF**

Description	Gets one of two values depending on the result of a logical expression
Syntax	IIF( <i>expression</i> , <i>if_true</i> , <i>if_false</i> )
Arguments	<i>expression</i> = the expression to evaluate <i>if_true</i> = the value to return if the expression is true <i>if_false</i> = the value to return if the expression is false
Example	IIF(#year# = 1776, 'Benjamin Franklin', 'Franklin D. Roosevelt') returns <b>Franklin D. Roosevelt</b> when <code>year</code> = 1948

**TRIM**

Description	Removes blank spaces (including <Space> <Tab> and <Enter>) from both ends of an expression
Syntax	TRIM( <i>expression</i> )
Arguments	<i>expression</i> = the expression to trim
Example	TRIM(\$name\$) returns <b>Theodore Roosevelt</b> when <code>name</code> = <Space><Tab>Theodore Roosevelt<Enter>

**SUBSTRING**

Description	Gets a substring of a specified length, starting at a specified point in the string
Syntax	<code>SUBSTRING(<i>expression</i>, <i>start</i>, <i>length</i>)</code>
Arguments	<p><i>expression</i> = the source string</p> <p><i>start</i> = the numbered position that the substring starts (within the source string)</p> <p><i>length</i> = the length of the desired substring</p>
Example	<p><code>SUBSTRING(\$trivia fact\$,13,28)</code> returns <b>Roosevelt died in April 1945</b> when <code>⚡ trivia fact</code> = Franklin D. Roosevelt died in April 1945, before the end of WWII.</p>

## Split

Divide the contents of a variable into 2 parts and place each part into a separate variable. (This is often called *parsing* in computer-speak.)

You can choose to split the variable:

- At the occurrence of a specific character (called a delimiter); or
- At a numbered location within the variable (i.e., by character position)

### Using the SPLIT command

- 1 Enter the name of the variable you want to split
- 2 Enter the delimiter or numbered character position at which you want to split the variable (can include free text and/or values copied from different variables)
- 3 Enter the names of the variables into which you'd like Leo to place the 2 parts of the original variable
- 4 Indicate if you want to remove any blank spaces at the beginning and end of these variables (includes: <Space> <Tab> and <Enter>)



## NOTE

How exactly is the variable split?

- **Delimiter:**
  - 1st variable = the part of the original variable preceding the delimiter
  - 2nd variable = the part of the original variable following the delimiter
  - The delimiter is erased
- **Character position:**
  - 1st variable = the part of the original variable up to **and including** the character position specified
  - 2nd variable = the part of the original variable following the delimiter
  - Spaces are included when counting character position



## TIP

Using **SPLIT** in combination with **LOOP**

**SPLIT** is one of the many advanced commands that is often used in combination with **LOOP**. When using **SPLIT** within a loop, it is often useful to split the original variable as follows:

1. Place one of the parts into a new variable
2. Overwrite the "pre-split" value of the original variable with the "post-split" value

Learn more about the **LOOP** command



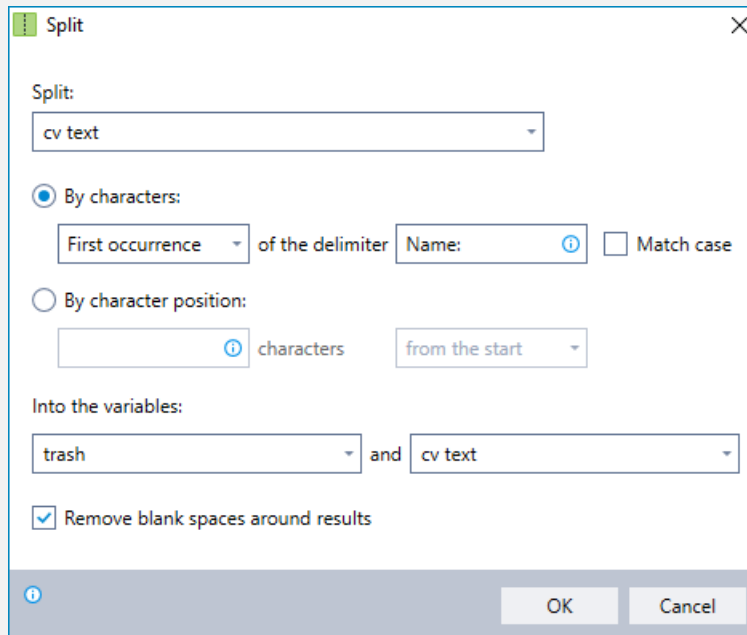
## EXAMPLE

Extracting information from a long text file

Let's say you have all the text from a new employee's CV stored in a variable named `cv text`. You'd like to extract only the necessary details and input them into your HR system. To start, extract the employee's name as follows:

1. Place everything prior to the employee name into a variable named `⚡ trash`. Keep the employee name and everything following it in `⚡ cv text`.

```
⚡ cv text = 1234 Main Street; Independence,
             Missouri 64052; Home - (816) 555-1234;
             Mobile - (816) 444-5678; Name: Harry
             S. Truman; Position sought: POTUS;
             Prior Experience: Senator, Vice
             President; Favorite expression: The
             buck stops here.
```



**Result:**

```
⚡ trash = 1234 Main Street; Independence,
           Missouri 64052; Home - (816) 555-1234;
           Mobile - (816) 444-5678;
```

```
⚡ cv text = Harry S. Truman; Position sought:
             POTUS; Prior Experience: Senator, Vice
             President; Favorite expression: The
             buck stops here.
```

- Place the employee name into a variable named `ename` and everything after it into `trash`.

#### Result:

`ename` = Harry S. Truman

`trash` = Position sought: POTUS; Prior  
Experience: Senator, Vice President;  
Favorite expression: The buck stops  
here.

## Get Array Data

Obtain specific information from a variable containing a string of items (an "array"). You can choose to obtain the following types of information:

- The value of a specific item in the array
- The total number of items in the array
- The result of a mathematical calculation performed on the array:
  - Average
  - Maximum Value
  - Minimum Value
  - Sum



### NOTE

In order for this command to work properly, all the items in the variable must be delimited (i.e., separated) in a consistent way.

## Using the GET ARRAY DATA command

The screenshot shows a dialog box titled "Get array data" with a close button (X) in the top right corner. The dialog is divided into several sections:

- Section 1:** "From the array stored in:" followed by a dropdown menu containing the text "Type a variable name".
- Section 2:** Three radio button options:
  - "Where items are delimited by:" followed by a text input field.
  - "Where items are preceded by:" followed by a text input field.
  - "and followed by:" followed by a text input field.
- Section 3:** Three radio button options:
  - "Get the value of item #" followed by a text input field.
  - "Get the total number of items"
  - "Get the average"
- Section 4:** "Return the result in variable:" followed by a dropdown menu containing the text "Type a variable name".
- Section 5:** "Error handling" with a dropdown arrow and an information icon (i).

At the bottom of the dialog, there is an information icon (i) on the left and "OK" and "Cancel" buttons on the right.

- 1 Enter the name of the variable from which you want to obtain data
- 2 Enter the delimiter or the characters that appear before and after each item in the array
- 3 Choose the type of data you want to obtain. For a mathematical calculation, select the operation.
- 4 Enter the name of the variable into which you'd like Leo to place the data obtained
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).





### CAUTION

In order to perform a mathematical calculation, all values in the array must be numeric. If one or more of the values in the array is non-numeric, Leo will return an empty result in the variable you have specified. (Note that numbers including currency symbols are treated as **non-numeric**.)

You can use the **CHECK TYPE** command to validate that variable values are numeric prior to using them in mathematical calculations.



### EXAMPLE

Let's say you run an e-commerce internet website. At the end of each day, you read the amounts of all the day's orders into a variable called `⚡ daily orders`. You'd like to extract the following information: (1) total number of orders; (2) average order amount; and (3) largest order amount.

`⚡ daily orders` = 65.00; 189.95; 645.76; 39.05; 254.36; 98.89; 369.56

**Result:** `⚡ number of orders` = 7

Get array data

From the array stored in:  
daily orders

Where items are delimited by: ;

Where items are preceeded by:

and followed by:

Get the value of item #

Get the total number of items

Get the average

Return the result in variable:  
average order

OK Cancel

**Result:** ⚡ average order = 237.51

Get array data

From the array stored in:  
daily orders

Where items are delimited by: ;

Where items are preceeded by:

and followed by:

Get the value of item #

Get the total number of items

Get the maximum value

Return the result in variable:  
largest order

OK Cancel

**Result:** ⚡ largest order = 645.76

## Get Table Data

Obtain specific information from a variable containing a table. You can choose to obtain the following types of information:

- The value of a specific item (identified by the item's row and column number)
- The total number of items in the table
- The total number of rows in the table
- The total number of columns in the table



### TIP

The **GET TABLE DATA** command works especially well with Excel and CSV files and in combination with Leo's [Excel Commands](#).

## Using the GET TABLE DATA command

Get table data
✕

From the table stored in:

1

Where columns are delimited by:

and rows are delimited by:

2

Get the value of item at row #

and column #

Get the total number of items

Get the total number of rows

Get the total number of columns

3

Return the result in variable:

4

**Example:**  
A1, B1, C1, D1  
A2, B2, C2, D2  
A3, B3, C3, D3

For the table above where columns are delimited by commas and rows are delimited by line breaks, the value of item in row #2 and column #4 is D2.

?

OK
Cancel

- 1 Enter the name of the variable from which you want to obtain data
- 2 Enter the delimiters that separate each column and each row
- 3 Choose the type of data you want to retrieve
- 4 Enter the name of the variable into which you'd like Leo to place the retrieved data



### EXAMPLE

Let's say you run an e-commerce website. At the end of each day, you read the details of all the day's orders into a CSV file and place the contents into a variable called `daily order table`. You'd like to know the total number of orders.

This scenario is an example of one in which a combination of two Advanced Commands (**GET TABLE DATA** and **MATHEMATICS**) work beautifully together.

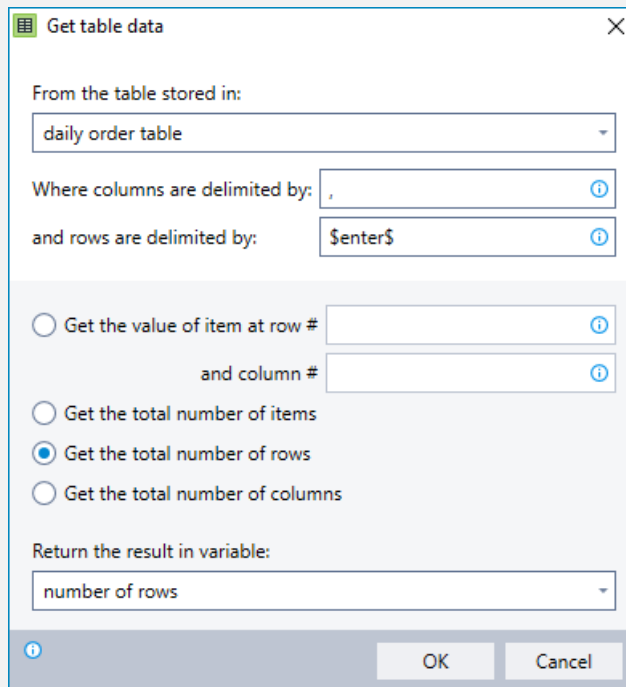
Here is how the data would look formatted as a table:

Order Number	Order Total	Number of Items
34567	350.00	10
34568	785.00	15
34569	649.00	14
34570	134.00	1
34571	100.00	1
Totals	2018.00	41

And here's how it would be read from a CSV file into a variable:

```
⚡ daily order table =Order Number, Order Total, Number of
Items
34567, 350.00, 10
34568, 785.00, 15
34569, 649.00, 14
34570, 134.00, 1
34571, 100.00, 1
Total, 2018.00, 41
```

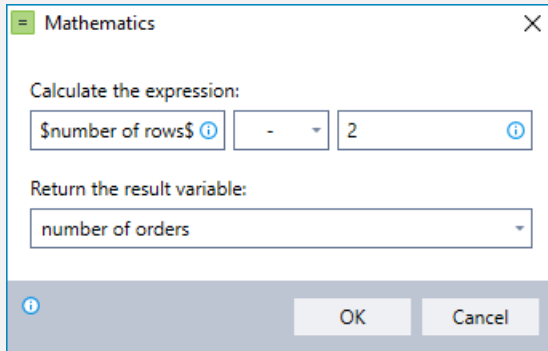
1. Obtain the total number of rows in the table



**Result:** ⚡ number of rows = 7

- For more information about using special characters such as <Space>, <Enter>, and <Tab> as delimiters, see [SET VALUE](#)

Adjust the total number of rows to obtain the number of orders (accounting for the header row and totals row).



**Result:** ⚡ number of orders = 5

## Get ASCII Character

There are times when you need Leo to use a special character, but the character does not appear on the keyboard. **THE GET ASCII CHARACTER** command allows you retrieve the character you need and place it into a variable to use later.



### NOTE

Today's Trivia: What does ASCII mean?

**ASCII** stands for **American Standard Code for Information Exchange**. It is standard for encoding characters in numeric format so they can be understood by computers, telecommunications equipment, and other devices. Originally developed for use with teletypes, today it is often used as a method for representing "non-printing" characters (those that do not appear on a keyboard).

## Using the GET ASCII CHARACTER command

 A screenshot of a dialog box titled 'Get ASCII character'. It has a search icon and a close button (X) in the top right. There are two radio buttons: 'ASCII code 13 (Enter)' (selected) and 'Type ASCII code to return matching character:'. Below the second radio button is a text input field with an information icon (i) on the right. Below that is a section 'Return the result in variable:' with a dropdown menu containing the text 'Type a variable name' and an information icon (i) on the right. At the bottom, there is an information icon (i) on the left and 'OK' and 'Cancel' buttons on the right.

- 1 Select the character that you want to place into a variable:
  - Use the shortcut provided for the <Enter> character; **or**
  - Enter the ASCII code for the character you need
    - For a list of all available ASCII codes, see [www.asciitable.com](http://www.asciitable.com)
    - Note that Leo does not support extended ASCII codes
- 2 Enter the name of the variable into which you'd like Leo to place the character



## EXAMPLE

At the end of each day, you download data from a web application for further analysis. The web application uses <Page Break> to separate individual records, so you also need Leo to separate the data at each <Page Break> when it enters the data into your system. How can you make this happen?

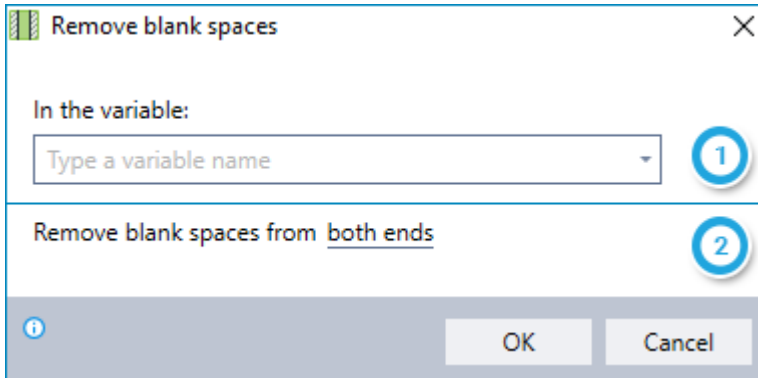
1. Use the **GET ASCII CHARACTER** command to place the <Page Break> character (ASCII code 12) into a variable called `⚡ page break`.
2. Use the **SPLIT** command with the `⚡ page break` variable as the delimiter to parse the downloaded data.



## Remove Blank Spaces

Remove blank spaces (including <Space> <Tab> and <Enter>) from the beginning, end, or both ends of the variable you specify.

### Using the REMOVE BLANK SPACES command



- 1 Enter the name of the variable from which you want to remove blank spaces
- 2 Select whether you'd like to remove blank spaces from the beginning, end, or both ends of the variable



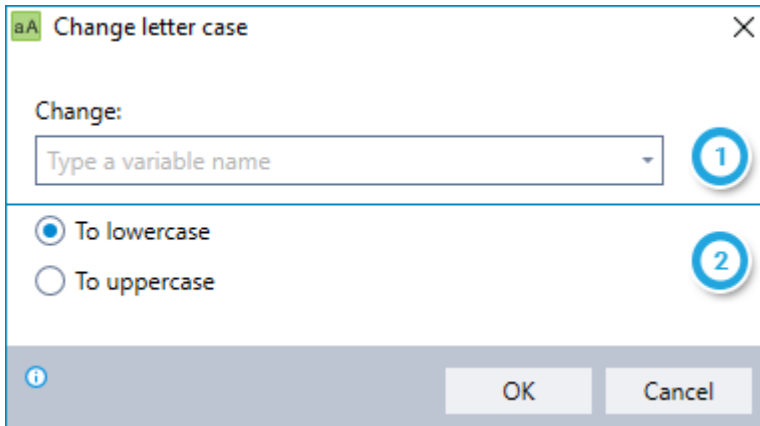
#### TIP

**REMOVE BLANK SPACES** is particularly useful for "cleaning up" data before entering it in other applications.

## Change Letter Case

Change the text within a variable to all uppercase or all lowercase letters.

### Using the CHANGE LETTER CASE command



- 1 Enter the name of the variable for which you'd like to change letter case
- 2 Select whether you'd like to change to all lowercase or all uppercase letters



#### TIP

The **CHANGE LETTER CASE** command changes **all** the text within the variable to either upper or lowercase. If you'd like to capitalize the just the first letter of certain words within a variable, using the **SPLIT** command in combination with **CHANGE LETTER CASE** can be especially handy.

## Check Type

Check whether the value of a variable is numeric or textual.

### Using the CHECK TYPE command

- 1 Enter the name of the variable whose value you'd like to check; *and* Choose the type of value you'd like to check for (numeric or textual)
- 2 Enter the name of the variable into which you'd like Leo to place the result. (The result will be either TRUE or FALSE, as applicable.)



### NOTE

Numbers including currency symbols are treated as non-numeric.



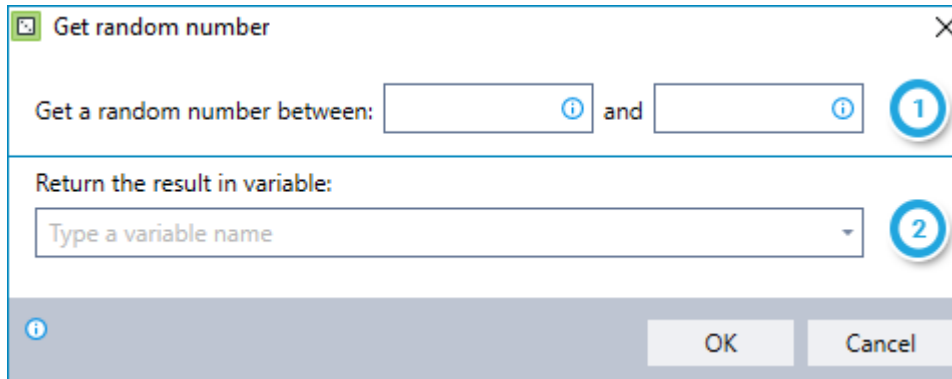
### TIP

The **CHECK TYPE** command can be used to ensure that variable values are numeric before they are used in Advanced Commands that perform mathematical calculations (e.g., **MATHEMATICS**, **GET ARRAY DATA**).

## Get Random Number

Generate a random number within the range you specify and store it in a variable.

### Using the GET RANDOM NUMBER command



- 1 Enter the range within which you'd like Leo to generate a random number
- 2 Enter the name of the variable into which you'd like Leo to place the random number generated



#### TIP

##### Why generate a random number?

**GET RANDOM NUMBER** can be useful in a number of circumstances, for example:

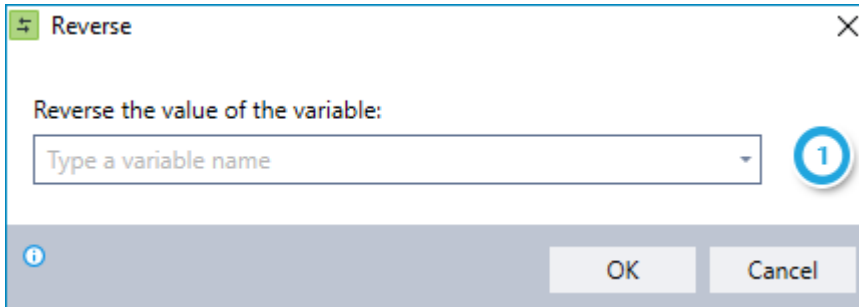
Creating single-use login codes for other applications

Creating unique record identifiers (e.g., tagging transactions)

## Reverse

Reverse the value of a variable, character-by-character. This can be particularly useful when working with applications that do not natively support right-to-left languages.

### Using the REVERSE command

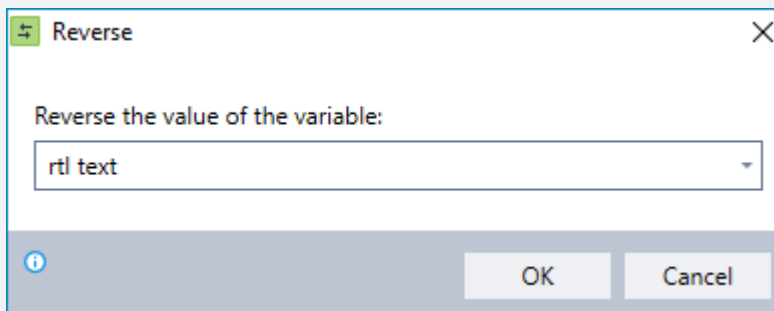


- 1 Enter the name of the variable whose value you'd like to reverse



#### EXAMPLE

```
⚡ rtl text = ydenneK .F nhoJ
```

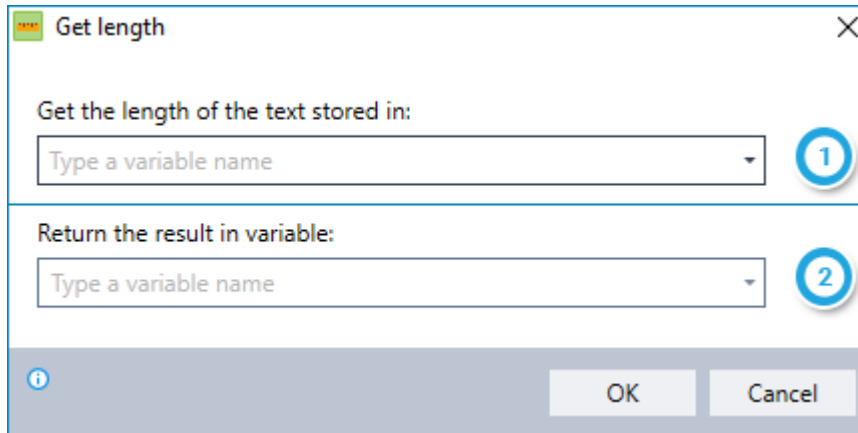


**Result:** ⚡ rtl text = John F. Kennedy

## Get Length

Count the characters (**including spaces**) of the value stored within a variable.

### Using the GET LENGTH command



- 1 Enter the name of the variable in which you'd like to count the number of characters
- 2 Enter the name of the variable into which you'd like Leo to place the result



#### EXAMPLE

##### Why use GET LENGTH?

This command can be especially helpful when performing validations to ensure that data was entered properly.

## Extract Numeric Values

Extract numbers from a variable that contains a mix of text and numbers.

### Using the **EXTRACT NUMERIC VALUES** command

- 1 Enter the name of the variable from which you want to extract numbers
- 2 Specify how you'd like the results to be presented:
  - Indicate if you'd like the formats of the numbers from the original variable to be maintained in the output
  - Choose the delimiter you'd like to use in the output to separate the numbers extracted from the original variable
- 3 Enter the name of the variable into which you'd like Leo to place the extracted numbers
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## NOTE

### How exactly are numbers extracted?

Leo recognizes numbers as separate from text by looking for specific characters within a string of data. In order for a number to be properly identified, it must be enclosed in quotes (either single or double) or separated from the surrounding text by one of the following characters:

- <Space>
- Comma ( , )
- Semicolon ( ; )
- Pipe ( | )

Leo uses the same characters to recognize numbers as separate from each other – with the **important exception of commas** (because many numbers utilize commas as part of their formatting).

**Try it out:** To ensure that numbers will be extracted as you expect, try some test data. Simply click the **NUMERIC EXTRACTOR TESTER** link from within **THE EXTRACT NUMERIC VALUES** command.



## NOTE

### A word about currency symbols

For purposes of the **EXTRACT NUMERIC VALUES** command, Leo will recognize a number immediately preceded by a currency symbol (e.g., \$, €, £, ¥, etc.) as a numeric value.

If you elect to preserve original number formats (and there is no space between a number and the currency symbol preceding it), the currency symbol will be preserved in the output.





## EXAMPLE

Let's say you run an e-commerce website. At the end of each day, you read a list of all the items ordered into a variable called `daily items ordered`. The downloaded data includes *item description*, *stock number*, *unit price*, and *quantity ordered*. For order fulfillment purposes, you need only the *stock number*, *unit price*, and *quantity ordered*.

```
daily items ordered = Item Description, Stock Number, Unit
                    Price, Quantity Ordered
                    Tennis Racquet, 6527895, €65.00, 10
                    Tennis Shoes, 6387429, €89.50, 15
                    Tennis Balls, 6572369, €0.85, 90
                    Tennis Shorts, 6354789, €14.00, 20
```

# Extract numeric values

Get all numeric values from the text stored in:

daily items ordered

Numbers will be extracted if separated by: [Space], [, ], [ ] or if placed within quotes

Preserve original number formats

Delimiter:

;

Return the results into variable:

daily items numeric

[Numeric extractor tester...](#)

▼ Error handling ⓘ

OK Cancel

### Result:

```
daily items numeric = 6527895;€65.00;10;6387429;€89.50;15;
                    6572369;€0.85;90;6354789;€14.00;20
```

# CHAPTER 2: Flow Commands

In this chapter:

If Else .....	51
Complex If Else .....	55
Multi-Value If Else .....	58
Loop .....	60
Loop: Break .....	62
Loop: Restart .....	63
Loop Items .....	64
Pause .....	66
Group .....	67

## If Else

Compare the value of a variable with another specified value to determine whether a condition is TRUE or FALSE. Based on the outcome, direct the logical flow of the wizard along two or more different paths (i.e., if the condition is TRUE, follow Path A; if the condition is FALSE, follow Path B.)

### Using the IF ELSE command

#### Step #1 - Define the condition

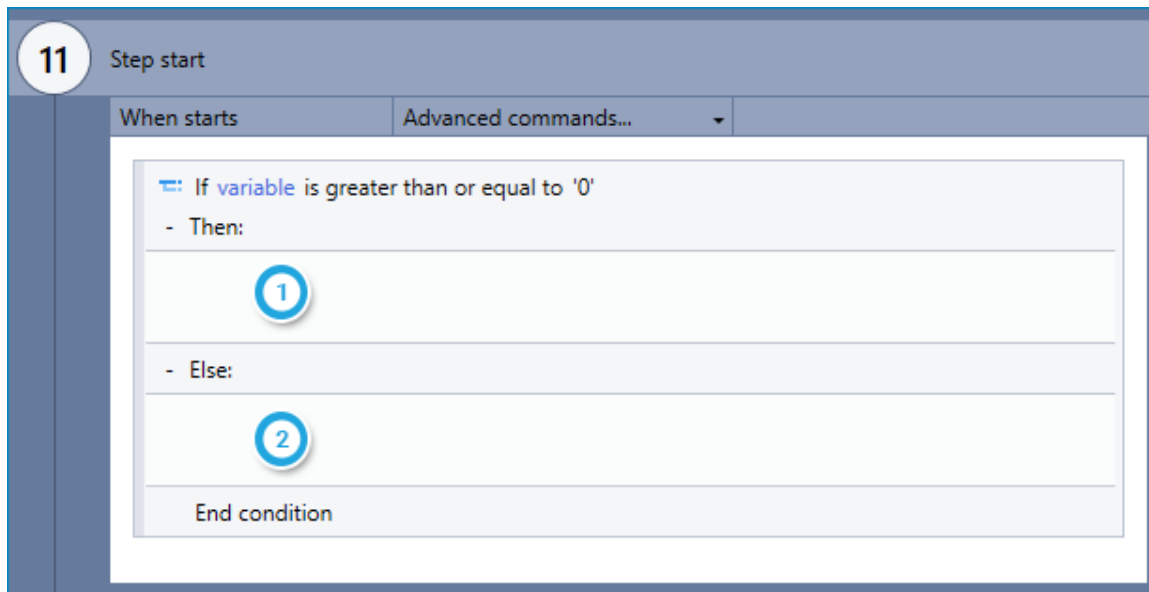
The first step in using the **IF ELSE** command is to define the condition (i.e., set up a comparison).

- 1 Enter the name of the variable whose value you wish to compare with another value
- 2 Select the type of comparison you wish to perform:
  - equals (with options to ignore letter case/use wildcards)
  - contains (with option to ignore letter case)
  - match **regular expression** (with option to ignore letter case)
  - is greater than
  - is greater than or equal to
  - is less than
  - is less than or equal to
  - begins with (with option to ignore letter case)
  - ends with (with option to ignore letter case)
  - is empty
  - is defined
- 3 Enter the value with which you wish to compare the variable's value (can be entered manually or copied from values stored in variables)

- 4 Indicate if you wish to perform a reverse comparison (e.g., `variable IS NOT` greater than or equal to 0)
- 5 (Optional) Use **ELSE IF** to set up multiple comparisons if you need to define 3 or more possible outcomes. To learn more, see [ELSE IF](#).

## Step #2 - Define the actions

Upon adding the **IF ELSE** command to your wizard, you will notice that it becomes an empty "container" within the Editor Pane, waiting for you to fill it with instructions:



- 1 Enter the action(s) the wizard should take if the condition is TRUE
  - You can do this by dragging the required Advanced Command(s) directly into the container
- 2 Enter the action(s) the wizard should take if the condition is FALSE



### EXAMPLE

Let's say you want your wizard to check if the folder `c:\my folder` exists.

- If it does (the condition is TRUE), you want to copy a file to it
- If it doesn't (the condition is FALSE), you want the wizard to skip to the next step

A combination of a few Advanced Commands will help you get this done:

1. Use the **DOES FOLDER EXIST** command to check if the folder exists

```
Check if the folder c:\my folder exists and set folder check to TRUE/FALSE accordingly
```

2. Based on the outcome of the check, use the **IF ELSE** command to direct the wizard on one of the two possible paths

```
If folder check equals (ignore case) 'TRUE'
- Then:
  Copy file c:\documents\my file.txt to c:\my folder
- Else:
  Go to step 12
End condition
```

### Else if

If you need direct the flow of your wizard among 3 or more different logical paths, you can use the **ELSE IF** option to add additional comparisons. The resulting logic (for 3 possible paths) might look something like this:

- If the result of Comparison #1 is TRUE → follow Path A
- If the result of Comparison #1 is FALSE → perform Comparison #2
  - If the result of Comparison #2 is TRUE → follow Path B
  - If the result of Comparison #2 is FALSE → follow Path C

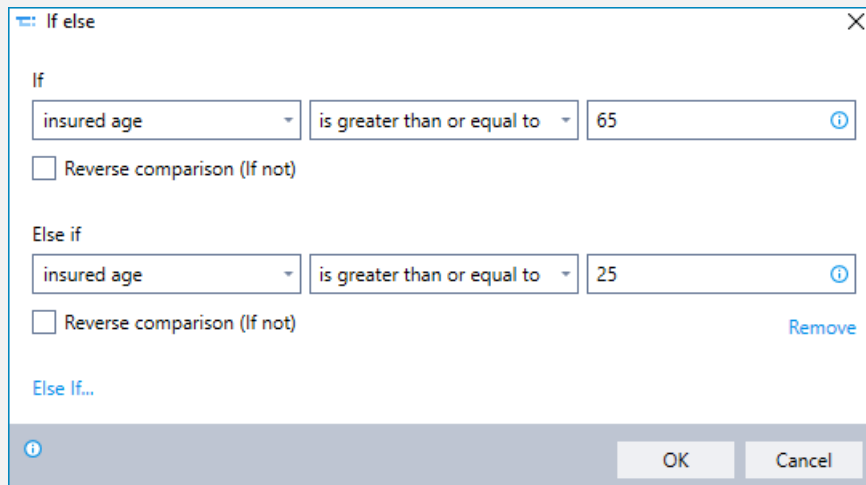


### EXAMPLE

Your auto insurance company assigns rates for its collision policies using different rate tables based on the insured's age:

Insured's Age	Rate Table
65 +	RT1
25 - 65	RT2
Younger than 25	RT3

1. Use the **IF ELSE** command with one **ELSE IF** option to perform the required comparisons and direct the wizard among the 3 possible paths



2. On each of the paths, use the **SET VALUE** command to store the correct rate table for the insured into a variable called **rate table**



## Complex If Else

Compare the values of one or more variables with other specified values to determine whether one or more conditions are TRUE or FALSE. Based on the outcome, direct the logical flow of the wizard along one of two different paths (i.e., if any/all of the conditions are TRUE, follow Path A; if any/all of the conditions are FALSE, follow Path B.).

### Using the COMPLEX IF ELSE command

#### Step #1 - Define the conditions

The first step in using the **COMPLEX IF ELSE** command to is define the conditions (i.e., set up comparisons).

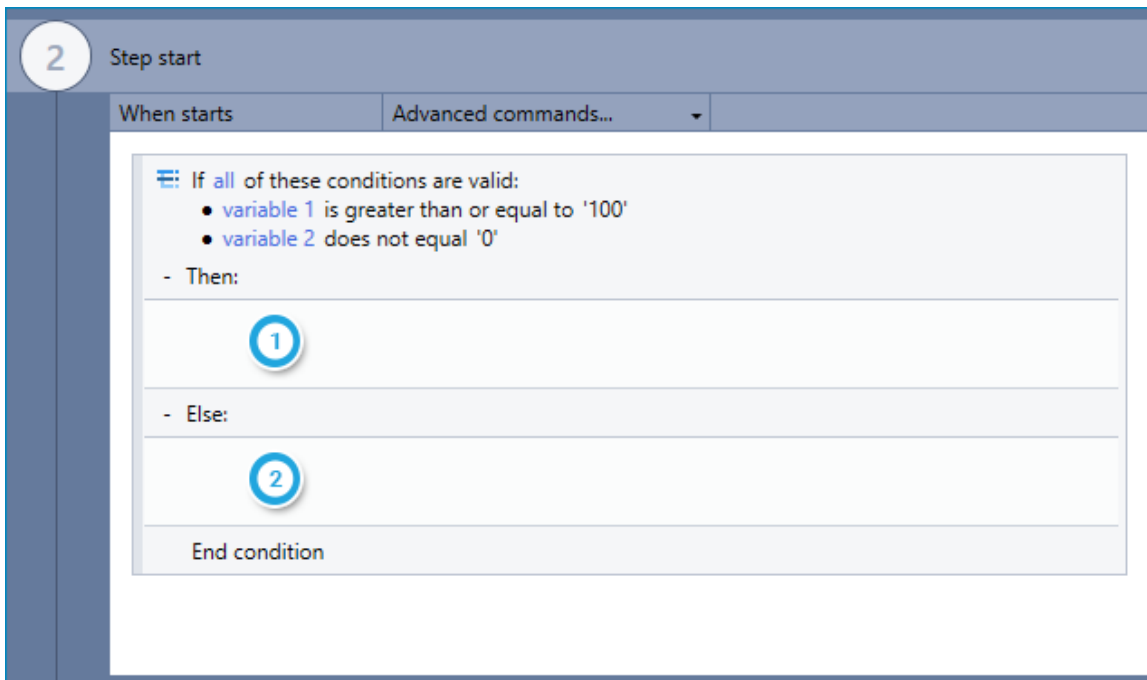
- 1 Choose whether **any** or **all** of the conditions must be true in order for the overall outcome to be TRUE
- 2 For each condition:
  - Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
  - equals (with options to ignore letter case/use wildcards)
  - contains (with option to ignore letter case)
  - match **regular expression** (with option to ignore letter case)
  - is greater than

- is greater than or equal to
- is less than
- is less than or equal to
- begins with (with option to ignore letter case)
- ends with (with option to ignore letter case)
- is empty
- is defined

- 4 Enter the value with which you wish to compare the variable's value (can be entered manually or copied from values stored in variables)
- 5 Indicate if you wish to perform a reverse comparison (e.g., **variable IS NOT** greater than or equal to 0)
- 6 Add/remove conditions as required

### Step #2 - Define the actions

Upon adding the **COMPLEX IF ELSE** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:





- 1 Enter the action(s) the wizard should take if the overall outcome is TRUE
  - You can do this by dragging the required Advanced Command(s) directly into the container
- 2 Enter the action(s) the wizard should take if the overall outcome is FALSE

## Multi-Value If Else

Compare the value of a variable with one or more other values to determine whether each condition is TRUE (i.e., the values are equal) or FALSE (i.e., the values are unequal). Based on the outcome, direct the logical flow of the wizard along two or more different paths, for example:

- If Condition #1 is TRUE → follow Path A
- If Condition #2 is TRUE → follow Path B
- If Condition #3 is TRUE → follow Path C

### Using the MULTI-VALUE IF ELSE command

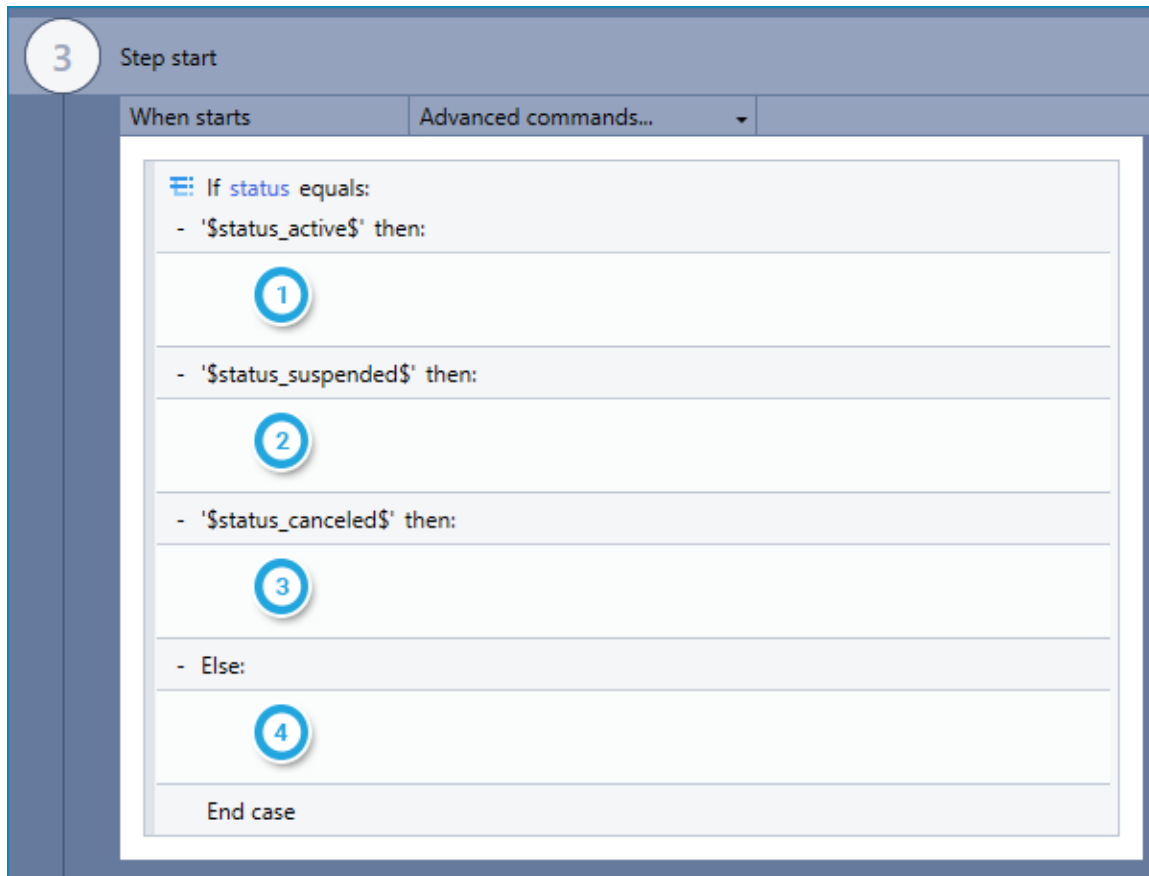
#### Step #1 - Define the conditions

The first step in using the **MULTI-VALUE IF ELSE** command is to define the conditions (i.e., set up comparisons).

- 1 Enter the name of the variable whose value you wish to compare with other values
- 2 For each condition, enter the value with which you wish to compare the variable's value (can be entered manually or copied from values stored in variables)
- 3 Add/remove conditions as required

#### Step #2 - Define the actions

Upon adding the **IF ELSE** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



- 1 Enter the action(s) the wizard should take if Condition #1 is TRUE
  - You can do this by dragging the required Advanced Command(s) directly into the container
- 2 Enter the action(s) the wizard should take if Condition #2 is TRUE
- 3 Enter the action(s) the wizard should take if Condition #3 is TRUE
- 4 Enter the action(s) the wizard should take if all of the specified conditions are FALSE

## Loop

Repeat an action or series of actions for as long as any/all of the defined conditions are TRUE.

- Each condition consists of a comparison between the value of a variable and another specified value
- Once the defined condition(s) are no longer TRUE, the wizard exits the loop and moves on

### Using the LOOP command

#### Step #1 - Define the conditions

The first step in using the **LOOP** command is to define the conditions (i.e., set up comparisons).

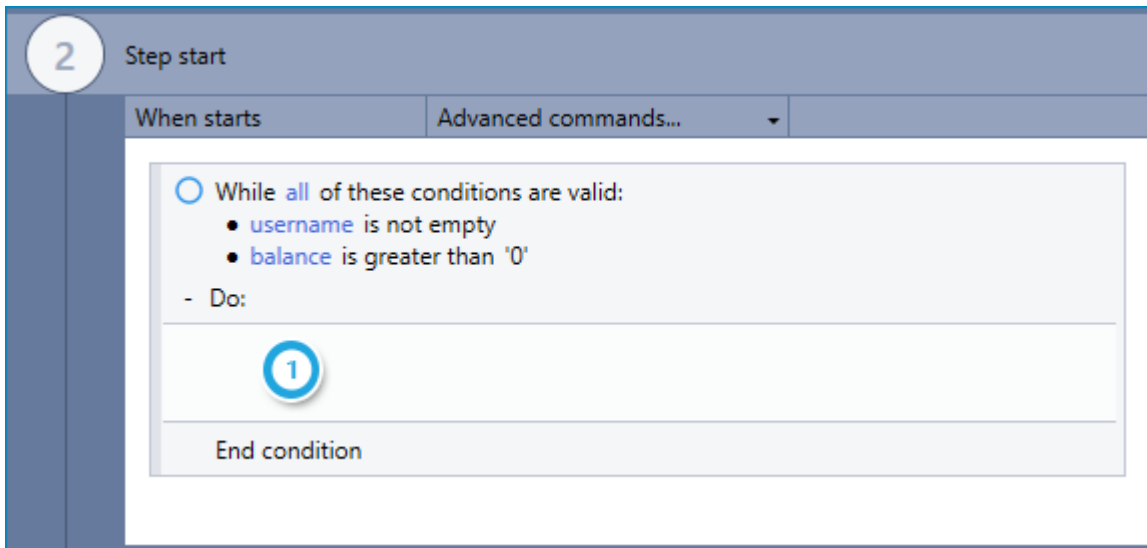
- 1 Choose whether **any** or **all** of the conditions must be true in order for the loop to continue
- 2 For each condition:  
Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
  - equals (with options to ignore letter case/use wildcards)
  - contains (with option to ignore letter case)
  - match **regular expression** (with option to ignore letter case)
  - is greater than
  - is greater than or equal to

- is less than
- is less than or equal to
- begins with (with option to ignore letter case)
- ends with (with option to ignore letter case)
- is empty
- is defined

- 4 Enter the value with which you wish to compare the variable's value (can be entered manually or copied from values stored in variables)
- 5 Indicate if you wish to perform a reverse comparison (e.g., **⚡ variable IS NOT** greater than or equal to 0)
- 6 Add/remove conditions as required

### Step #2 - Define the actions

Upon adding the **LOOP** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



- 1 Enter the action(s) the wizard should take while the defined condition(s) are TRUE
  - You can do this by dragging the required Advanced Command(s) directly into the container

## Loop: Break

Exit a **LOOP** before it reaches its defined conclusion (i.e., though the conditions for continuing the loop are still TRUE). This command can be especially useful when an event occurs that makes completing the loop unnecessary (for example, when a particular text is located partway through a file).

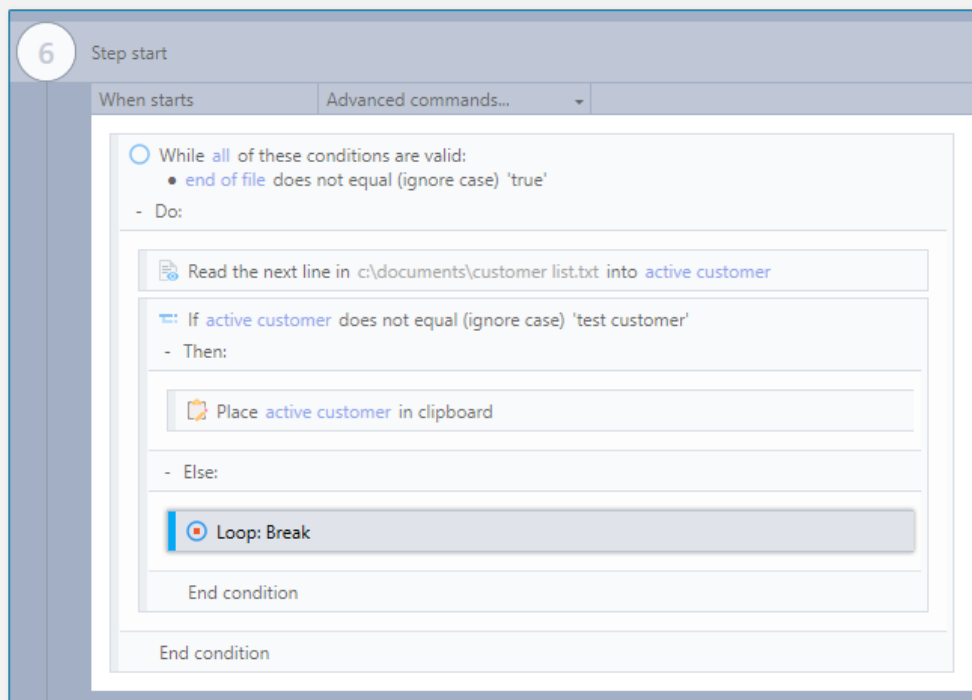


### NOTE

The **LOOP: BREAK** command can be used only within the following "container"-type Advanced Commands that use a looping mechanism:

- **LOOP**
- **LOOP ITEMS**
- **GET EMAIL MESSAGES**
- **ANALYZE DIGITAL PDF FILE**
- **DOCUMENT ANALYSIS**

It has no configurable options and can be added to a wizard simply by dragging it into the relevant command's container in the Editor Pane.



## Loop: Restart

Return to the first logical step of a **LOOP** (i.e., evaluating the condition(s) for continuing the loop) without completing the defined actions for the current item.

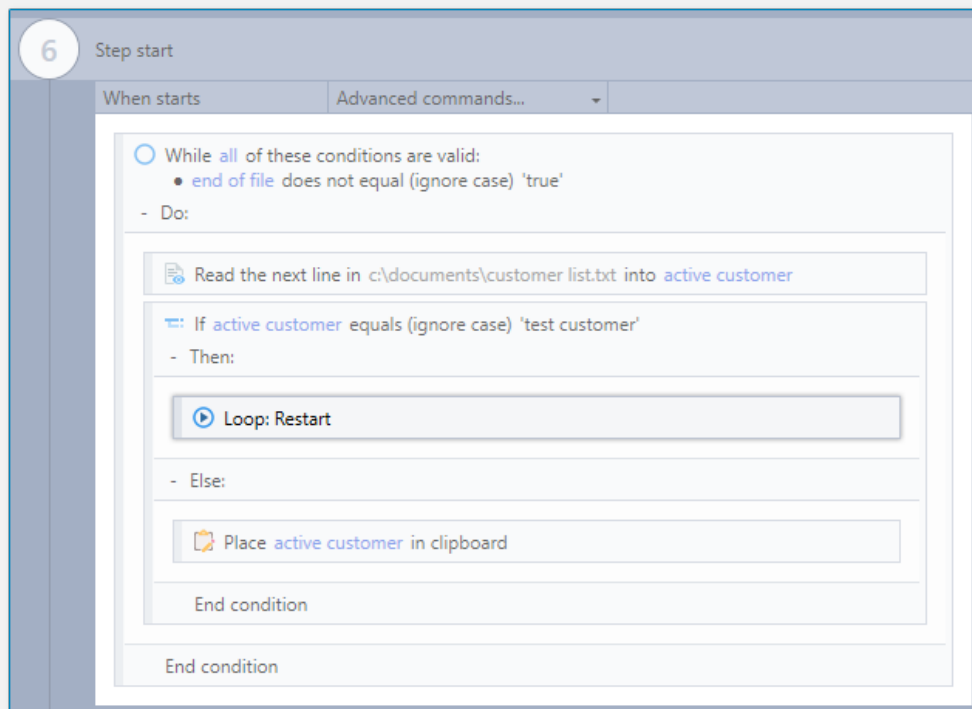


### NOTE

The **LOOP: RESTART** command can be used only within the following "container"-type Advanced Commands that use a looping mechanism:

- **LOOP**
- **LOOP ITEMS**
- **GET EMAIL MESSAGES**
- **ANALYZE DIGITAL PDF FILE**
- **DOCUMENT ANALYSIS**

It has no configurable options and can be added to a wizard simply by dragging it into the relevant command's "container" in the Editor Pane.



## Loop Items

For each item in a string of items stored in a variable (an "array"):

- Place the individual item into a new or existing variable; **and**
- Perform a specified action or series of actions

After completing the action(s) for the last item in the array, the wizard exits the loop and moves on.



### NOTE

In order for this command to work properly, all the items in the variable must be delimited (i.e., separated) in a consistent way.

## Using the LOOP ITEMS command

### Step #1 - Identify the array

The first step in using the **LOOP ITEMS** command is to identify the array and define a few settings.

Loop items
✕

Loop the items of the array stored in:

▼
1

Where items are delimited by:

i
2

Set each item in the variable:

▼
3

**Example:**  
 For the array `first,second,third,fourth`, where items are delimited by commas, the loop will occur 4 times.

i

OK

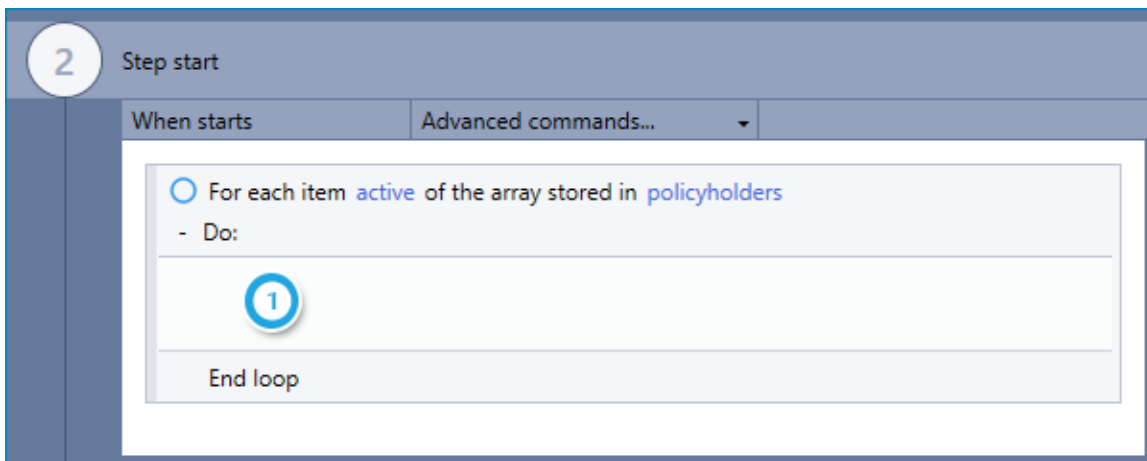
Cancel



- 1 Enter the name of the variable in which the array is stored
- 2 Enter the delimiter that separates each item in the array
- 3 Enter the name of the variable into which you'd like to place each individual item
  - **Why?** Since the action(s) in the loop will be performed on each item in the array (one at a time), it makes sense to first place the item into a variable, then perform the defined actions on the value of that variable.

### Step #2 - Define the actions

Upon adding the **LOOP ITEMS** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:

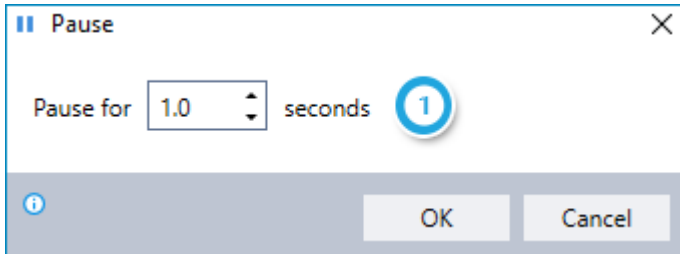


- 1 Enter the action(s) the wizard should take for each item in the array
  - You can do this by dragging the required Advanced Command(s) directly into the container

## Pause

Pause command execution for a specified time. This command is useful when the active application needs some time to accept data before it can move on.

### Using the PAUSE command



- 1 Select required pause time (in seconds)

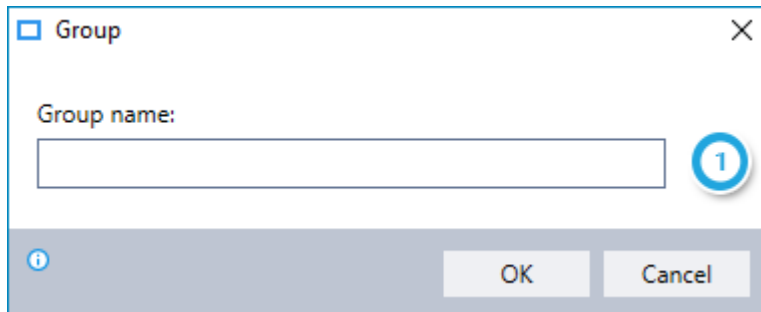
## Group

Group a series of commands into a single unit – making it easy to view, manage, and use.

- Use the [SET GROUP AS GLOBAL option](#) to enable universal availability and editing of a group throughout the wizard.

### Using the GROUP command

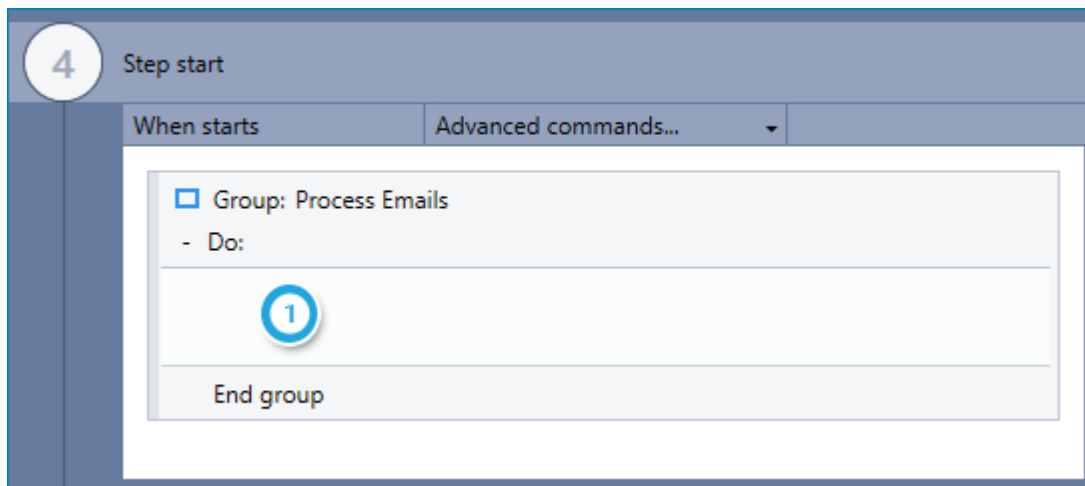
#### Step #1 - Name the group



- 1 Enter the name you would like to give the group

#### Step #2 - Define the actions

Upon adding the **GROUP** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



- 1 Enter the action(s) to include in the group
  - You can do this by dragging the required Advanced Command(s) directly into the container





### TIP

#### Creating groups in a snap

Often, you'll want to create groups from commands you've already added to the wizard. Good news... there are lots of ways to do just that. Choose the one you like best!

Select the commands you want to group and...

- Type <CTRL>+G
- Click the  button on the toolbar
- Right-click anywhere on the selected commands and click  Group

## SET GROUP AS GLOBAL option

If the group you've created is one you expect to use numerous times throughout the wizard, you might want to make it global.

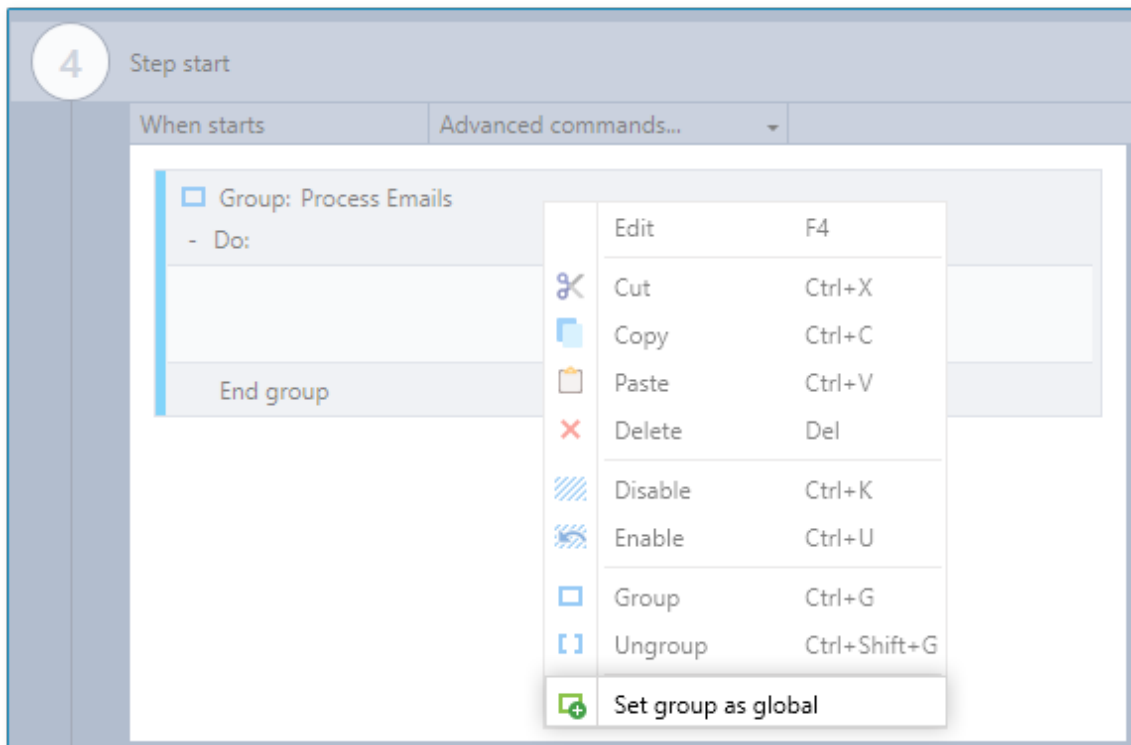


### NOTE

A global group is available in a **single wizard**. It is not shared among different wizards.

#### How do you do it?

Just right-click on the group and select the **SET GROUP AS GLOBAL** option.



### How does it help?

- Setting a group as global makes it available at the very top of the Toolbox Pane. Use it just as if it were one command by dragging it into the Editor Pane.
- When you edit a global group, the changes are automatically made everywhere the group has been used throughout the wizard.

# CHAPTER 3: Wizard Commands

In this chapter:

Continue Wizard .....	71
End Wizard .....	72
Go To Step .....	73
Get Step Data .....	74
Get Wizard Data .....	75
Check Leo Application .....	76
Get Leo User Data .....	77
Check Run Mode .....	78
Check Video Recording Mode .....	79
Set User Interrupt Mode .....	80
Show Message .....	81
Raise Wizard Error .....	83
Get Last Failure Type .....	84
Resume Error .....	85
Report Wizard Output .....	86

## Continue Wizard

Skip all remaining Advanced Commands within the same section of the wizard and move to the next section. It is generally used in combination with [Flow Commands](#) such as **IF ELSE**.

If the **CONTINUE WIZARD** command appears in:

- Step Start → the wizard will skip to the Core Action of the same step
- Step End → the wizard will skip to Step Start of the next step

The **CONTINUE WIZARD** command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.

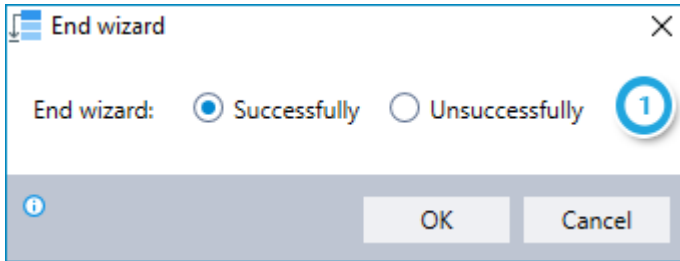
## End Wizard

End the wizard immediately. The wizard will skip:

- All remaining Advanced Commands and the Core Action of the current step; and
- All remaining steps of the wizard

It is generally used in combination with [Flow Commands](#) such as **IF ELSE**.

### Using the END WIZARD command



- 1** Choose whether to end the wizard successfully or unsuccessfully



#### NOTE

##### Successfully or Unsuccessfully: What difference does it make?

Whether a wizard ends successfully or unsuccessfully is significant in two primary respects:

- **Global Actions: WIZARD END** actions can be defined differently based on whether the wizard ends successfully or unsuccessfully; and
- **Leo Report Generator:** Leo differentiates between successful/unsuccessful wizard end for reporting purposes, allowing you to more accurately analyze the usage and effectiveness of your wizards



## Go To Step

Move immediately to the step you specify. The wizard will skip:

- All remaining Advanced Commands and the Core Action of the current step; and
- All steps between the current step and the step you specify in the command (the "**destination step**")

It is generally used in combination with [Flow Commands](#) such as **IF ELSE**.

### Using the GO TO STEP command

- 1 Choose whether to identify the **destination step** by step number/name or by the value stored in a variable
- 2 Instruct Leo how to handle any errors encountered. Read more about [Error Handling](#).



### CAUTION

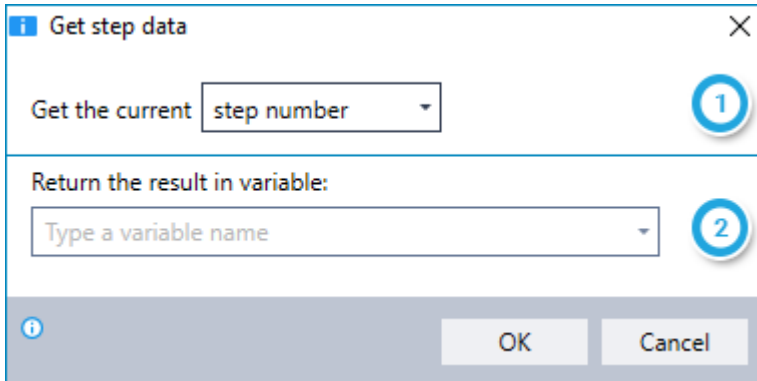
When the **destination step** is identified by variable, the specified variable must contain a numeric value that matches one of the wizard's step numbers.

If the variable specified contains an invalid value, the wizard will proceed sequentially – as if the **GO TO STEP** command were not there.

## Get Step Data

Retrieve the number or name of the current step and place the result into a new or existing variable.

### Using the GET STEP DATA command

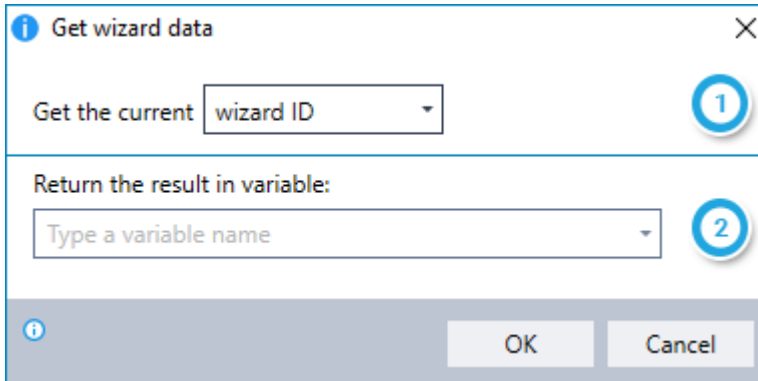


- 1 Choose whether to retrieve the number or name of the current step
- 2 Enter the name of the variable into which you'd like to place the result

## Get Wizard Data

Retrieve the number or name of the current step and place the result into a new or existing variable.

### Using the GET WIZARD DATA command



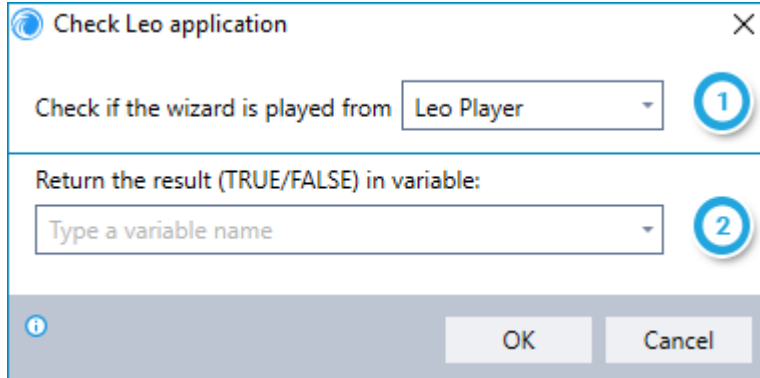
The screenshot shows a dialog box titled "Get wizard data". It has a close button (X) in the top right corner. The first section is labeled "Get the current" and has a dropdown menu with "wizard ID" selected. A circled "1" is next to this dropdown. The second section is labeled "Return the result in variable:" and has a text input field with "Type a variable name" entered. A circled "2" is next to this input field. At the bottom, there are "OK" and "Cancel" buttons.

- 1 Choose whether to retrieve the ID or name of the current wizard
- 2 Enter the name of the variable into which you'd like to place the result

## Check Leo Application

Check whether the current wizard is being played from Leo Player or Leo Studio.

### Using the CHECK LEO APPLICATION command

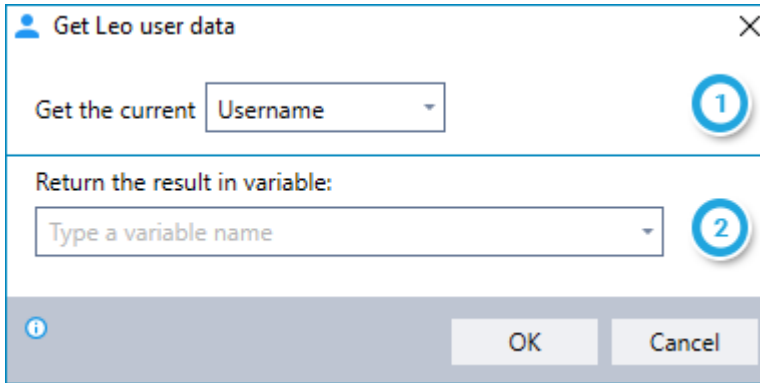


- 1 Choose the Leo application you'd like to check for (Leo Player or Leo Studio)
- 2 Enter the name of the variable into which you'd like to place the result. (The result will be either TRUE or FALSE, as applicable.)

## Get Leo User Data

Retrieve the Username or User ID of the user running the current wizard and place the result into a new or existing variable.

### Using the GET LEO USER DATA command

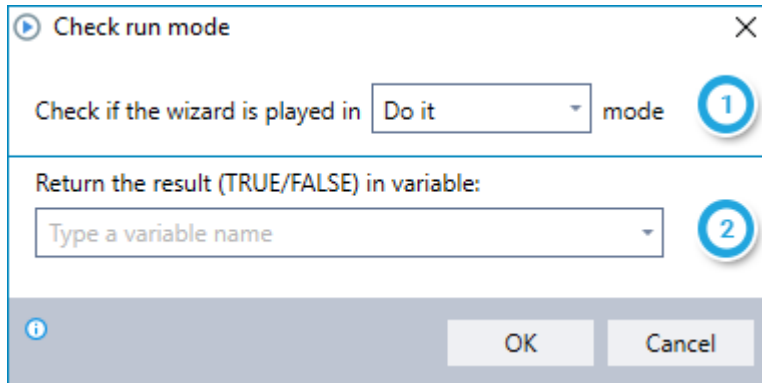


- 1 Choose whether to retrieve the Username or User ID of the user running the wizard
- 2 Enter the name of the variable into which you'd like to place the result

## Check Run Mode

Check whether the current wizard is being played in **DO IT** or **GUIDE ME** mode.

### Using the CHECK RUN MODE command

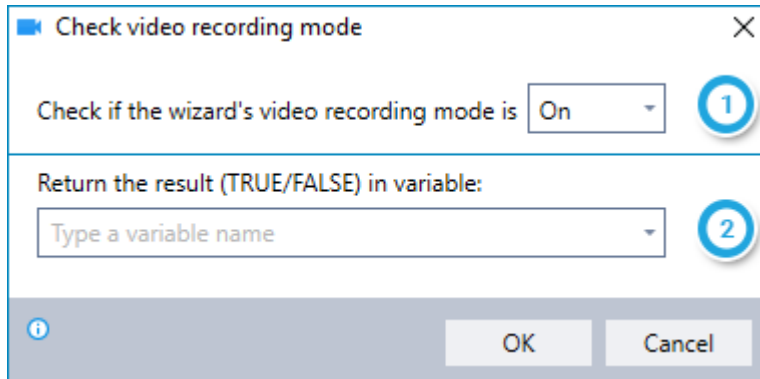


- 1 Choose the mode you'd like to check for (**DO IT** or **GUIDE ME**)
- 2 Enter the name of the variable into which you'd like to place the result. (The result will be either TRUE or FALSE, as applicable.)

## Check Video Recording Mode

Check whether the current wizard is being recorded with Leo's built-in recording feature.

### Using the CHECK VIDEO RECORDING MODE command



- 1 Choose which status of video recording mode you'd like to check for: ON or OFF
- 2 Enter the name of the variable into which you'd like to place the result. (The result will be either TRUE or FALSE, as applicable.)

## Set User Interrupt Mode

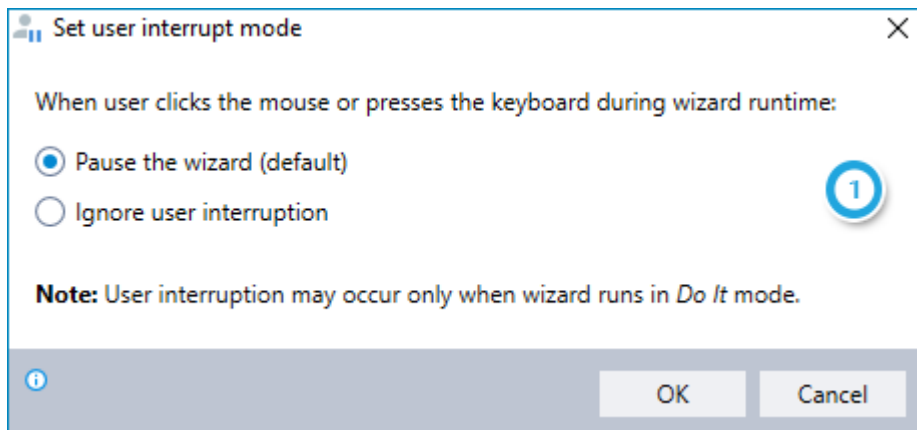
Select whether or not to pause a wizard when the end user clicks the mouse or uses the keyboard while it's running.



### NOTES

- This command is applicable only to a wizard running in **DO IT** mode
- By default, a wizard is paused when the end user clicks the mouse or uses the keyboard while it's running

### Using the SET USER INTERRUPT MODE command



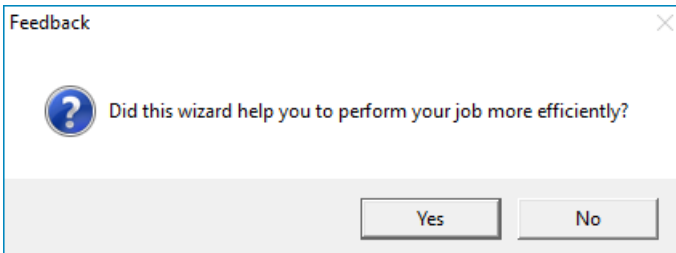
Choose what Leo will do if the user clicks the mouse or keyboard while the wizard is running, either:

- Pause the wizard (allow user interruption); or
- Continue playing the wizard (ignore user interruption)

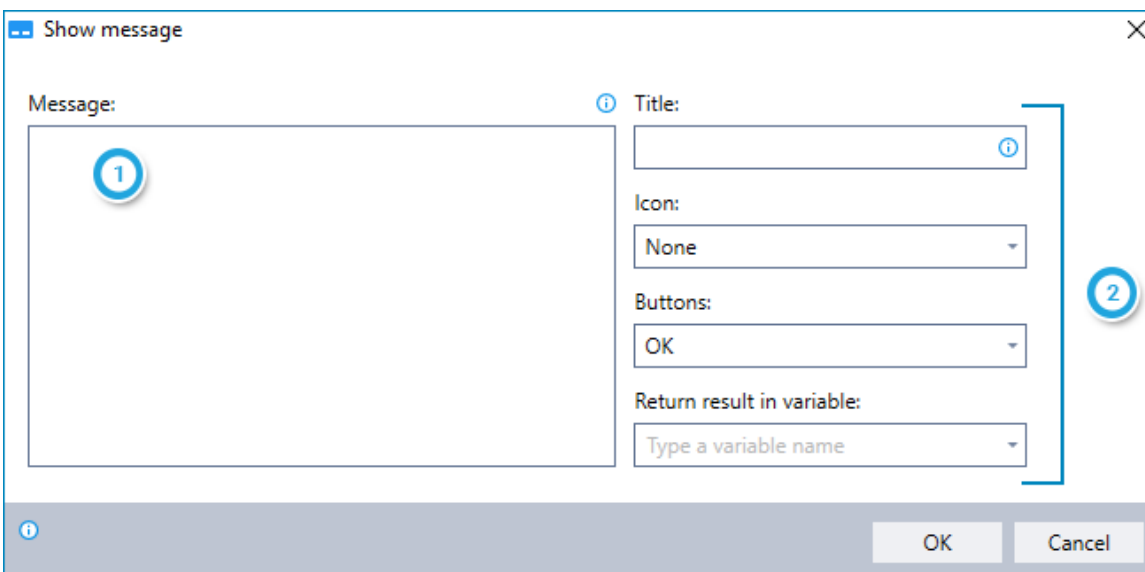


## Show Message

Display a customized message to the end user (or request basic information from the user) while a wizard is running. Here's a sample:



## Using the SHOW MESSAGE command



**1 Required:** Enter the text of the message you'd like to display

**2 Optional:**

- **Title** – Enter the text to appear in the title bar of the message
- **Icon** – Choose the icon to appear in the message (from the following options):
  - None (*default*)
  - Error
  - Question
  - Warning
  - Information

- **Buttons** – Choose the buttons available to the user for responding to the message (from the following available options):
  - OK (*default*)
  - OK / Cancel
  - Yes / No / Cancel
  - Yes / No
- **Result** – Enter the name of the variable into which you'd like to place the result (i.e., the button selected by the user in response to the message)



### TIP

#### When should I place the **SHOW MESSAGE** result into a variable?

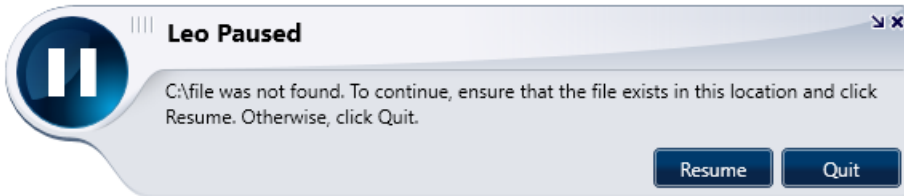
While placing the result of **SHOW MESSAGE** into a variable is optional, there are times when it can be particularly useful:

- When the flow of the wizard is based on the user's response (for example, when the response is used in combination with the **IF ELSE** command)
- When you want to log the user's response to review and analyze later

However, when the message is used solely to provide the user with information, it is generally not necessary to place the result into a variable.

## Raise Wizard Error

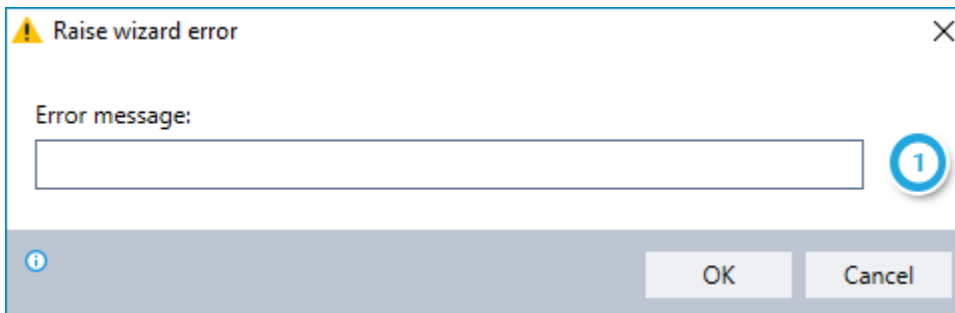
- Pause the currently running wizard;
- Display a short customized message to the end user; *and*
- Give the user the option to resume or quit the wizard



If the user:

- **Resumes** the wizard successfully → the wizard proceeds to the next step
- **Quits** the wizard → the wizard ends and is considered failed for reporting and notification purposes

## Using the RAISE WIZARD ERROR command



- 1 Enter the text of the message you'd like the end user to see.

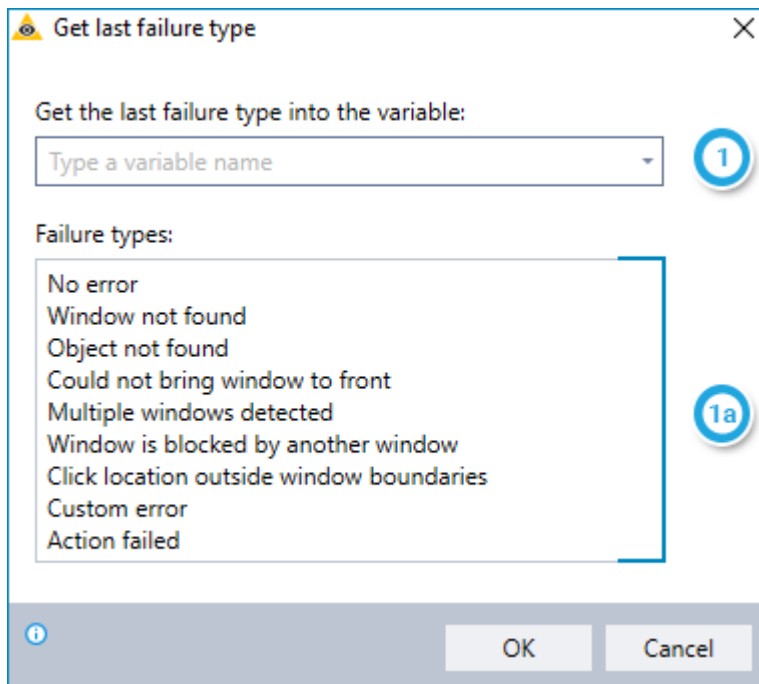
**Recommended:** The message should instruct the user how to correct the error before he clicks **RESUME**.

## Get Last Failure Type

Designed to work in conjunction with Leo's global error handling features (i.e., used as part of a globally-defined fallback procedure for when an error occurs), this command allows you to retrieve the type of the last failure handled and place it into a new or existing variable.

- For more information on Fallbacks, see the *Fallbacks Tab* section of the Leo Studio User Guide.
- For more information about global error handling, see the *Global Actions* section of the Leo Studio User Guide.

### Using the GET LAST FAILURE TYPE command



**1** Enter the name of the variable into which you'd like to place the last failure type.

Section **1a** of the **GET LAST FAILURE** type dialog lists the possible types of failures. (No need for you to do anything with this section... it's there just for your information.)

## Resume Error

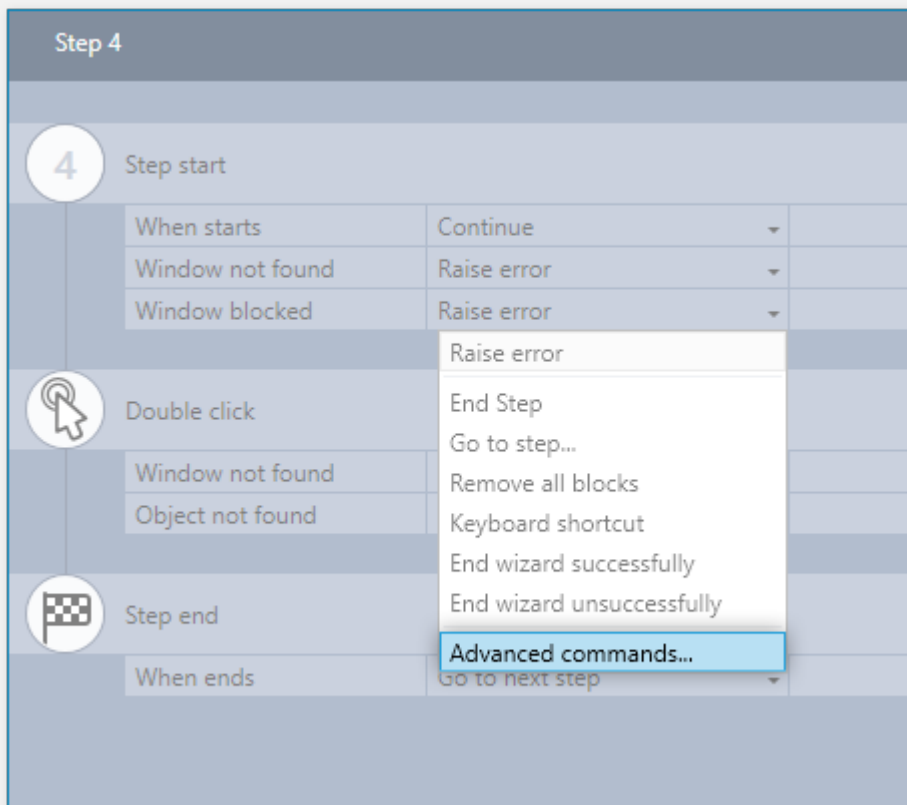
Designed to work in conjunction with Leo's fallback features (i.e., an action or series of actions you have defined for when an error occurs), this command instructs Leo to return to the default procedures for handling an error after performing the fallback procedures you have defined. For more information on Fallbacks, see the *Fallbacks Tab* section of the Leo Studio User Guide.

The **RESUME ERROR** command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.



### NOTE

The **RESUME ERROR** command is applicable only when using the **ADVANCED COMMANDS** option for defining fallback procedures (either for a single step or globally).



## Report Wizard Output

Hybrid Mode Feature

API Feature

Output the wizard's result, enabling it:

- to be placed in a variable in the wizard that initiated it; or
- returned in an API call



### NOTE

This command is relevant only to a wizard:

- Initiated by another wizard, using the **ADD AUTOMATION TASK TO QUEUE** command; or
- Invoked by an API call
  - For additional details, see the document: *Leo RPA - Web Service API* (Get Status: LeoWebResponse Parameters)

## Using the REPORT WIZARD OUTPUT command

**1** Choose whether to report the wizard's output as a variable value or as free text

- For **VARIABLE VALUE**, enter the name of the existing variable that contains the result to report
- For **FREE TEXT**, enter the text of the result to report. To incorporate variable value(s) in the text, type variable names between dollar signs (e.g., \$MyVar\$).

# CHAPTER 4: Mouse and Keyboard Commands

In this chapter:

Use Keyboard Shortcut .....	88
Input Text .....	89
Get Mouse Position .....	90
Drag & Drop .....	92
Mouse Click .....	93
Wait for Busy Cursor .....	94
Set Caps Lock State .....	95
Get Caps Lock State .....	96

## Use Keyboard Shortcut

Enter predefined keystrokes into the active application. It is especially useful when you need to enter keystroke combinations utilizing <CTRL>, <ALT>, <TAB>, etc.

### Using the USE KEYBOARD SHORTCUT command

- 1 Type the exact keystrokes you want the wizard to send to the active application. This can include single keystrokes, combinations, and/or a series of multiple keystrokes and combinations;

Choose whether the entered keystrokes must be in the language in which the wizard was recorded or if they can be in any language; *and*

Select the speed at which the keystrokes will be sent. The longer the interval selected, the slower they are sent. (Applications/configurations can vary widely in how quickly they are able to accept data, so finding the best interval may require a bit of experimentation.)

- 2 Choose whether or not you want the sent keystrokes to appear on the end user's screen while the wizard is running.



## Input Text

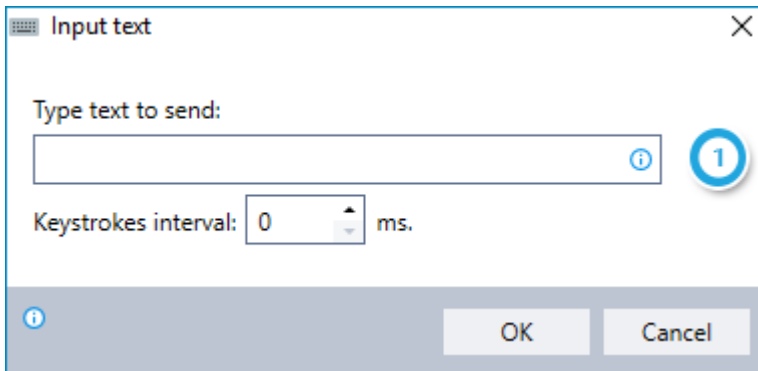
Send the text you've entered (or the text stored in a variable) to the active application.



### NOTE

You must activate the field into which the text will be entered before using this command.

## Using the INPUT TEXT command



Type the text you want Leo to send. (This can include free text and/or values copied from different variables.); **and**

Select the speed at which the keystrokes will be sent. The longer the interval selected, the slower they are sent. (Applications/configurations can vary widely in how quickly they are able to accept data, so finding the best interval may require a bit of experimentation.)

## Get Mouse Position

Retrieve the position of the mouse and store the **x** and **y** coordinates in variables.

It is generally used in conjunction with the **DRAG & DROP** and **MOUSE CLICK** commands.

### Using the GET MOUSE POSITION command

- 1 Enter the name of the variable into which you'd like to place the **x** coordinate of the mouse location; **and**  
Enter the name of the variable into which you'd like to place the **y** coordinate of the mouse location

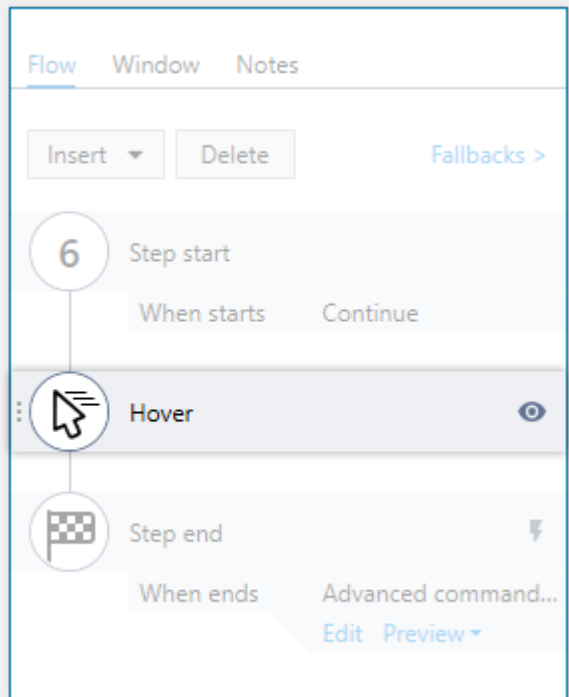


#### TIP

##### Getting the mouse where it needs to be

Before using the **GET MOUSE POSITION** command, you must instruct Leo to place the mouse in the location whose coordinates you want to retrieve. Here's how:

1. When recording a wizard, click the mouse in the relevant location.
2. Change the core action for that step from **CLICK** to **HOVER**.



3. Now, in **STEP END**, use the **GET MOUSE POSITION** Advanced Command to grab the coordinates for the location of the mouse.

## Drag & Drop

Pick up an object at a source location (either the current mouse position or a specific screen location identified by **x** and **y** coordinates) and drag it to a target location.

This command is especially useful when you need to move data from one column to another in the active application.

### Using the DRAG & DROP command

- 1 Identify the source location either as the current mouse position or a location identified by **x** and **y** coordinates stored in variables; *and* Enter the names of the variables in which the **x** and **y** coordinates of the target location are stored

(For more information about how to retrieve these coordinates, see [GET MOUSE POSITION](#).)

- 2 Specify additional details about the desired drag & drop operation:
  - Optional delay before drag and/or drop
  - Whether or not the mouse movement will appear on the end user's screen while the wizard is running
  - Combination with the <CTRL> or <SHIFT> keys

## Mouse Click

Send various types of mouse clicks to the active application either at the current mouse position or at a specific screen location identified by **x** and **y** coordinates.

### Using the **MOUSE CLICK** command

1 Choose whether to perform the mouse click at the current mouse position or at a location identified by **x** and **y** coordinates stored in variables. (For more information about how to retrieve these coordinates, see [GET MOUSE POSITION](#).)

2 Select precisely the type of mouse click you'd like Leo to perform:

- Left, right, or middle mouse button
- Single, double, or triple click
- Mouse click in combination with the <CTRL> or <SHIFT> keys

Additionally, the **MOUSE MOVEMENT** field allows you to choose whether or not the mouse movement will appear on the end user's screen while the wizard is running.

## Wait for Busy Cursor

Inevitably, complex computer systems sometimes move slower than we would like, and we see that dreaded spinning cursor. The **WAIT FOR BUSY CURSOR** command instructs Leo to wait for the cursor to stop spinning before performing the next step of the wizard.

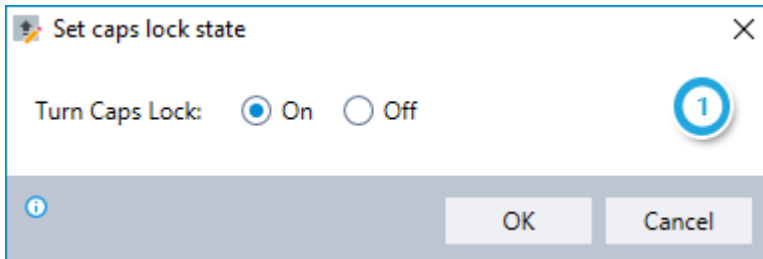
The **WAIT FOR BUSY CURSOR** command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.

## Set Caps Lock State

Set **Caps Lock** to **On** or **Off**, as you specify.

It is often used after checking the current state with the **GET CAPS LOCK STATE** command.

### Using the SET CAPS LOCK STATE command



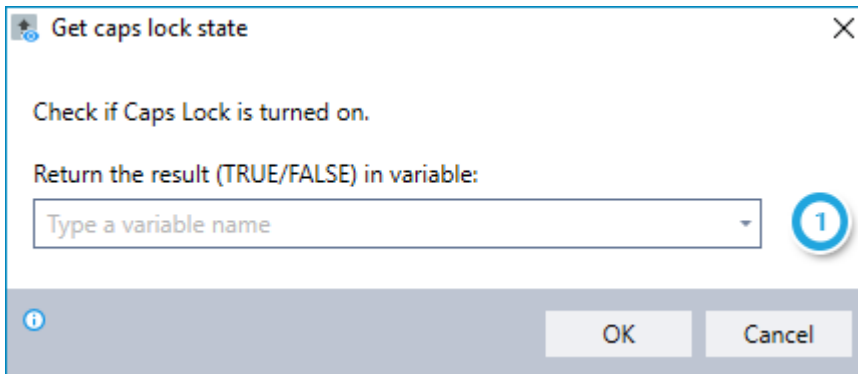
- 1 Choose whether to set **Caps Lock** to **On** or **Off**.

## Get Caps Lock State

Check whether **Caps Lock** is set to **On**.

It is often used in conjunction with the **SET CAPS LOCK STATE** command, which allows you to set **Caps Lock** to **On** or **Off** as required.

### Using the GET CAPS LOCK STATE command



- 1 Enter the name of the variable into which you'd like to place the result of the check:
- The result will be TRUE if **Caps Lock** is **On**
  - The result will be FALSE if **Caps Lock** is **Off**



# CHAPTER 5: Block Commands

Various types of blocks can be added to Leo wizards to prevent end user actions on the target application while the wizard is running. For a more detailed explanation of different types of blocks and how to use them, see the *Blocks* section of the Leo Studio User Guide.

If the end user tries to execute an action that has been blocked (such as a mouse click on a particular button), Leo stores the blocked action, and the wizard proceeds according to the logic you have defined. Then, you can use the following Advanced Commands to either allow or deny the stored blocked action, complete the logical flow, and achieve the desired result:

**ALLOW LAST STORED ACTION**

**DENY LAST STORED ACTION**

**REMOVE ALL BLOCKS**



## NOTE

A block is not specific to a certain step of the wizard. Unless removed, it remains in effect throughout the current run of the wizard.

## Allow last stored action

Instruct Leo to send the last blocked action to the active application.

This command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.

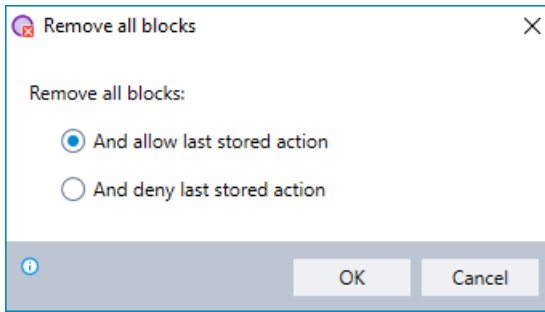
## Deny last stored action

Instruct Leo **NOT** to send the last blocked action to the active application.

This command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.

## Remove all blocks

Instruct Leo to cancel all blocks currently in effect, with the option either to allow or deny the last stored action:



# CHAPTER 6: Date and Time Commands

Leo's **DATE AND TIME COMMANDS** allow you to perform complex calculations and comparisons with dates and times. The results of these calculations may be an end in themselves. But more often, they are utilized to direct the logical flow of the remainder of the wizard.

In this chapter:

Get Current Date .....	101
Validate Date .....	102
Get Day of Month .....	103
Compare Dates .....	105
Add/Subtract Date .....	107
Calculate Date Range .....	108
Check Day of Week .....	110
Format Date .....	112
Get Current Time .....	114
Compare Time .....	115
Add/Subtract Time .....	117
Calculate Time Range .....	119
Convert Between Time Zones .....	122



## A NOTE ABOUT FORMATTING

Use the following formatting to ensure that your **DATE AND TIME COMMANDS** are executed properly:

### Date

- **Day:** Use 1 or 2 digits (i.e., the first of the month could be entered as either as 1 or 01)
- **Month:** Use 1 or 2 digits (i.e., August could be entered as either as 8 or 08)

- **Year:** Use 4 digits (e.g., 1948)

#### **Time**

- **Hour:** Use 1 or 2 digits in 24-hour time notation (i.e., two o'clock in the morning could be entered as either as 2 or 02; two o'clock in the afternoon must be entered as 14)
- **Minutes:** Use 1 or 2 digits (i.e., three minutes after the hour can be entered as either as 3 or 03)
- **Seconds:** Use 1 or 2 digits (i.e., six seconds after the minute can be entered as either as 6 or 06)

## Get Current Date

Retrieve the current day, month, or year (according to the system clock of the machine on which Leo Player is running) and place the result into a new or existing variable.

### Using the GET CURRENT DATE command

- 1 Choose which date parameter to retrieve: day, month, or year
- 2 Enter the name of the variable into which you'd like to place the result



#### NOTE

Each date parameter must be stored in its own variable. In order to retrieve the full date, use the **GET CURRENT DATE** command 3 times – each time choosing a different parameter.

If you wish, you can then combine these individual variables into a single string for the full date using the **SET VALUE** command.

## Validate Date

Check whether the date parameters (day, month, and year) stored in 3 individual variables together constitute a valid date.

To ensure Leo can read your date parameters, see [A NOTE ABOUT FORMATTING](#).

### Using the VALIDATE DATE command

- 1 Enter the names of the variables in which each of the individual date parameters is stored
- 2 Enter the name of the variable into which you'd like to place the result of the check:
  - The result will TRUE if the date is valid
  - The result will be FALSE if the date is not valid
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get Day of Month

Retrieve the first or last day of any month. This command is most-often used in order to determine the first or last **working** day of a month.



### NOTE

Prior to using this command, specify the relevant month and year by placing these values into variables. To ensure that the formatting of these variables is correct, see [A NOTE ABOUT FORMATTING](#).

### Using the GET DAY OF MONTH command

- 1 Choose whether you'd like to retrieve the first or last day of the specified month; **and** Enter the names of the variables in which the relevant month and year are stored
- 2 Indicate whether you'd like Leo to consider only working days when calculating the first or last day of the month
  - If you have chosen to consider only working days, provide the days that constitute a work week

- 3 Enter the name of the variable into which you'd like to place the result
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



### EXAMPLE

Let's say you have a series of reports that must be run as of the last working day of each month. The **GET DAY OF MONTH** command can determine the exact date on which the reports must be run.



## Compare Dates

Compare 2 dates (each stored in variables) and determine whether one is before, after, or equal to the other.

Each date parameter (day, month, year) must be stored in its own variable. Therefore, when using this command, you are working with a total of 6 variables – 3 variables for each of the 2 dates being compared.

To learn more about the correct formats for these variables, see [A NOTE ABOUT FORMATTING](#).

### Using the COMPARE DATES command

The screenshot shows the 'Compare dates' dialog box. It has a title bar with a close button. The main area is divided into several sections:

- Check if:** This section contains two columns of date parameters. Each column has three dropdown menus for 'Day', 'Month', and 'Year'. A comparison operator dropdown (currently set to 'is before') is positioned between the two columns. Callout 1 points to the first 'Day' dropdown, and callout 2 points to the second 'Day' dropdown.
- Return the result (TRUE/FALSE) in variable:** This section has a single dropdown menu for entering a variable name. Callout 4 points to this dropdown.
- Error handling:** This section has a dropdown menu for selecting an error handling option. Callout 5 points to this dropdown.

At the bottom of the dialog, there are 'OK' and 'Cancel' buttons, and an information icon on the left.

- 1 For the first of the 2 dates to be compared: Enter the names of the variables in which each of the individual date parameters is stored
- 2 For the second of the 2 dates to be compared: Enter the names of the variables in which each of the individual date parameters is stored
- 3 Select whether you'd like to check if the date in column 1 is before, after, or equal to the date in column 2.
- 4 Enter the name of the variable into which you'd like to place the result. (The result will be either TRUE or FALSE, as applicable.)
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



### EXAMPLE

Let's say your insurance company needs to notify all customers whose policies have elapsed. You can use the **COMPARE DATES** command to compare whether the current date is after the expiration date of the policy and send notices only to those customers for whom this comparison is TRUE.

## Add/Subtract Date

Calculate a date by adding or subtracting days, months, and/or years to an existing date. **The variables containing the existing date parameters will be overwritten by the result of the calculation.**



### NOTE

- The existing date must be stored in variables.
- The days, months, and/or years to be added to or subtracted from the existing date can be entered manually or copied from values stored in variables.

To learn more about properly formatting these variables, see [A NOTE ABOUT FORMATTING](#).

## Using the ADD/SUBTRACT DATE command

- 1 For the existing date: Enter the names of the variables in which each of the individual date parameters is stored
- 2 Enter the number of days/months/years you would like to add to or subtract from the existing date.
  - To **subtract** days/months/years, enter the values as negative. (If you are subtracting using variables, the values stored in the variables must be negative.)
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Calculate Date Range

Determine the number of days between 2 specified dates.

Each date parameter (day, month, year) must be stored in its own variable. Therefore, when using this command, you are working with a total of 6 variables – 3 variables for each of the 2 dates in the range.

To learn more about the correct formats for these variables, see [A NOTE ABOUT FORMATTING](#).

### Using the CALCULATE DATE RANGE command

**1** Enter the names of the variables in which each of the individual date parameters is stored for both the **FROM DATE** and **TO DATE**

- To prevent errors when the wizard is run, ensure that the values of the variables are within the listed ranges
  - **Tip:** It's easy to check variable values as they would stand at any point during execution of the wizard by using the [VIEW VARIABLE LIST](#) command

- 2 Indicate whether you'd like Leo to count only working days within the range
  - If you have chosen to count only working days, provide the days that constitute a work week
- 3 Enter the name of the variable into which you'd like to place the result
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



### EXAMPLE

Let's say your insurance company offers a grace period to customers whose policies have lapsed due to non-payment, giving them the option to reinstate their policies if payment is made within 7 working days . You can use the **CALCULATE DATE RANGE** command to check whether payment was received during the 7-day grace period.

## Check Day of Week

Check the day of the week on which a specified date falls.



### NOTE

Prior to using this command, place the parameters of the relevant date (day, month, and year) into variables. To ensure that the formatting of these date parameters is correct, see [A NOTE ABOUT FORMATTING](#).

### Using the CHECK DAY OF WEEK command

- 1 Enter the names of the variables in which each of the individual date parameters is stored
- 2 Indicate the day(s) of the week for which you'd like to check. (For example, if you'd like to determine if the specified date falls on a weekend, check the boxes for Saturday and Sunday.)
- 3 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## EXAMPLE

Let's say you have a series of reports that must be run daily **except on weekends**. Before running the reports, use the **CHECK DAY OF WEEK** command to ensure that the report date is not a Saturday or Sunday.

## Format Date

Convert any date to:

- Long date format (e.g., Monday, June 5, 2017)
- Short date format (e.g., 6/5/2017)
- A custom-defined date format (for details, see [Specifying custom date formats](#))

### Using the FORMAT DATE command

- 1 Enter the date to be formatted

  - Can be free text and/or values copied from different variables
  - Month should precede day in a numeric date string (i.e., 6/5/2017 is interpreted as June 5, 2017)
  - See [A NOTE ABOUT FORMATTING](#) for additional information about correctly entering this date
- 2 Choose the format to which you'd like the specified date to be converted
- 3 Enter the name of the variable into which you'd like to place the result



## Specifying custom date formats

To display	Use this code <b>**case-sensitive**</b>
Months as 1-12	M
Months as 01-12	MM
Months as Jan-Dec	MMM
Months as January-December	MMMM
Days as 1-31	d
Days as 01-31	dd
Days as Sun-Sat	ddd
Days as Sunday-Saturday	dddd
Years as 00-99	YY
Years as 1900-9999	YYYY



### EXAMPLES

dddd d-MMM-yyyy → Monday 5-Jun-2017

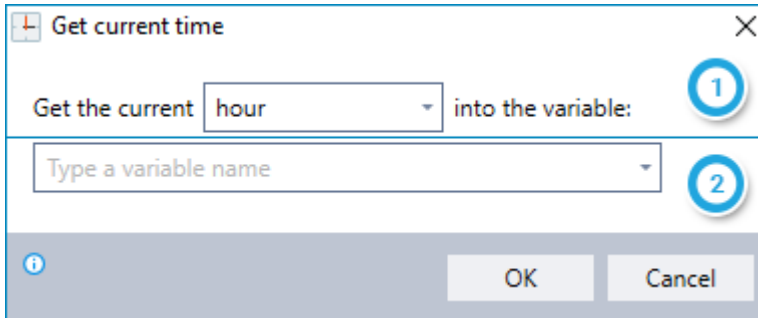
MMMM dd, yyyy → June 05, 2017

MM/dd/yy (dddd) → 06/05/17 (Monday)

## Get Current Time

Retrieve the current hour, minutes, or seconds (according to the system clock of the machine on which Leo Player is running) and place the result into a new or existing variable.

### Using the GET CURRENT TIME command



- 1 Choose which time parameter to retrieve: hour, minutes, or seconds
- 2 Enter the name of the variable into which you'd like to place the result



#### NOTE

Each date parameter must be stored in its own variable. In order to retrieve the full date, use the **GET CURRENT TIME** command 3 times – each time choosing a different parameter.

If you wish, you can then combine these individual variables into a single string for the full time using the **SET VALUE** command.

## Compare Time

Compare 2 times (each stored in variables) and determine whether one is before, after, or equal to the other.

Each time parameter (hour, minutes, seconds) must be stored in its own variable. Therefore, when using this command, you are working with a total of 6 variables – 3 variables for each of the 2 times being compared.

To learn more about the correct formats for these variables, see [A NOTE ABOUT FORMATTING](#).

### Using the COMPARE TIME command

- 1 For the first of the 2 times to be compared: Enter the names of the variables in which each of the individual time parameters is stored
- 2 For the second of the 2 times to be compared: Enter the names of the variables in which each of the individual time parameters is stored
- 3 Select whether you'd like to check if the time in column 1 is before, after, or equal to the time in column 2.
- 4 Enter the name of the variable into which you'd like to place the result. (The result will be either TRUE or FALSE, as applicable.)
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## EXAMPLE

Your HR department needs to provide management with a report of all employees who clocked in late for their scheduled shift. You can use the **COMPARE TIME** command to compare whether the *shift start* time is before the *clock-in* time for each employee and use this data to prepare a report listing all employees for whom this comparison is TRUE.

## Add/Subtract Time

Calculate a time by adding or subtracting hours, minutes, and/or seconds to an existing time. **The variables containing the existing time parameters will be overwritten by the result of the calculation.**



### NOTE

- The existing time must be stored in variables.
- The hours, minutes, and/or seconds to be added to or subtracted from the existing time can be entered manually or copied from values stored in variables.

To learn more about properly formatting these variables, see [A NOTE ABOUT FORMATTING](#).

## Using the ADD/SUBTRACT TIME command

- 1 For the existing time: Enter the names of the variables in which each of the individual date parameters is stored
- 2 Enter the number of hours/minutes/seconds you would like to add to or subtract from the existing time.
  - To subtract hours/minutes/seconds, enter the values as negative. (To subtract using variables, the values stored in the variables must be negative.)
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Calculate Time Range

Determine the number of hours, minutes, or seconds between 2 specified times.

Each time parameter (hour, minutes, seconds) must be stored in its own variable. Therefore, when using this command, you are working with a total of 6 variables – 3 variables for each of the 2 times in the range.

To learn more about the correct formats for these variables, see [A NOTE ABOUT FORMATTING](#).

### Using the CALCULATE TIME RANGE command

- 1 Choose whether you'd like the result of the calculation to be presented in hours, minutes, or seconds; *and*
  - Enter the names of the variables in which each of the individual time parameters is stored for both the **FROM TIME** and **TO TIME**
    - To prevent errors when the wizard is run, ensure that the values of the variables are within the listed ranges
      - **Tip:** It's easy to check variable values as they would stand at any point during execution of the wizard by using the [VIEW VARIABLE LIST](#) command

- 2 Enter the name of the variable into which you'd like to place the result



Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).





## EXAMPLE

Your HR department needs to provide management with a report of all employees who clocked in late for their scheduled shift, including the amount of time by which they were late. A combination of two **DATE AND TIME COMMANDS** can help you get this done in a snap:

1. Use the **COMPARE TIME** command to compare whether the *shift start time* is before the *clock-in time* for each employee
2. For the employees for whom this comparison is TRUE, use the **CALCULATE TIME RANGE** command to determine the amount of time by which they were late

## Convert Between Time Zones

Convert a date/time from an origin time zone to a destination time zone.

Each parameter (day, month, year, and time) can be manually entered or can be copied from values stored in variables.

To learn more about the correct formats for these parameters, see [A NOTE ABOUT FORMATTING](#). Note that for this command, there is only one time parameter (which should be formatted as hh:mm).

- 1 Enter the date and time at the origin time zone (can be entered manually or copied from values stored in variables)
- 2 Choose to specify the origin time zone by selecting from the list or by entering the value (either manually or copied from a value stored in a variable)
- 3 Choose to specify the destination time zone by selecting from the list or by entering the value (either manually or copied from a value stored in a variable)
- 4 Indicate whether daylight savings time should be considered in the conversion

- 5 Enter the names of the variables into which you'd like to place the result
- 6 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 7: Window Commands

In this chapter:

Get Step Window Handle .....	125
Find Matching Window Handles .....	126
Get Active Application .....	128
Control Window State .....	129
Check Window State .....	130
Get Active Window/Web Page .....	131
Capture Step Window Image .....	132

## Get Step Window Handle

Retrieve the handle of the window on which the wizard's current step is running and place the result into a new or existing variable. You can then use this information to control window-specific actions (such as maximizing the window, bringing it to the front, and closing it).



### NOTE

#### Wait, wait, wait... what is a handle?

If you asked a developer this question, he or she would probably give you a definition including concepts such as *abstraction*, *pointer*, *API*, and *physical memory*. But for our purposes, a handle is a unique numeric identifier for each window currently running on a Windows desktop.

## Using the GET STEP WINDOW HANDLE command

- 1 Enter the name of the variable into which you'd like to place the handle
- 2 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Find Matching Window Handles

Retrieve the handles of all running windows that match the current step's window detection criteria and place them into a new or existing variable.

- To learn more about window handles, see [Wait, wait, wait... what is a handle?](#)
- To learn more about how Leo identifies the window on which the wizard action is performed, see the *Window Detection* section of the Leo Studio User Guide.

### Using the FIND MATCHING WINDOW HANDLES command

Find matching window handles

Find all windows that match the current step's window/page detection.

Return matching window handles in variable:

Type a variable name

Separator: ,

OK Cancel

- 1 Enter the name of the variable into which you'd like to place the matching window handles
- 2 Enter the separator you want to use to separate each matching window handle found



## EXAMPLE

### Finding all open Excel windows

Let's say you have recorded a wizard that writes data to an Excel spreadsheet, and to ensure it runs properly, you want identify all open instances of Excel on the end user's desktop.

The window detection properties for a wizard step running on an Excel window might look something like this:

The screenshot shows a configuration window with three tabs: 'Flow', 'Window', and 'Notes'. The 'Window' tab is selected. At the top, there are 'Insert' and 'Delete' buttons. Below them is a 'Window data' section with a window icon. It contains two checked items: 'Class name' with the operator 'Equals' and value 'XLMAIN', and 'Caption' with the operator 'Contains' and value 'Excel'. Below this is an 'Options' section with a checkbox for 'Wait for window to appear ( 1 sec.)', a dropdown for 'Bring window to front' set to 'Automatically', and a checkbox for 'Custom window name' with an information icon and an empty text field.

Use the **FIND MATCHING WINDOW HANDLES** command to retrieve the handles of all open windows with **CLASS NAME** equal to `XLMAIN` and **CAPTION** containing `Excel`.

You might then want to utilize the **CONTROL WINDOW STATE** command together with the retrieved handles in order to minimize or close all open Excel windows other than the one being used by the wizard.

## Get Active Application

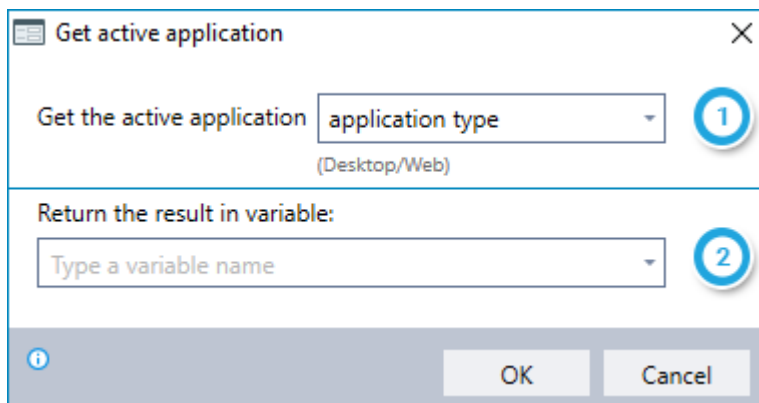
Retrieve information about the application on which the current wizard is being played and place it into a new or existing variable.

You can choose to obtain the following types of information:

- Application type (desktop/web)
- Main window caption
- Browser type (IE/Firefox/Chrome/none)
- Browser version
- Process name
- Process ID

This command can be especially useful when the logical flow of the wizard varies based on the specs of the application on which it is being run.

### Using the GET ACTIVE APPLICATION command



- 1 Choose the type of information you want to retrieve
- 2 Enter the name of the variable into which you'd like to place the result

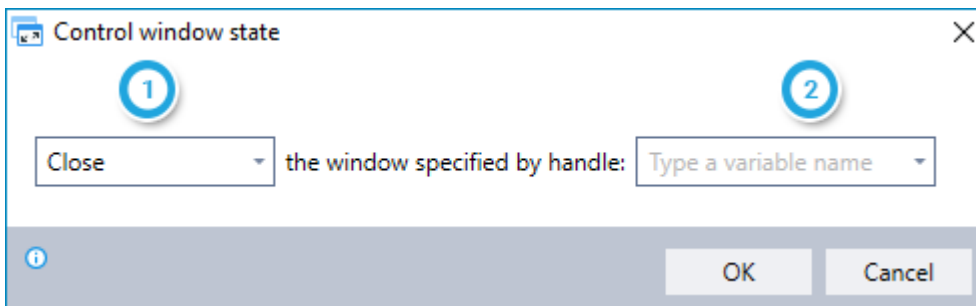


## Control Window State

Control the state of a window running on the desktop of the end user.

- The relevant window is identified by its handle (previously retrieved with the **GET STEP WINDOW HANDLE** or **FIND MATCHING WINDOW HANDLES** command)
- Available actions:
  - Close
  - Minimize
  - Maximize
  - Restore
  - Bring window to front

### Using the **CONTROL WINDOW STATE** command



- 1 Choose the action you would like to take on the relevant window
- 2 Enter the name of the variable in which the handle of the window is stored

## Check Window State

Check the state of a window running on the desktop of the end user. If necessary, you can then use the **CONTROL WINDOW STATE** command to change it to the required state.

- The relevant window is identified by its handle (previously retrieved with the **GET STEP WINDOW HANDLE** or **FIND MATCHING WINDOW HANDLES** command)
- Available window states for which to check:
  - Exists
  - Is visible
  - Is active
  - Is minimized
  - Is maximized

### Using the CHECK WINDOW STATE command

- 1 Enter the name of the variable in which the handle of the relevant window is stored; **and** Choose the window state for which you would like to check
- 2 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)

## Get Active Window/Web Page

Retrieve information about the currently active window or web page and place it into a new or existing variable.

You can choose to obtain the following types of information:

- Window caption or web page URL
- Window handle
- Window caption

### Using the Get ACTIVE WINDOW/WEB PAGE command

Get active window/web page

Get the active  into the variable:

OK Cancel

- 1 Choose the type of information you want to retrieve
- 2 Enter the name of the variable into which you'd like to place the result

## Capture Step Window Image

Take a screen capture of the window on which the wizard's current step is running and save it to a file. This command can be especially useful for troubleshooting and debugging wizards or for providing further training to end users.

### Using the CAPTURE STEP WINDOW IMAGE command

Capture step window image

Capture the current step's window image and save it to a file.

Target folder:

Image type:

Return image full path in variable:

Error handling

OK Cancel

- 1 Enter the folder in which the file should be saved
- 2 Enter the file format in which you'd like the image to be saved: JPG, PNG, or BMP
- 3 Enter the name of the variable into which you'd like to place the full path of the image
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 8: File Commands

In this chapter:

Create a Text File .....	134
Read From Text File .....	136
Write to Text File .....	137
Does File Exist .....	138
Copy a File .....	139
Move a File .....	140
Rename a File .....	141
Delete a File .....	142
Delete File(s) .....	143
Monitor File Changes .....	145

## Create a Text File

Create a new text file and place its file path in a new or existing variable. This command is often used to create a file that will be later written to using the **WRITE TO TEXT FILE** command.

### Using the CREATE A TEXT FILE command

1 Enter the name of the variable into which you'd like to place the file path of the new file

2 Indicate if you'd like the file to be created:

- In a folder other than the Windows Temp folder
- Using a custom file name
- Using a custom file extension
  - Enter the extension either in the format **.xyz** or simply **xyz** (for example, `.txt` or `txt`). Do not include an `*`.

If you elect not to specify these options, the new file will be created by default in the Windows Temp folder, with a unique file name, and the `.tmp` file extension.

3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.



### CAUTION

If a file with the same name already exists in the specified location, it will be overwritten by the new file.



### TIP

#### Don't forget to clean house!

Despite its name, Windows does not automatically delete files from the Windows Temp folder. From a safety perspective, this is great. But it does mean that your Windows Temp folder could become quite huge (and something of a resource hog) unless you manually clean it out from time to time.

## Read From Text File

Read the contents of a text file into a new or exiting variable.

### Using the READ FROM TEXT FILE command

- 1 Select the text file whose contents you want to read into a variable
- 2 Enter the name of the variable into which you'd like to place the text
- 3 Choose whether to read the full contents of the file into the variable or a single line at a time
  - For a single line at a time, enter the name of the variable into which you'd like to place the result TRUE when the end of the file is reached
    - You can then use this variable with the **LOOP** command so that Leo exits the loop when reaching the end of the file
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.



## Write to Text File

Write the contents of a variable into a new or existing text file.

### Using the WRITE TO TEXT FILE command

- 1 Enter the **full file path and file name** of the text file to which you would like to write
  - If the file does not exist, when the wizard is run, Leo will create one with the name you entered
- 2 Enter the name of the variable whose contents will be written to the file
- 3 Choose how to treat any existing text in the file:
  - Add the contents of the variable to the end of the existing text (**APPEND TEXT**); *or*
  - Replace the existing text with the contents of the variable (**OVERWRITE TEXT**)
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Does File Exist

Check to see if a file exists and place the result of the check (TRUE/FALSE) into a variable.

### Using the DOES FILE EXIST command

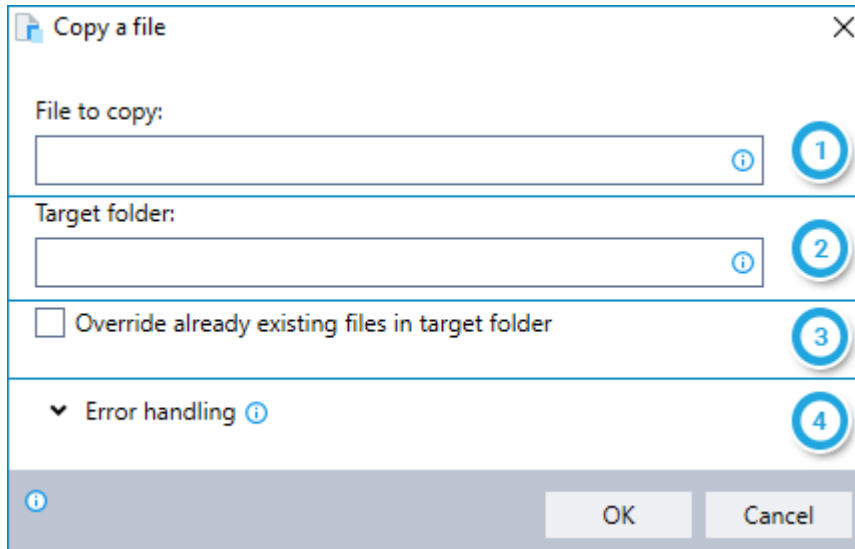
The screenshot shows a dialog box titled "Does file exist". It has a close button (X) in the top right corner. The main content area is divided into two sections. The first section is labeled "Check if the file" and contains a text input field followed by the word "exists.". A blue circular callout with the number "1" points to this input field. The second section is labeled "Return the result (TRUE/FALSE) in variable:" and contains a dropdown menu with the placeholder text "Type a variable name". A blue circular callout with the number "2" points to this dropdown menu. At the bottom of the dialog are "OK" and "Cancel" buttons. There is also an information icon (i) in the bottom left corner.

- 1 Enter the **full file path and file name** of the file for whose existence you would like to check
- 2 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)

## Copy a File

Copy a file from its current location to a location you choose.

### Using the COPY A FILE command

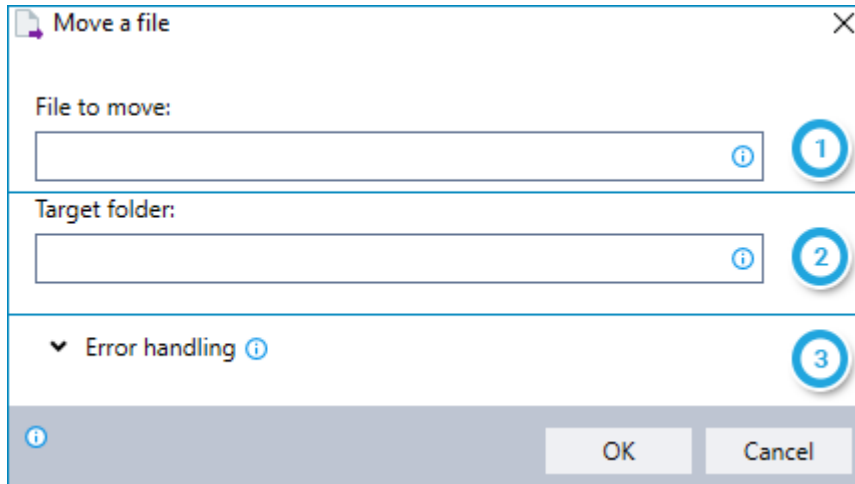


- 1 Enter the **full file path and file name** of the file you would like to copy
- 2 Enter the **full path** of the folder to which you would like to copy the file
- 3 Indicate whether or not to overwrite any existing file of the same name in this location
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Move a File

Move a file from its current location to a location you choose.

### Using the MOVE A FILE command



The screenshot shows a dialog box titled "Move a file" with a close button (X) in the top right corner. It contains three sections: "File to move:" with a text input field and an information icon (i) and a callout bubble with the number 1; "Target folder:" with a text input field, an information icon (i), and a callout bubble with the number 2; and "Error handling" with a dropdown arrow, an information icon (i), and a callout bubble with the number 3. At the bottom, there are "OK" and "Cancel" buttons, along with a small information icon (i) on the left.

- 1 Enter the **full file path and file name** of the file you would like to copy
- 2 Enter the **full path** of the folder to which you would like to move the file
  - If a file of the same name already exists in the target folder, the file will not be moved (and will not be deleted from its original location)
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Rename a File

Rename an existing file.

### Using the RENAME A FILE command

The screenshot shows a dialog box titled "Rename a file" with a close button (X) in the top right corner. It contains three main sections:

- File to rename:** A text input field with a blue circular callout '1' to its right.
- New name:** A text input field with a blue circular callout '2' to its right. Below the field is the text "Enter file name including extension. Example: Log.txt".
- Error handling:** A section with a dropdown arrow and the text "Error handling" followed by a blue circular callout '3'.

At the bottom of the dialog are "OK" and "Cancel" buttons, and a small information icon (i) on the left.

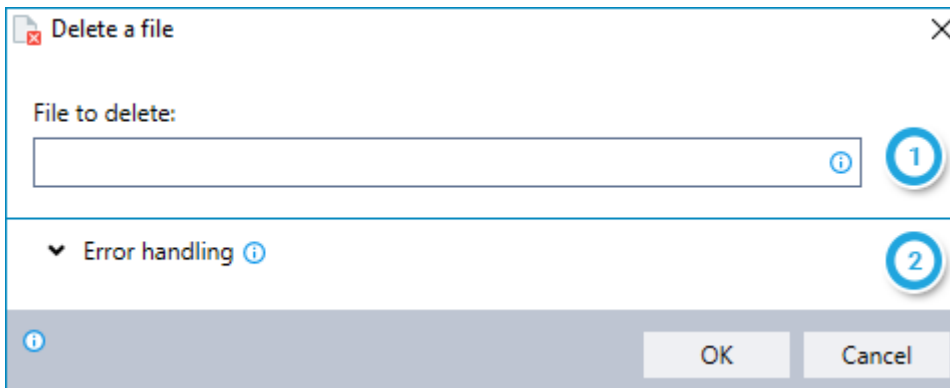
- 1 Enter the **full file path and file name** of the file you would like to rename
- 2 Enter the new name you would like to give the file, including the file extension
  - If you enter only the file name (with no path), the file will be renamed and remain in its current location
  - If you enter a new file path along with the file name, the file will be renamed **AND** moved to the new location you entered
    - If a file of the same name already exists in the new location, the file will **NOT be renamed or moved**
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Delete a File

Delete a single existing file.

If you want to delete more than one file at a time, take a look at the **DELETE FILE(S)** command.

### Using the DELETE A FILE command



- 1 Enter the **full file path and file name** of the file you would like to delete
- 2 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.



#### CAUTION

**Make sure you really want to delete it!**

The **DELETE A FILE COMMAND** does not put the deleted file into the Recycle Bin... it deletes it permanently.

## Delete File(s)

Delete one or more existing files from a specified folder, with the option to include subfolders. Deleting more than one file requires that the files are named according to a pattern that can be represented using asterisks (\*) as wildcards.

### Using the DELETE FILE(S) command

The screenshot shows the 'Delete file(s)' dialog box with the following fields and options:

- Specify a folder name to delete from:** A text input field with an info icon.
- Make sure the folder is accessible for all end users.** A checkbox labeled 'Include sub-folders'.
- File to delete:** A text input field with an info icon.
- Return the number of deleted files in variable (optional):** A dropdown menu with the placeholder text 'Type a variable name'.
- Return the number of files that failed to delete in variable (optional):** A dropdown menu with the placeholder text 'Type a variable name'.
- Error handling:** A dropdown menu with an info icon.

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

- 1 Enter the **full file path** of the folder from which you would like to delete files; *and* Indicate whether or not to also delete files from subfolders
- 2 **To delete a single file:** Enter the name of a the file to delete; *or*  
**To delete multiple files:** Enter a naming pattern of the files to delete, using asterisks (\*) as wildcards to represent one or more characters in the file name, for example:
  - The pattern `file*.txt` will delete `file.txt`, `file1.txt`, `file48.txt`, `file1948.txt`, etc.
  - The pattern `file.*` will delete `file.txt`, `file.docx`, `file.xlsx`, `file.png`, etc.

**3** (Optional) Specify variables in which to place the results of the delete operation:

- The number of files that were deleted
- The number of files matching the specified pattern that failed to delete

**4** Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



### CAUTION

**Make sure you really want to delete them!**

The **DELETE FILE(S) COMMAND** does not put deleted files into the Recycle Bin... it deletes them permanently.



## Monitor File Changes

Monitor one or more files in a designated folder for specific events, with the option to include subfolders:

- Monitoring more than one file requires that the files are named according to a pattern that can be represented using asterisks (\*) as wildcards.
- Leo will monitor the specified files until:
  - One of the defined events occurs; or
  - The command reaches the timeout limit you have specified
- Events to be monitored can include one or more of the following:
  - File created or renamed (to match the specified pattern)
  - File modified
  - File deleted

This command can be particularly useful if a wizard requires that a certain file be added or updated prior to proceeding.

### Using the MONITOR FILE CHANGES command

The screenshot shows a dialog box titled "Monitor file changes" with the following fields and options:

- 1** Root folder: [Text input field]
- Make sure the folder is accessible for all end users.
- Include sub-folders
- 2** File name to monitor: [Text input field]
- Enter file name or pattern (use stars: \*). Example: \*.xlsx
- File created (or renamed)
- File modified
- File deleted
- 3** (Callout for the event checkboxes)
- 4** Return file path in variable: [Dropdown menu]
- Type a variable name
- Timeout: 5 [Spin box] minutes
- 5** (Callout for the timeout field)
- 6** Error handling [Dropdown menu]
- 6** (Callout for the error handling dropdown)

Buttons: OK, Cancel

- 1 Enter the **full path** of the top-level folder in which you would like to monitor files; **and** indicate whether or not to also monitor files in subfolders
- 2 **To monitor a single file:** Enter the name of a the file to monitor; **or**  
**To monitor multiple files:** Enter a naming pattern for the files to monitor, using asterisks (\*) as wildcards to represent one or more characters in the file name, for example:
  - The pattern `file*.txt` will monitor `file.txt`, `file1.txt`, `file48.txt`, `file1948.txt`, etc.
  - The pattern `file.*` will monitor `file.txt`, `file.docx`, `file.xlsx`, `file.png`, etc.
- 3 Select one or more events for which you would like to monitor
- 4 Enter the name of the variable into which you'd like to place the file path of the new, renamed, modified, or deleted file
- 5 Indicate if you would like Leo to stop monitoring (i.e., timeout) after a certain number of minutes
- 6 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 9: Folder Commands

In this chapter:

Create a Folder .....	148
Get Folder Location .....	149
Get Files .....	150
Does Folder Exist .....	152
Is Folder Empty .....	153
Copy a Folder .....	154
Move a Folder .....	155
Rename a Folder .....	156
Delete a Folder .....	157
Monitor Folder Changes .....	158

## Create a Folder

Create a new folder and place its path in a new or existing variable.

### Using the CREATE A FOLDER command

- 1 Enter the name of the variable into which you'd like to place the path of the new folder
- 2 Indicate if you'd like the folder to be created:
  - In a parent folder other than the Windows Temp folder
  - Using a custom folder name

If you elect not to specify these options, the new folder will be created by default in the Windows Temp folder, with a unique name.
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



#### TIP

##### Don't forget to clean house!

Despite its name, Windows does not automatically delete folders and files from the Windows Temp folder. From a safety perspective, this is great. But it does mean that your Windows Temp folder could become quite huge (and something of a resource hog) unless you manually clean it out from time to time.

## Get Folder Location

Retrieve the path of a system folder and place it in a variable.

Folders for which this command is available:

- My Documents
- Desktop
- Program Files
- System
- Windows
- Cache



### NOTE

The location of certain system folders varies by Windows user. This command retrieves the folder location for the **current user** at the time the wizard is run.

## Using the GET FOLDER LOCATION command

Get folder location

Get the  folder location into the variable: 1

2

OK Cancel

- 1 Select the system folder whose location you would like to retrieve
- 2 Enter the name of the variable into which you'd like to place the folder's path

## Get Files

Retrieve a list of files contained within a designated top-level (i.e., root) folder, with the option to include subfolders.

### Using the GET FILES command

1 Enter the **full path** of the root folder from which you would like to retrieve a list of files

2 **To retrieve a list of all files:** Leave this field blank;  
**To retrieve a single file:** Enter the name of the file; *or*  
**To retrieve a list of matching files:** Enter a naming pattern for the files, using asterisks (\*) as wildcards to represent one or more characters in the file name, for example:

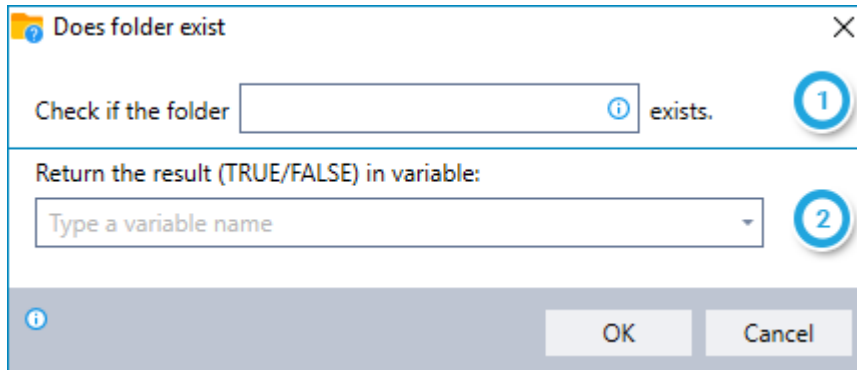
- The pattern `file*.txt` will retrieve `file.txt`, `file1.txt`, `file48.txt`, `file1948.txt`, etc.
- The pattern `file.*` will retrieve `file.txt`, `file.docx`, `file.xlsx`, `file.png`, etc.

- 3 Indicate whether or not to list files from subfolders
- 4 Indicate whether or not to include file paths (folder names) in the list
- 5 Select how you would like to sort the list
- 6 Enter the name of the variable into which you'd like to place the list
- 7 Enter the delimiter to use to separate the name of each file in the list

## Does Folder Exist

Check to see if a folder exists and place the result of the check (TRUE/FALSE) into a variable.

### Using the DOES FOLDER EXIST command



Does folder exist

Check if the folder  exists. 1

Return the result (TRUE/FALSE) in variable:  
 2

OK Cancel

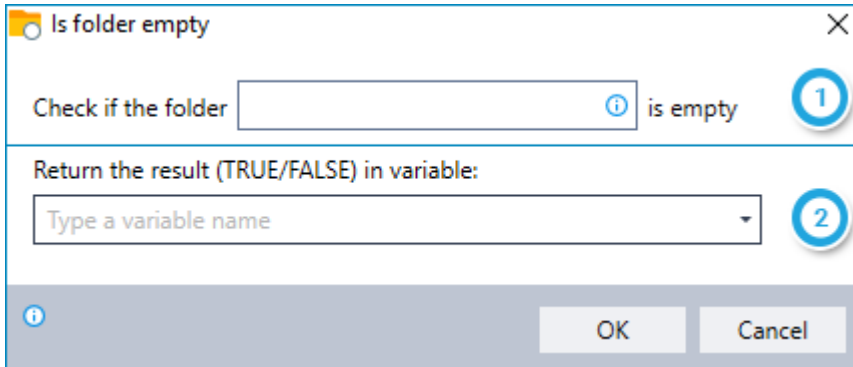
- 1 Enter the **full path** of the folder for whose existence you would like to check
- 2 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)



## Is Folder Empty

Check to see if a folder is empty and place the result of the check (TRUE/FALSE) into a variable. It's a great idea to use this command before you use the **DELETE A FOLDER** command.

### Using the IS FOLDER EMPTY command



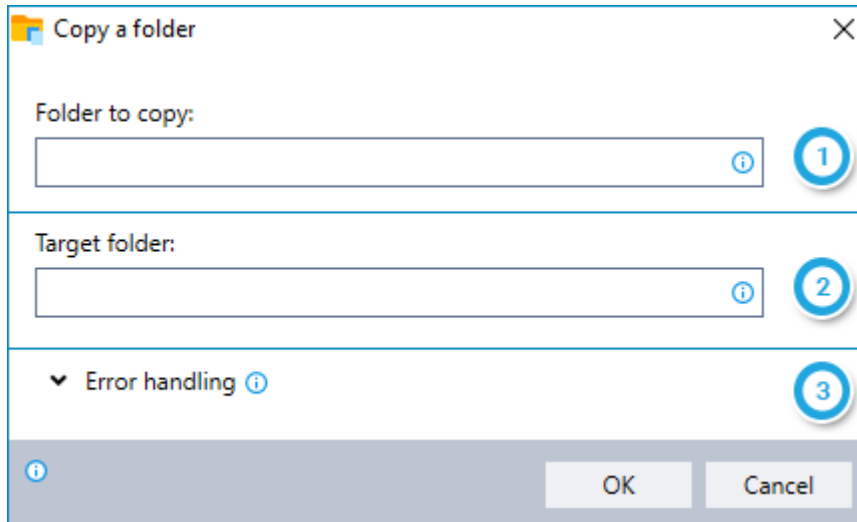
The screenshot shows a dialog box titled "Is folder empty". It has a close button (X) in the top right corner. The main area contains a text input field for the folder path, followed by the text "is empty". Below this is a section titled "Return the result (TRUE/FALSE) in variable:" with a dropdown menu that currently shows "Type a variable name". At the bottom of the dialog are "OK" and "Cancel" buttons. A blue circle with the number "1" is positioned to the right of the folder path input field, and another blue circle with the number "2" is positioned to the right of the variable name dropdown.

- 1 Enter the **full path** of the folder you would like to check
- 2 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)

## Copy a Folder

Copy a folder from its current location to a location you choose.

### Using the COPY A FOLDER command

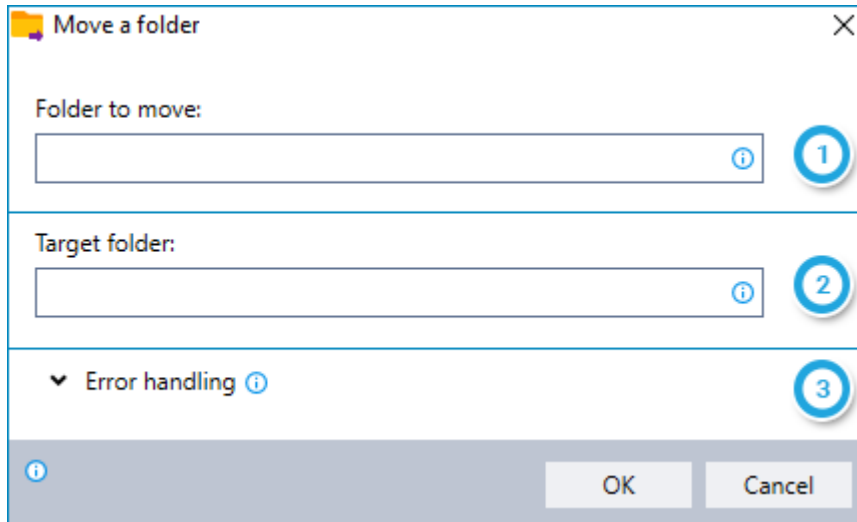


- 1 Enter the **full path** of the folder you would like to copy
- 2 Enter the **full path** of the target folder to which you would like to copy the folder
  - If a folder of the same name already exists in the target folder, the folders will be merged. If duplicate file names exist when the folders are merged, files from the source folder will overwrite files with the same names in the target.
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Move a Folder

Move a folder from its current location to a location you choose.

### Using the MOVE A FOLDER command



- 1 Enter the **full path** of the folder you would like to move
- 2 Enter the **full path** of the target folder to which you would like to move the folder
  - If a folder of the same name already exists in the target folder, the folders will be merged. If duplicate file names exist when the folders are merged, files from the source folder will overwrite files with the same names in the target.
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Rename a Folder

Rename an existing folder.

### Using the RENAME A FOLDER command

The screenshot shows a dialog box titled "Rename a folder" with a close button (X) in the top right corner. It contains three input sections:
 

- Folder to rename:** A text input field with a blue circular callout '1' to its right.
- New name:** A text input field with a blue circular callout '2' to its right.
- Error handling:** A dropdown menu with a blue circular callout '3' to its right.

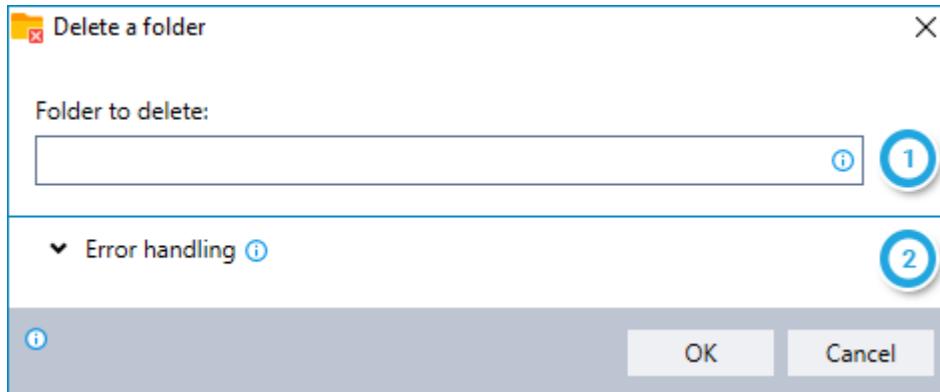
 At the bottom of the dialog are "OK" and "Cancel" buttons. A small information icon (i) is located in the bottom left corner of the dialog's border.

- 1 Enter the **full path** of the folder you would like to rename
- 2 Enter the new name you would like to give the folder
  - If you enter only the folder name (with no path), the folder will be renamed and remain in its current location
  - If you enter a new path along with the folder name, the folder will be renamed **AND** moved to the new location you entered
    - If a folder of the same name already exists in the new location, the folder will **NOT be renamed or moved**
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Delete a Folder

Delete an existing folder.

### Using the DELETE A FOLDER command



- 1 Enter the **full path** of the folder you would like to delete
- 2 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.



#### CAUTION

**Make sure you really want to delete it!**

The **DELETE A FOLDER COMMAND** doesn't put the deleted folder into the Recycle Bin... it deletes it permanently.

Also note that the contents of the folder will be deleted along with the folder itself. If you want to be sure that a folder doesn't contain any files before you delete it, use the **IS FOLDER EMPTY** command.

## Monitor Folder Changes

Monitor one or more folders within a designated top-level (i.e., root) folder for specific events, with the option to include subfolders:

- Monitoring more than one folder requires that the folders are named according to a pattern that can be represented using asterisks (\*) as wildcards.
- Leo will monitor the specified folders until:
  - One of the defined events occurs; or
  - The command reaches the timeout limit you have specified
- Events to be monitored can include one or more of the following:
  - Folder created or renamed (to match the specified pattern)
  - Folder deleted

This command can be particularly useful if a wizard requires that a certain folder be created prior to proceeding.

### Using the MONITOR FOLDER CHANGES command

**Monitor folder changes** [X]

Root folder:  ⓘ 1  
 Make sure the folder is accessible for all end users.  
 Include sub-folders

---

Folder name to monitor (optional):  ⓘ 2  
 Enter folder name or pattern (use stars: \*). Example: Logs\*

Folder created (or renamed) 3  
 Folder deleted

Return folder path in variable:  ⓘ 4

Timeout:  minutes 5

▼ Error handling ⓘ 6

ⓘ OK Cancel

- 1 Enter the **full path** of the root folder in which you would like to monitor folders; **and** indicate whether or not to also monitor subfolders
- 2 **To monitor the root folder:** Leave this field blank;  
**To monitor a single folder within the root folder:** Enter the name of the folder to monitor; **or**  
**To monitor multiple folders within the root folder:** Enter a naming pattern for the folders to monitor, using asterisks (\*) as wildcards to represent one or more characters in the folder name, for example:
  - The pattern `log folder*` will monitor `log folder 2017`, `log folder-temp`, `log folders`, etc.
- 3 Select one or more events for which you would like to monitor
- 4 Enter the name of the variable into which you'd like to place the path of the new, renamed, or deleted folder
- 5 Indicate if you would like Leo to stop monitoring (i.e., timeout) after a certain number of minutes
- 6 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 10: Scanned Document Analysis (OCR) Commands

In this chapter:

Document Analysis .....	161
Document: Get Text .....	165
Document: Get Value .....	166
Document: Get Checkbox State .....	168
Document: Does Word Exist .....	170
Document: Get Date/Time .....	171
Document: Save as Image .....	173
Document: Save to Excel .....	174
Convert to Text (SmartScan+) .....	176
Convert to Text (Tesseract) .....	177



## Document Analysis

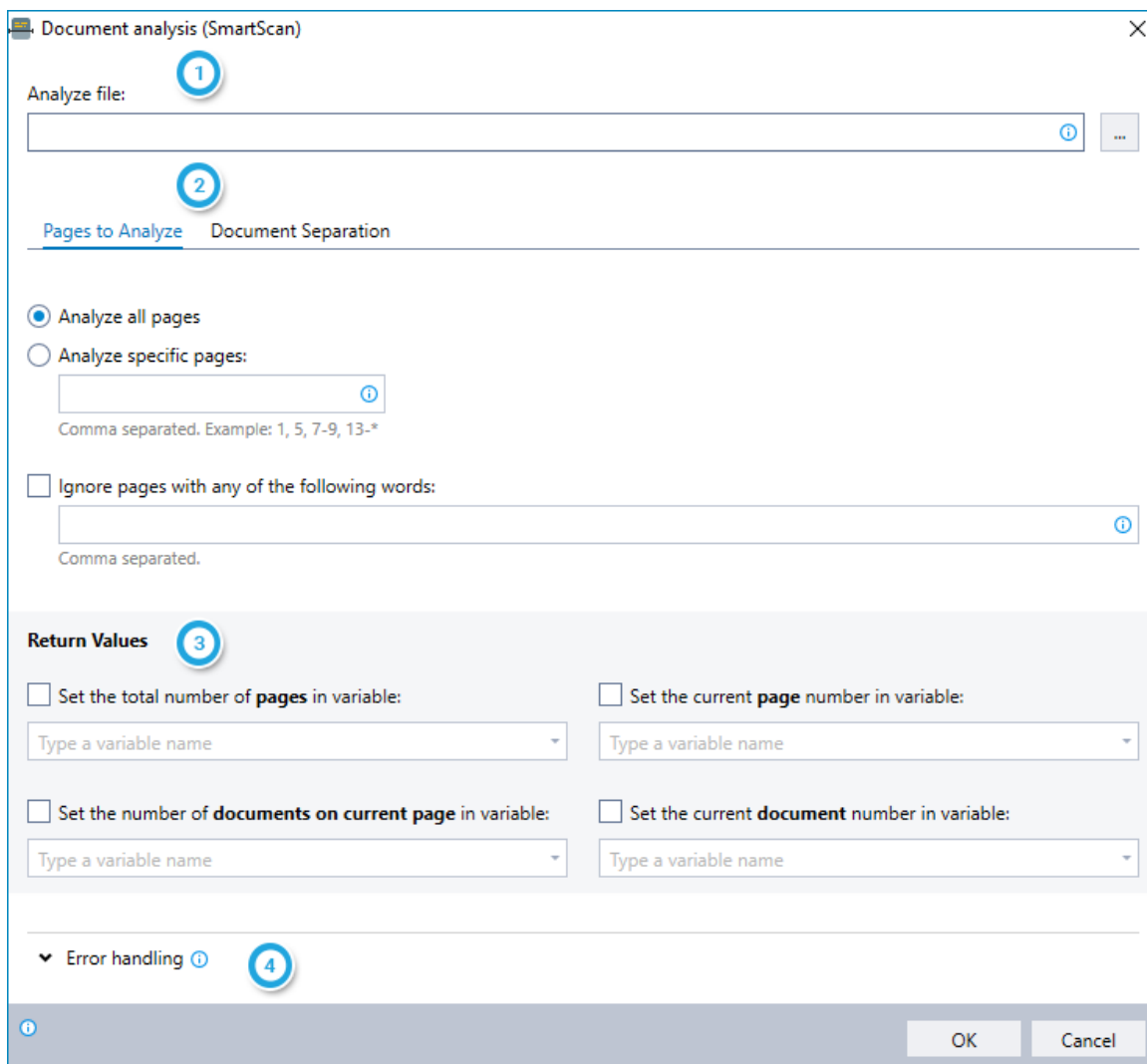
### SmartScan Feature

Analyze a file containing scanned documents for the purpose of performing a sequence of actions on each document.

### Using the DOCUMENT ANALYSIS command

#### Step #1 - Analyze & separate the documents

The first step in using the **DOCUMENT ANALYSIS** command is to analyze and separate the documents on which the specified actions will be performed.



1 Select the file containing scanned documents to analyze

2 Enter required options in 2 tabs:

**Pages to Analyze:**

- Choose whether to analyze all pages of the file or enter specific pages to analyze
- Indicate if you wish to ignore pages containing specified words

The screenshot shows the 'Pages to Analyze' tab selected. It contains two radio button options: 'Analyze all pages' (selected) and 'Analyze specific pages:'. Below the second option is a text input field with a help icon and the text 'Comma separated. Example: 1, 5, 7-9, 13-\*'. There is also a checkbox option 'Ignore pages with any of the following words:' followed by another text input field with a help icon and the text 'Comma separated.'

**Document Separation:**

- Choose the method for separating the scanned documents to be analyzed

The screenshot shows the 'Document Separation' tab selected. It features three options, each with a radio button and an icon: 'Do Not Separate' (stack of papers icon), 'By Page' (three separate page icons), and 'By Inner Document' (grid of page icons). Below each option is a brief description of the method.

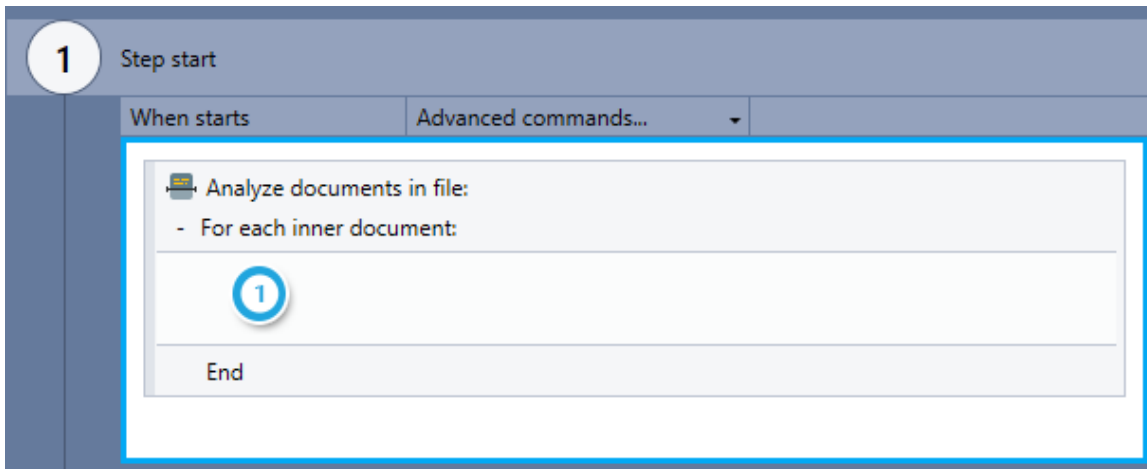
3 Indicate if you wish to place information about total pages/documents and current page/document into variables

- **Note:** Available fields will vary based on the method for separating documents selected in step 2 above

4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Step #2 - Define the actions

Upon adding the **DOCUMENT ANALYSIS** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



### 1 Enter the action(s) the wizard should take on each analyzed document

- You can do this by dragging the required Advanced Command(s) directly into the container



## NOTES

### Loop-the-loop

Leo perform the actions defined within the container by **looping** through each page or inner document (i.e., it will perform the complete sequence of actions on a single page/document, then move on to perform the sequence on each remaining page/document in turn).

### No limits

You can use any available Advanced Command within the **DOCUMENT ANALYSIS** container (i.e., don't feel limited to using just the Scanned Document Analysis commands!)

A combination of these two notes leads us to a...



## TIP

### Break that loop!

Under certain conditions, you may want to break and/or restart the loop created by the **DOCUMENT ANALYSIS** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

## Document: Get Text

### SmartScan Feature

Place text from a scanned document into a variable.



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: GET TEXT command

Document: Get text

Set document text into a variable:

Type a variable name

Preserve formatting  
Keep line breaks and text indentation as they appear in the original document

OK Cancel

- 1 Enter the name of the variable into which to place the text
- 2 Indicate whether to preserve formatting (line breaks and indentation) from the scanned document

## Document: Get Value

### SmartScan Feature

Retrieve a value from a scanned document by searching for it next to a specified word (the "label word").



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: GET VALUE command

The screenshot shows the 'Document: Get value' dialog box with the following fields and options:

- 1** Search for the word: [Text input field]
- 2** And read value from: A diagram with four radio buttons labeled 'Above', 'Left side', 'Right side', and 'Below'. 'Right side' is selected.
- 3**  Read entire line  
 Allow close match ⓘ  
Required accuracy [Slider] 0%
- 4** Value type: [Dropdown menu showing 'Text']
- 5** Set the value in variable: [Dropdown menu showing 'Type a variable name']
- 6** Error handling [Dropdown menu]

Buttons: [OK] [Cancel]

- 1 Enter all possible label words (separated by commas)
- 2 Choose where the value to be retrieved appears in relation to the label word (right side, left side, above, or below); **and**  
Indicate whether to search for the value in the entire line in which the label appears
  - If the option to read the entire line is not selected, Leo will search only in close proximity to the label word
- 3 Indicate whether you wish to allow **close matching** of the label word(s); **and** the level of accuracy required for the close match to be accepted
- 4 Select whether the retrieved value should be treated as text or a number
- 5 Enter the name of the variable into which to place the retrieved value
- 6 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Document: Get Checkbox State

### SmartScan Feature

Determine whether a checkbox in a scanned document is checked or unchecked by searching for it next to a specified word (the "label word").



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: GET CHECKBOX STATE command

Document: Get checkbox state

Determine if the following checkbox is checked

Checkbox is located to the right of the words:

Specify all possible words (comma separated).

Allow close match

Required accuracy 100%

Add another condition

Set the value in variable:

Type a variable name

▼ Error handling

OK Cancel

- 1 Choose where the relevant checkbox appears in relation to the label word (to the right, to the left, above, or below); **and** Enter all possible label words (separated by commas)
- 2 Indicate whether you wish to allow **close matching** of the label word(s); **and** the level of accuracy required for the close match to be accepted
- 3 (Optional) Enter a second condition for determining the location of the label word



- 4 Enter the name of the variable into which you'd like Leo to place the result (will be either TRUE or FALSE, as applicable)
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Document: Does Word Exist

### SmartScan Feature

Check for whether a specified word (or one of a set of specified words) appears in a scanned document.



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: DOES WORD EXIST command

- 1 Enter all possible words to check for (separated by commas)
- 2 Enter the name of the variable into which you'd like Leo to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Document: Get Date/Time

### SmartScan Feature

Retrieve a date (and time, if it appears) from a scanned document.



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: GET DATE/TIME command

- 1 Select which date to retrieve from the scanned document: the last date, the first date, or any date that appears
- 2 Choose whether day or month appears first in numeric date patterns
- 3 (Optional): Enter a custom pattern by which Leo should identify dates
  - Use a regular expression to enter the pattern (To learn more, see [What is a regular](#)

expression?)

- 4 Enter the years during which the retrieved date must occur
- 5 Enter the name of the variable into which to place the retrieved date
- 6 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Document: Save as Image

### SmartScan Feature

Save a scanned document as an image file.



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: SAVE AS IMAGE command

- 1 Select the folder into which to save the image file
- 2 Enter the name of the variable into which to save the full file path of the image file
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Document: Save to Excel

### SmartScan Feature

Save text from a scanned document into an Excel spreadsheet.



#### NOTE

This command can be used only within an **DOCUMENT ANALYSIS** container.

### Using the DOCUMENT: SAVE TO EXCEL command

The screenshot shows a dialog box titled "Document: Save to Excel". It contains the following fields and controls:

- Document layout:** A dropdown menu with "Preserve layout" selected. A blue circle with the number "1" is next to it.
- Save file in folder:** A text input field with an information icon and a browse button ("..."). A blue circle with the number "2" is next to it.
- Return full file path in variable:** A dropdown menu with "Type a variable name" selected. A blue circle with the number "3" is next to it.
- Error handling:** A dropdown menu with a downward arrow. A blue circle with the number "4" is next to it.
- At the bottom, there are "OK" and "Cancel" buttons.

1 Select option for laying out the document in Excel:

Preserve layout	Maintain rows and columns from scanned document. For cells in the scanned document containing more than one line of text, place each line of text into a separate row.
Preserve division into rows and columns	Maintain rows and columns from scanned document. For cells in the scanned document containing more than one line of text, maintain as a single cell (with line breaks).
Do not preserve layout	Place all text from scanned document into a single column in Excel

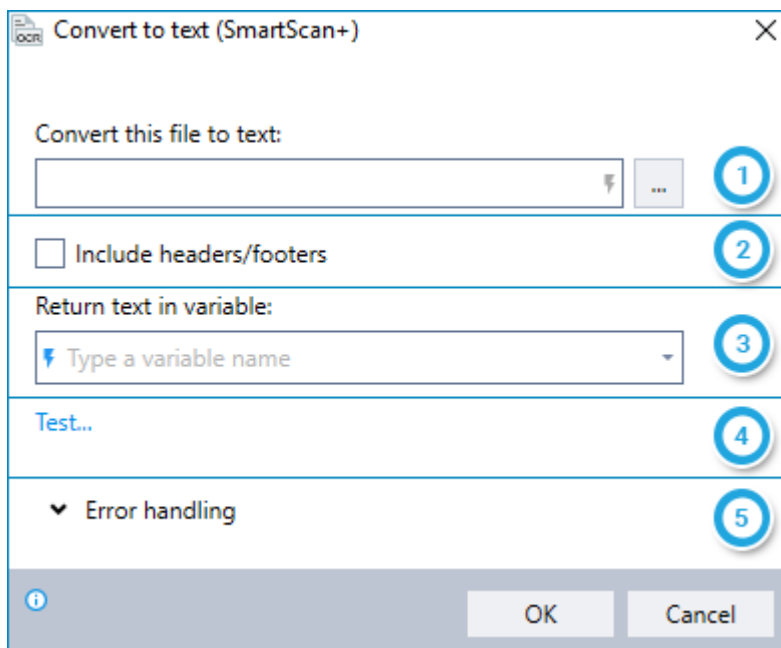
- 2 Select the folder into which to save the Excel file
- 3 Enter the name of the variable into which to save the full file path of the Excel file
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Convert to Text (SmartScan+)

### SmartScan+ Feature

Convert an image or PDF file to text using the ABBYY recognition engine (OCR) and place the text into a new or existing variable.

### Using the CONVERT TO TEXT (SMARTSCAN+) command



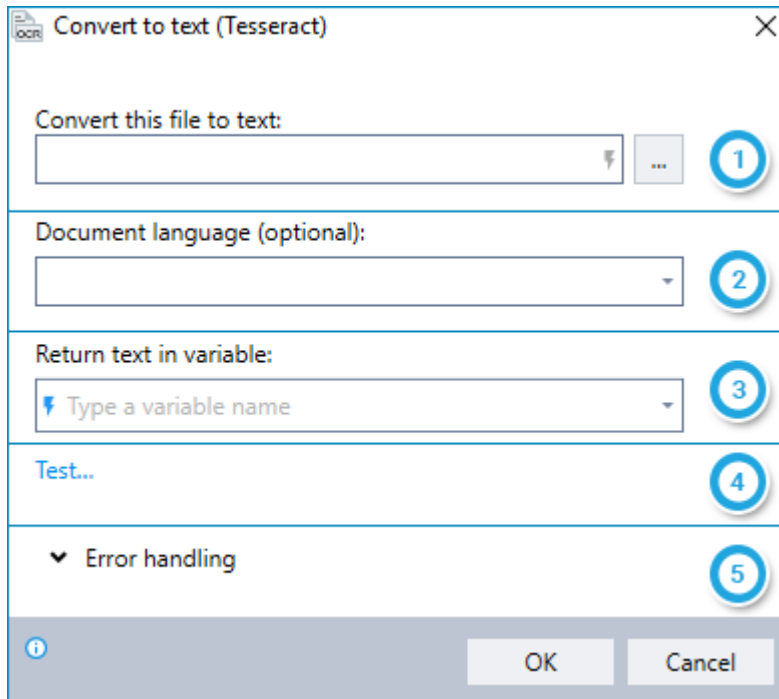
- 1 Select the PDF or image file to convert
- 2 Indicate whether you'd like to include the text of the original document's headers/footers (if any)
- 3 Enter the name of the variable into which you'd like to place the recognized text
- 4 (Optional) Test the text recognition results
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Convert to Text (Tesseract)

Convert an image or PDF file to text using the Tesseract recognition engine (OCR) and place the text into a new or existing variable.

### Using the **CONVERT TO TEXT (TESSERACT)** command



- 1 Select the PDF or image file to convert
- 2 (Optional) Select the primary language of the text in the selected file
- 3 Enter the name of the variable into which you'd like to place the recognized text
- 4 (Optional) Test the text recognition results
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 11: Digital PDF Analysis

## Commands

In this chapter:

Analyze Digital PDF File .....	179
Page: Get Text .....	182

## Analyze Digital PDF File

Analyze a digital PDF file for the purpose of performing a sequence of actions on each page.

### Using the ANALYZE DIGITAL PDF FILE command

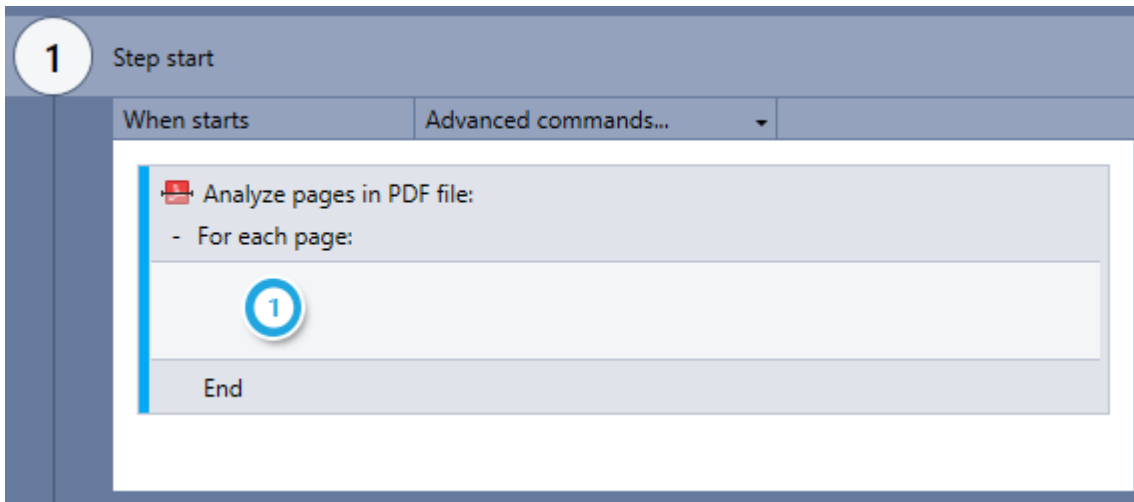
#### Step #1 - Analyze & identify the pages

The first step in using the **ANALYZE DIGITAL PDF FILE** command is to analyze and identify the pages on which the specified actions will be performed.

- 1 Select the file to analyze
- 2 Choose whether to analyze all pages of the file or enter specific pages to analyze
- 3 Indicate if you wish to ignore pages containing specified words
- 4 Indicate if you wish to place information about total pages and current page into variables
- 5 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Step #2 - Define the actions

Upon adding the **ANALYZE DIGITAL PDF FILE** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



1 Enter the action(s) the wizard should take on each analyzed page

- You can do this by dragging the required Advanced Command(s) directly into the container



### NOTES

#### Loop-the-loop

Leo perform the actions defined within the container by **looping** through each page (i.e., it will perform the complete sequence of actions on a single page, then move on to perform the sequence on each remaining page in turn).

#### No limits

You can use any available Advanced Command within the **ANALYZE DIGITAL PDF FILE** container (i.e., don't feel limited to using just the Digital PDF Analysis commands!)

A combination of these two notes leads us to a...



**TIP**

**Break that loop!**

Under certain conditions, you may want to break and/or restart the loop created by the **ANALYZE DIGITAL PDF FILE** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

## Page: Get Text

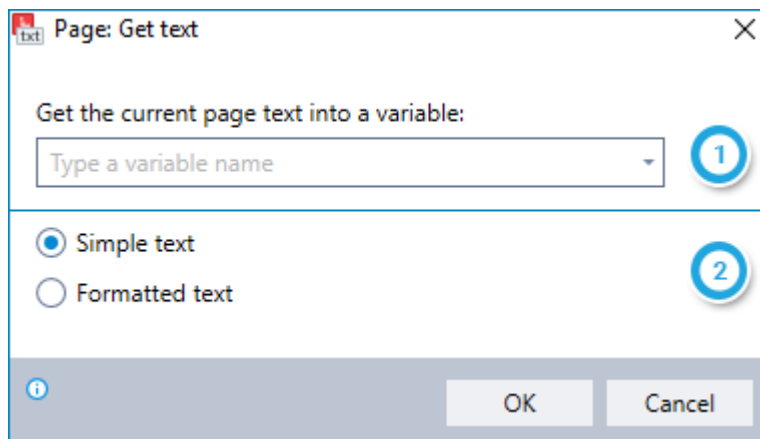
Place text from the current page of a digital PDF file into a variable.



### NOTE

This command can be used only within an **ANALYZE DIGITAL PDF FILE** container.

## Using the PAGE: GET TEXT command



- 1 Enter the name of the variable into which to place the text
- 2 Indicate whether to save the text as simple or formatted text

# CHAPTER 12: External Data Commands

In this chapter:

Get From Clipboard .....	184
Place in Clipboard .....	185
Copy Active Field Value .....	186
Get OS Version .....	187
Get Input Language .....	188
Read From Registry .....	189
Remove From Registry .....	191
Write to Registry .....	192
Get Environment Variable .....	193
Get Windows Event Log Data .....	194
Set BI Field .....	196
Report User Action .....	197
Query XML .....	198

## Get From Clipboard

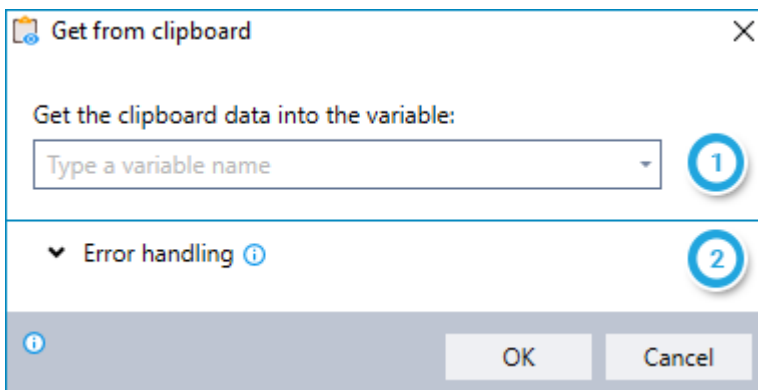
Retrieve the contents of the Windows clipboard and place them into a new or existing variable.



### NOTE

Only text contained in the clipboard can be copied. Images will be ignored.

## Using the GET FROM CLIPBOARD command



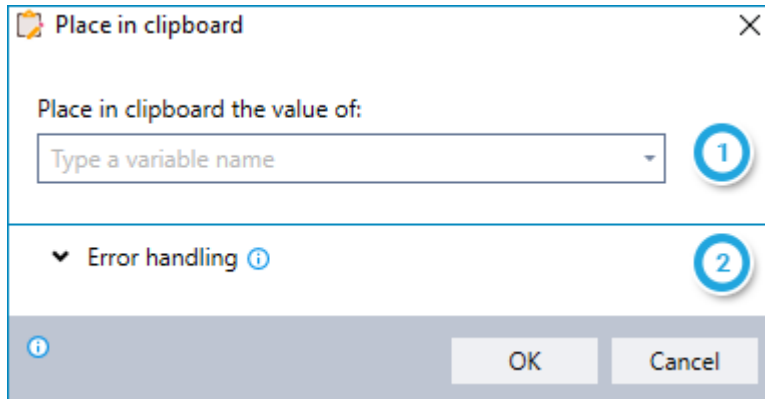
- 1 Enter the name of the variable into which you'd like to place the contents of the clipboard
- 2 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Place in Clipboard

Copy a value stored in a variable into the Windows clipboard.

### Using the PLACE IN CLIPBOARD command



- 1 Enter the name of the variable whose value you would like to copy into the clipboard
- 2 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Copy Active Field Value

Copy the value from the active field in the wizard's active application and place it into a new or existing variable. You can choose to copy the value of the entire field or just a single line.



### TIP

Don't forget to make sure the field you need is selected before using this command!

Either <TAB> over to it or click inside it, then you'll be set to go.

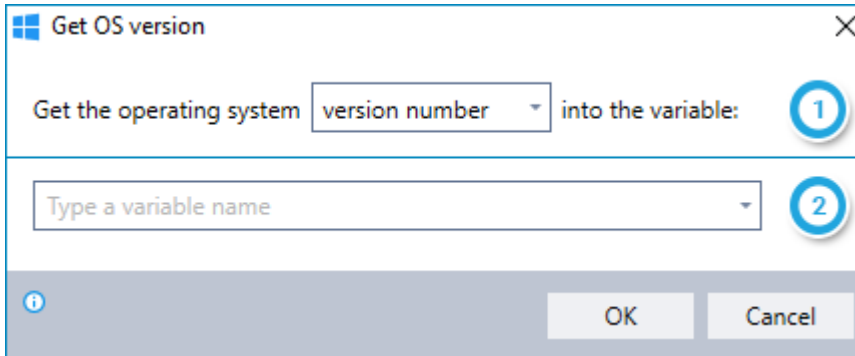
## Using the COPY ACTIVE FIELD VALUE command

- 1 Enter the name of the variable into which you'd like to place the value of the active field
- 2 Choose whether you'd like to copy:
  - The value of the entire field (the equivalent of **Ctrl + A**); *or*
  - The value of a single line (the equivalent of **Home, Shift + End**)
    - The single line copied is the line in which the cursor is located
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get OS Version

Retrieve the current Windows version (either by number or name) and place it into a new or existing variable.

### Using the GET OS VERSION command

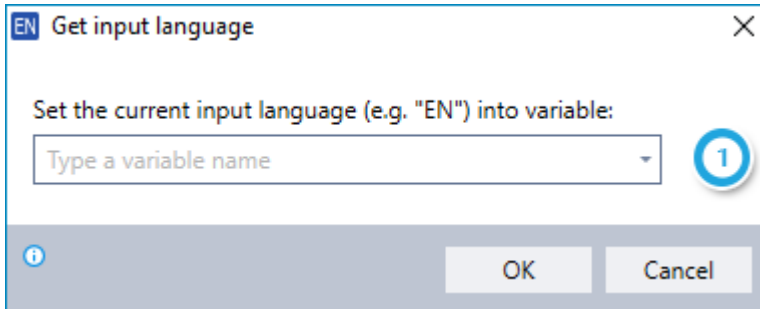


- 1 Choose whether you'd like to retrieve the Windows version number or version name
- 2 Enter the name of the variable into which you'd like to place the result

## Get Input Language

Retrieve the code for the currently selected Windows input language and place it into a new or existing variable.

### Using the GET INPUT LANGUAGE command



- 1 Enter the name of the variable into which you'd like to place the language code

## Read From Registry

Retrieve the **value data** for a Windows Registry key you specify and place it into a new or existing variable.



### NOTE

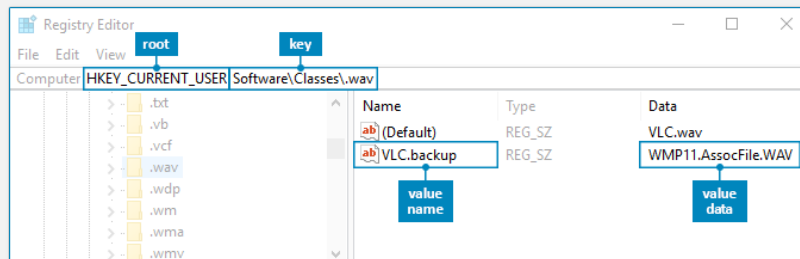
#### The anatomy of the Windows Registry

The Windows Registry contains information, settings, options, and other values for the programs and hardware installed on a Windows system (and for components of Windows itself). For example, when a new program is installed, settings such as its location, version, and how to start it are all added to the Windows Registry.

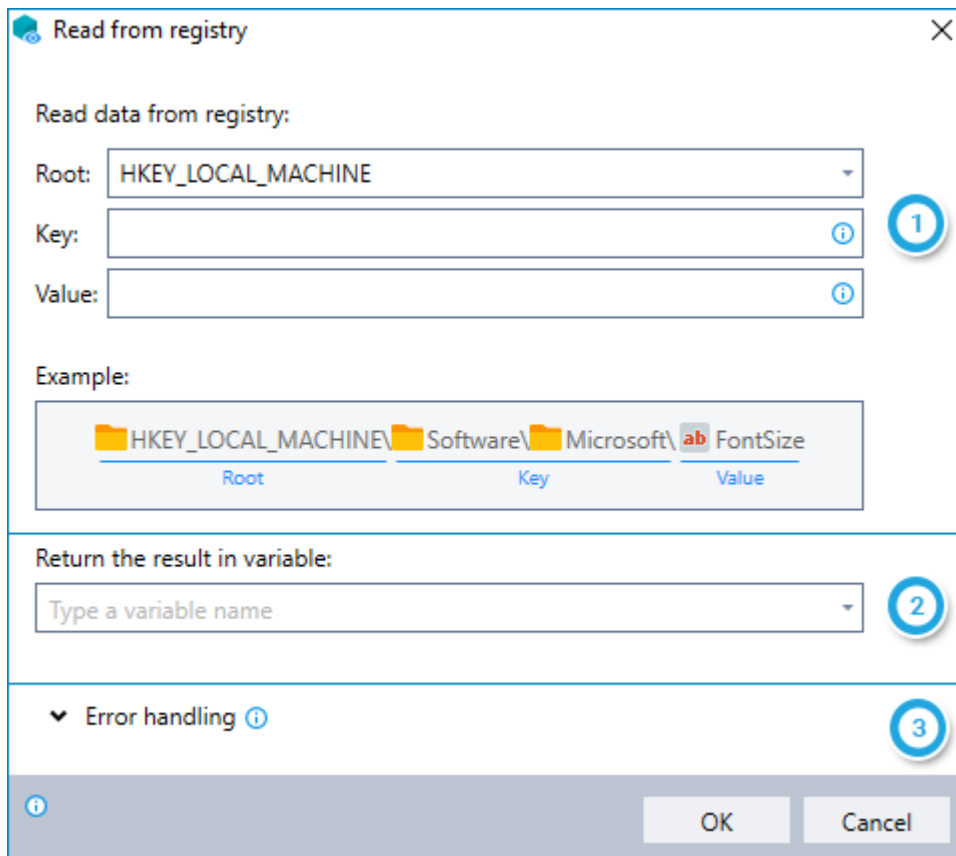
**Registry Editor** is the Windows component that allow you to view and edit the Registry. To access it, type `Run` from the Windows Start Menu, then `regedit` in the dialog box .

Be sure to take **extreme care** when editing (or even viewing) the Registry. It's at the very heart of the way Windows works, and changes can cause unexpected results. It's highly recommended to backup the Registry before making modifications so you can always revert to the way things were before.

Every entry in the Windows Registry is built according to a defined structure. Here's the way it works:



## Using the READ FROM REGISTRY command



- 1 Enter the **KEY** and **VALUE** for which you would like to retrieve value data
  - If you enter a **KEY** without specifying a **VALUE**, Leo will retrieve the data for the (*Default*) value (as shown in the Windows Registry Editor)
- 2 Enter the name of the variable into which you'd like to place the value data retrieved
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Remove From Registry

Remove specific data from the Windows Registry, either:

- A specified value (both the **value name** and **value data**); *or*
- A key

To learn more about how the Windows Registry is built, see [The anatomy of the Windows Registry](#).

### Using the REMOVE FROM REGISTRY command

#### 1 Specify the data you would like to remove

- If you enter a **VALUE**, Leo will remove both the **value name** and **value data** from the key you have specified
- If you enter just the **ROOT** and a **KEY** (without entering a **VALUE**), Leo will remove the specified **KEY**

#### 2 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Write to Registry

Inserts the contents of an existing variable as the **value data** for a Windows Registry key you specify.

To learn more about how the Windows Registry is built, see [The anatomy of the Windows Registry](#).

### Using the WRITE TO REGISTRY command

- 1 Enter the **KEY** and **VALUE** into which you would like to insert value data
  - If you enter a **KEY** without specifying a **VALUE**, Leo will write data for the *(Default)* value (as shown in the Windows Registry Editor)
- 2 Enter the name of the variable into which you'd like to place the value data retrieved
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Get Environment Variable

Retrieve the value of an existing Windows environment variable and place it into a new or existing Leo variable.



### NOTE

#### What is an environment variable?

Environment variables are strings that contain information such as drive, path, or file name. They control the behavior of various programs. For example, the TEMP environment variable specifies the location in which programs place temporary files.

To get a list of all existing environment variables and their values, open the Windows Command Prompt (type **cmd** from the Start Menu), then type **set** at the command line.

## Using the GET ENVIRONMENT VARIABLE command

- 1 Enter the name of the environment variable for which you would like to retrieve the value
- 2 Enter the name of the Leo variable into which you'd like to place the result

## Get Windows Event Log Data

Retrieve information from Windows Event Logs (according to a filter you define) and place it into a new or existing variable.



### NOTE

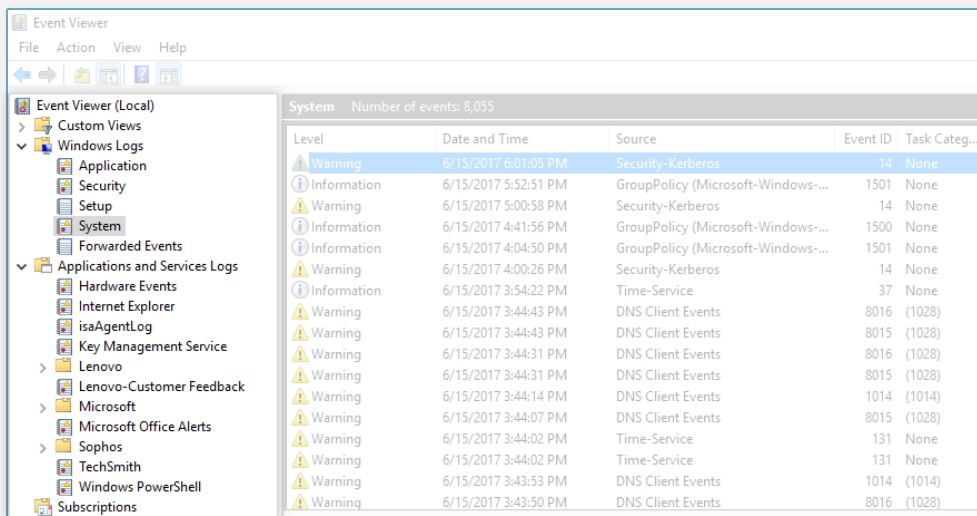
#### What are Windows Event Logs?

Any time your computer is running, Windows works in the background to monitor and log application and system messages – errors, warnings, and information.

Windows Event Logs are the ongoing records of all these messages.

Windows maintains several different logs for tracking information from different sources (Windows components, software, hardware, etc.) To take a look at any or all of them, head to the Windows Event Viewer simply by typing **event viewer** from the Windows Start Menu. From the left column, choose the event log you wish to see.

These are the same event logs you can query using the **GET WINDOWS EVENT LOG DATA** advanced command.



## Using the GET WINDOWS EVENT LOG DATA command

Get Windows event log data

Get the log events (XML) that match the following filter:

Event log name: System

Logged time:  last 1 minutes  last 0 hours  last 0 days

Source (provider):

Level:  Critical  Error  Warning  Information

Contains text (optional):

Return result in variable: Type a variable name

**Note:**  
Result is limited to maximum 5000 log events.

▼ Error handling

OK Cancel

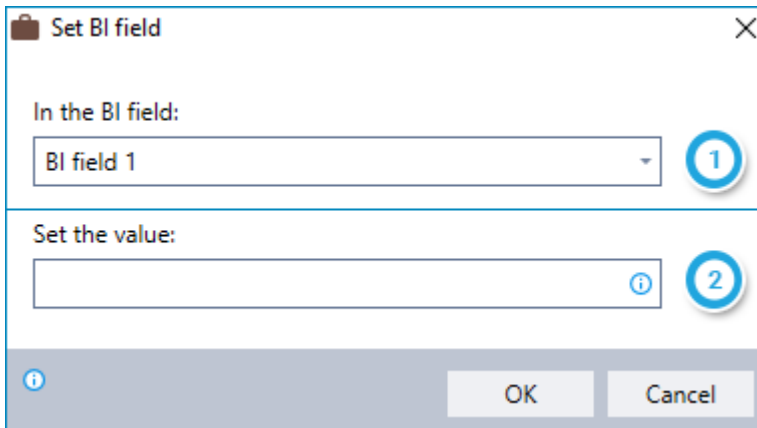
- 1 Define the filter for the events you would like to retrieve:
  - Name of the log to query
  - Time during which the events were logged
  - (Optional) Source of the event logged (i.e., the program or component that caused the event)
  - (Optional) Significance level of the event logged
  - (Optional) Free text filter
- 2 Enter the name of the variable into which you'd like to place the results
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Set BI Field

Report custom Business Intelligence (BI) information into the Leo database. This information can be retrieved later for detailed analysis of wizard usage and effectiveness.

For additional information about the BI information available and how to access it, see the *BI Fields in the Leo Database* section of the Leo Studio User Guide.

### Using the SET BI FIELD command



- 1 Select the BI field into which you would like to place the specified value
- 2 Enter the value you would like to place in the selected field (can be free text and/or values copied from different variables)

## Report User Action

Log the occurrence of any custom activity and value during the running of the wizard. (The action is logged upon reaching the point in the wizard where you have placed the **REPORT USER ACTION** command.)

You can later access this information by generating the **RUNTIME USER ACTIONS** report from the **Leo Report Generator**.

### Using the Report User Action command

- 1 Enter the name you would like to assign the action logged
- 2 Enter the value you would like to assign to the action logged (generally, the value of a variable)



#### EXAMPLE

##### Reporting on wizard activity

Assume you run a wizard daily that deletes files from the **c:\temp** folder (using the **DELETE FILE(S)** command). You can use the **REPORT USER ACTION** command to log that the deletion occurred, along with the names of the files deleted.

## Query XML

Use an XPTH query to extract specific information from XML data stored in a variable and place it into a new or existing variable.



### NOTE

#### What is XPATH?

XPath (XML Path Language) is a syntax or language used for finding elements in an XML document.

**Try it out:** To test your XPATH query while developing your wizard, simply click the **XPATH TESTER** link from within **QUERY XML** command.

## Using the QUERY XML command

The screenshot shows the 'Query XML' dialog box with the following fields and options:

- 1** XML source: A dropdown menu with the placeholder text 'Type a variable name'.
- 2** XPATH query: A text input field with a 'XPath Tester...' link and an information icon.
- 3** Selection options: Radio buttons for 'First result', 'Last result', and 'All results' (which is selected). Below these is a 'Delimiter' field with a comma character and an information icon.
- 4** Return values in variable: A dropdown menu with the placeholder text 'Type a variable name'.
- 5** Error handling: A dropdown menu with a downward arrow and an information icon.

At the bottom of the dialog are 'OK' and 'Cancel' buttons, and an information icon on the left.

- 1 Enter the name of the variable containing the XML data from which you want to extract information
- 2 Enter your XPATH query
- 3 Choose whether to retrieve the first matching result, the last result, or all results
  - When choosing to retrieve all results, enter the delimiter to use to separate each result
- 4 Enter the name of the variable into which you'd like to place the results
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 13: Email Commands

In this chapter:

Send Email Message .....	201
Get Email Messages .....	206
Email: Get Data .....	211
Email: Move to Folder .....	212
Email: Forward .....	213
Email: Reply .....	215
Email: Delete .....	216
Email: Mark as Read/Unread .....	217
Email: Save Attachments .....	218



## Send Email Message

Send an email message.

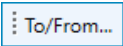
### Using the SEND EMAIL MESSAGE command

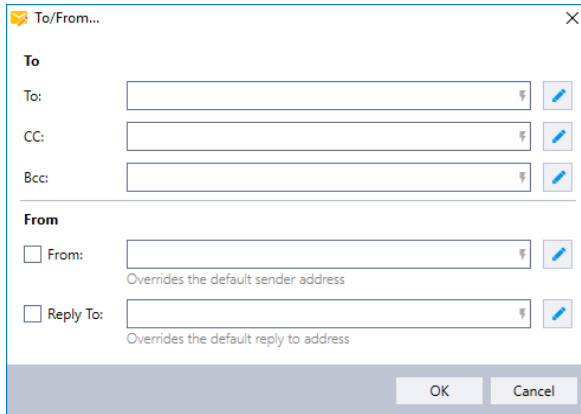
#### Message tab

The screenshot shows the 'Send email message' dialog box with the following components and numbered callouts:


- 1**: 'Message' tab and 'Email account' dropdown.
- 2**: Importance and text format options: 'To/From...', 'High importance', 'Low importance', 'Plain text', and 'Rich text'.
- 3**: 'To:', 'Subject:', and 'Attachments:' input fields.
- 4**: 'Email Body:' section with a rich text editor toolbar (font, size, bold, italic, underline, list, link, image, etc.) and a large text area.
- 5**: 'Return result in variable:' dropdown menu with the placeholder text 'Type a variable name'.
- 6**: 'Error handling' dropdown menu.

At the bottom of the dialog, there is an information icon (i), 'OK', and 'Cancel' buttons.

- 1 (Optional) Click the  button to access advanced address options for the email



**Advanced address options:**

- Click the  icon to:
  - Enter a long list of addresses (or addresses with display names) on the **TO:** ,**CC:** ,or **BCC:** lines
  - Enter addresses with display names on the **FROM:** or **REPLY TO:** lines

- 2 Choose whether to send the email in **Plain Text** or **Rich Text/HTML** format

- 3 Enter recipient email addresses, subject, and attachments:

- Separate multiple email addresses with commas
- Separate multiple attachments with commas
- Identify attachments by the full file name and file path

- 4 Enter the body of the email

- 5 Enter the name of the variable into which you'd like place the send result

- 6 Customize the codes/error messages for send results



**NOTE**

**Variables galore!**

All of the following fields can include free text and/or variables:

- Email addresses (**TO:** ,**CC:**, **BCC:**, **FROM:**, and **REPLY TO:**)
- Subject
- Attachments
- Email body

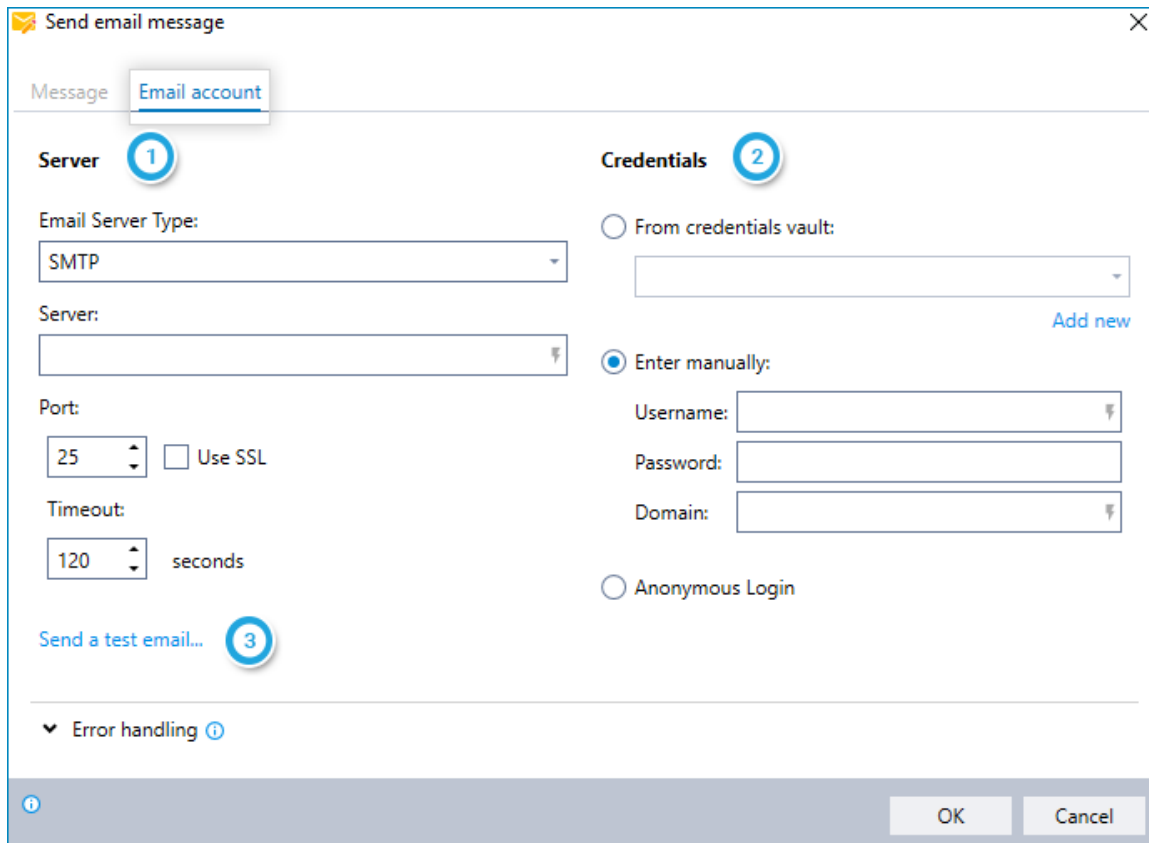
To include the value of a variable, indicate its name by typing it

between dollar signs (e.g., \$MyVar\$). When the wizard is run, the variable name will be replaced by its value.

### Email account tab

The settings available on the **Email account** tab vary for SMTP and Exchange servers.

#### SMTP servers



**1** Enter the settings for your email server

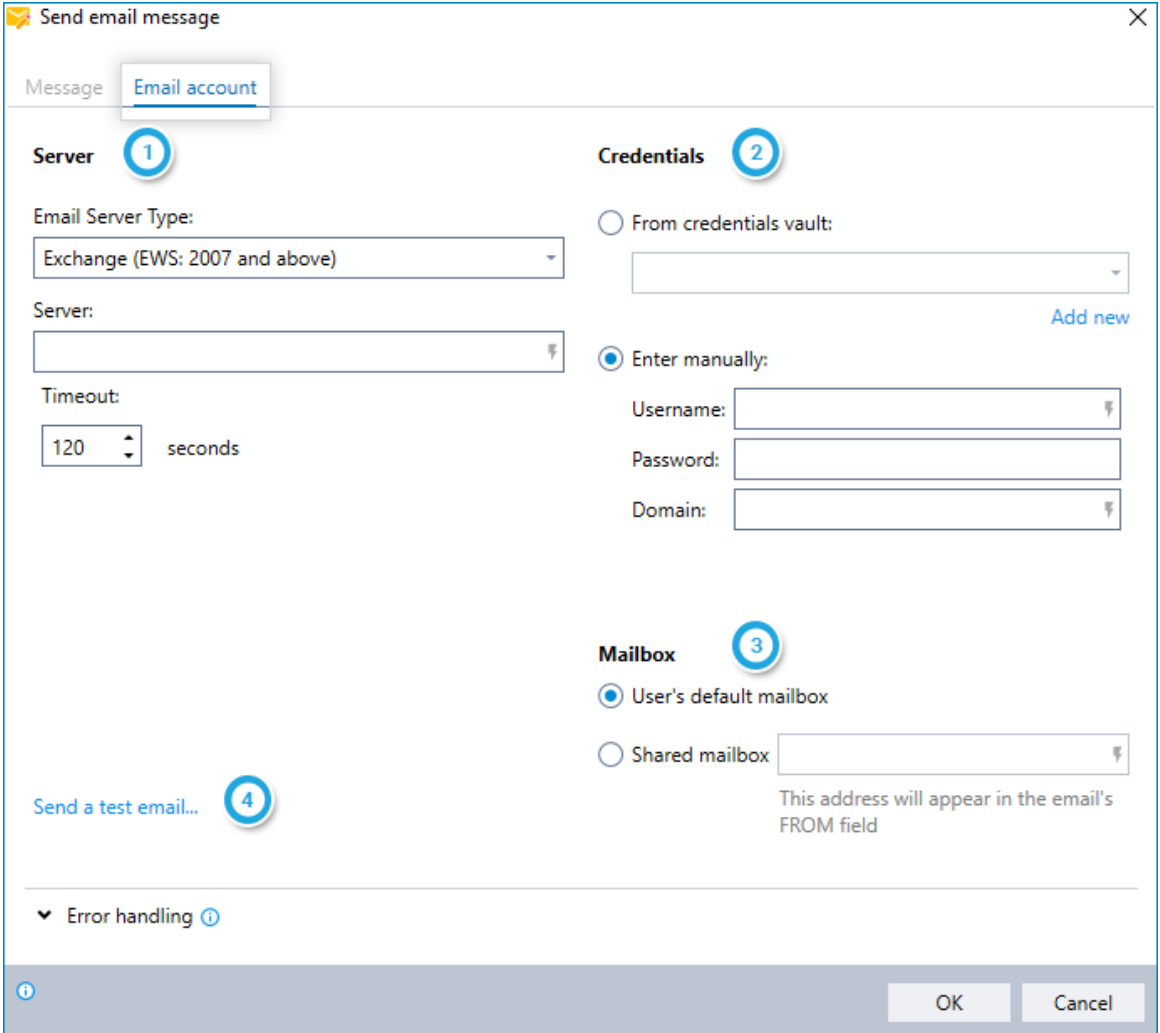
**2** Email server login credentials:

- Enter from the Leo credentials vault
- Enter manually; *or*
- Use anonymous login (for email accounts that allow this option)

**Note:** The login credentials used will provide the default **FROM:** and **REPLY TO:** addresses for the message. These addresses can be overridden by utilizing [advanced address options](#) in the **Message** tab.

3 (Optional) Send a test email to verify your settings

Exchange servers



1 Enter the settings for your email server

2 Email server login credentials:

- Enter from the Leo credentials vault; *or*
- Enter manually

3 Mailbox:

- Choose whether to send the message from the user's default mailbox (as entered in 2 above) or a shared mailbox
  - For a shared mailbox, enter the mailbox address

**Note:** The mailbox data entered will provide the default **FROM:** and **REPLY TO:** addresses for the message. These addresses can be overridden by utilizing [advanced address options](#) in the **Message** tab.



(Optional) Send a test email to verify your settings

## Get Email Messages

Identify all email messages matching a specified filter or a single message matching a key for the purpose of performing a sequence of actions on the message(s).

### Using the GET EMAIL MESSAGES command

#### Step #1 - Identify the messages

The first step in using the **GET EMAIL MESSAGES** command is to identify the messages on which the specified actions will be performed.

#### Get messages tab

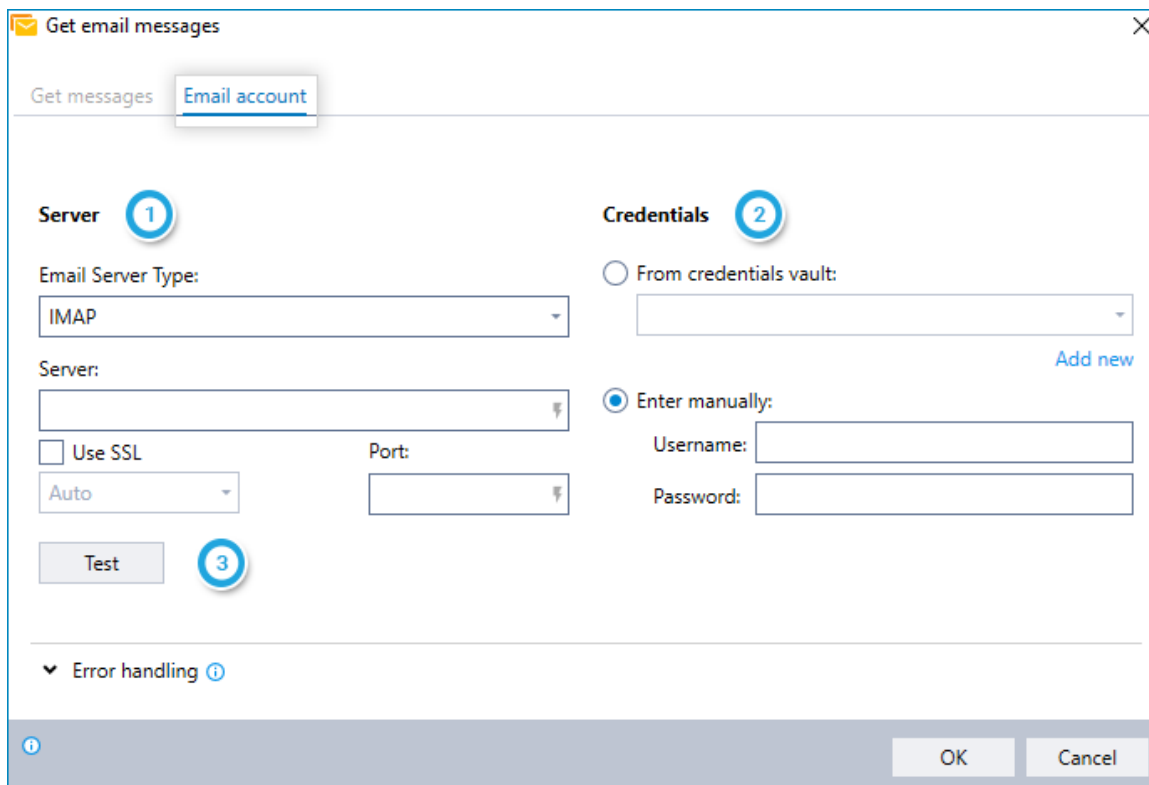
- 1 Define the filter by which the messages will be identified and retrieved; *or*
- 1a
- 1b Enter the name of the variable into which you have set an email key. (For more information, see [GET EMAIL TRIGGER INPUT.](#))
- 2 Enter the name of the variable into which you'd like place the result (the number of messages matching the filter criteria)

- 3 Indicate if you would like retry the search until at least one message is found and, if applicable, specify retry settings
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

**Email account tab**

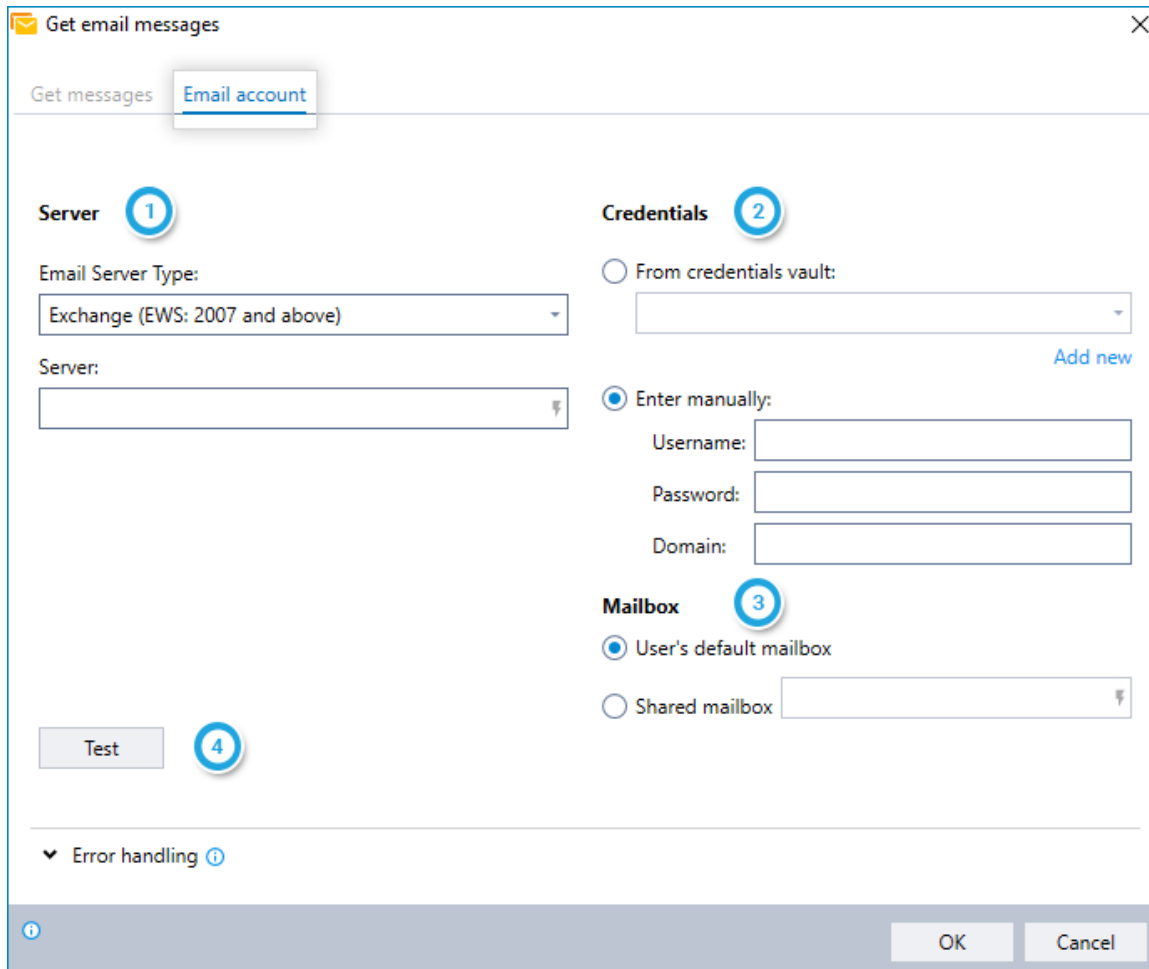
The settings available on the **Email account** tab vary for IMAP and Exchange servers.

**IMAP servers**



- 1 Enter the settings for your email server
- 2 Email server login credentials:
  - Enter from the Leo credentials vault; *or*
  - Enter manually
- 3 (Optional) Test your email account settings

### Exchange servers

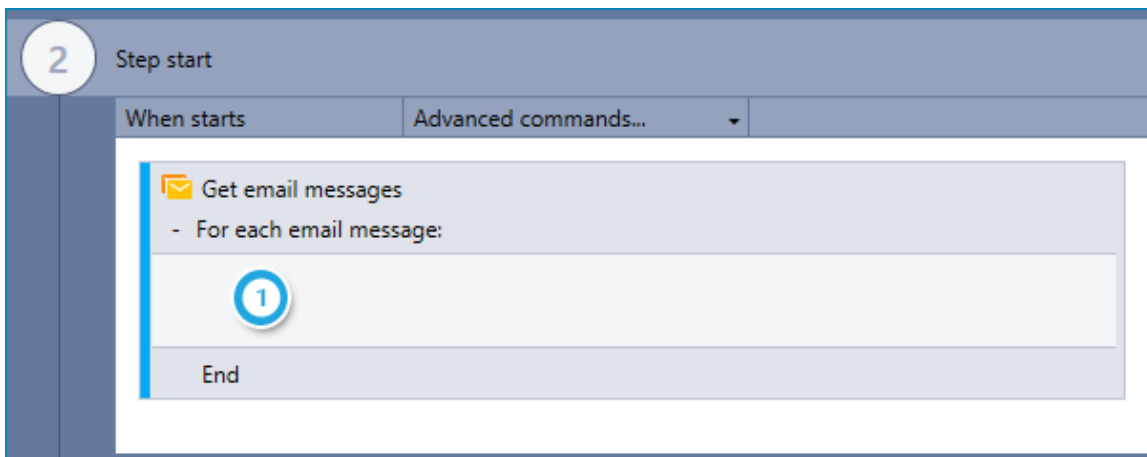


- 1 Enter the settings for your email server
- 2 Email server login credentials:
  - Enter from the Leo credentials vault; *or*
  - Enter manually
- 3 Mailbox:
  - Choose whether to retrieve the messages from the user's default mailbox (as entered in 2 above) or a shared mailbox
    - For a shared mailbox, enter the mailbox address
- 4 (Optional) Test your email account settings



## Step #2 - Define the actions

Upon adding the **GET EMAIL MESSAGES** command to your wizard, you will notice that it becomes an empty "**container**" within the Editor Pane, waiting for you to fill it with instructions:



- 1 Enter the action(s) the wizard should take on each matching email message
  - You can do this by dragging the required Advanced Command(s) directly into the container



### NOTES

#### Loop-the-loop

Leo perform the actions defined within the container by **looping** through each retrieved message (i.e., it will perform the complete sequence of actions on a single message, then move on to perform the sequence on each remaining message in turn).

#### No limits

You can use any available Advanced Command within the **GET EMAIL MESSAGES** container (i.e., don't feel limited to using just the Email commands!)

A combination of these two notes leads us to a...



### TIP

#### Break that loop!

Under certain conditions, you may want to break and/or restart the loop created by the **GET EMAIL MESSAGES** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

## Email: Get Data

Obtain selected information about a message retrieved via the **GET EMAIL MESSAGES** command and place it into a variable.



### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.

## Using the EMAIL: GET DATA command

1 Select the type of information to retrieve:

- From
- To
- Subject
- Body (in plain text format)
- Body (in HTML format)
- Date
- Message key
- Attachment name(s)

2 Enter the name of the variable into which you'd like to place the result

3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Email: Move to Folder

Move a message retrieved via the **GET EMAIL MESSAGES** command to a specified email folder.



### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.



### CAUTION

Once you move a retrieved message to a different folder, additional email commands will no longer function for this message.

## Using the EMAIL: MOVE TO FOLDER command

**Email: Move to folder** [Close]

**Note:** Once a message is moved to another folder, other email commands pertaining to this message will no longer be functional.

Folder path: (provide full path, e.g. *Inbox/Promotions*)

  
 Create folder if doesn't exist (folder names are case sensitive)

▼ Error handling ⓘ

OK Cancel

- 1 Enter the full folder path to which you'd like to move the email (syntax: `folder/subfolder/sub-subfolder/etc.`); **and** indicate whether Leo should create the specified folder if it doesn't exist
- 2 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Email: Forward

Forward a message retrieved via the **GET EMAIL MESSAGES** command.





### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.

## Using the EMAIL: FORWARD command

- 1 Enter recipient email addresses
  - Separate multiple email addresses with commas
- 2 Enter the address from which the email will be sent
  - This is the address that will appear to the recipient as the sender of the email

-  3 Enter any text you wish to add to original message
  - This text will appear prior to text of the forwarded message
-  4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Email: Reply

Reply to a message retrieved via the **GET EMAIL MESSAGES** command.



### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.

## Using the EMAIL: REPLY command

- 1 Enter any text you wish to add to original message
  - This text will appear prior to text of the original message
- 2 Indicate whether the reply should be sent to all recipients of the original message
  - If left unchecked, the reply will be sent only to the sender of the original message
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Email: Delete

Delete message(s) retrieved via the [GET EMAIL MESSAGES](#) command.



### NOTE

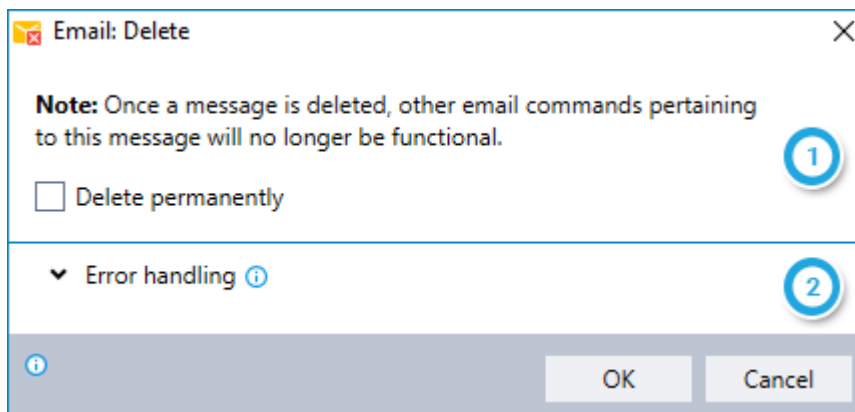
This command can be used only within a [GET EMAIL MESSAGES](#) container.



### CAUTION

Once you delete a retrieved message (either permanently or by moving it to the **Deleted Items** folder), additional email commands will no longer function for this message.

## Using the EMAIL: DELETE command



This command does not **require** that you configure any options and can be added to a wizard simply by dragging it into the [GET EMAIL MESSAGES](#) container in the Editor Pane. However, you can configure some optional settings:

- 1 Indicate whether the message should be deleted permanently
  - If left unchecked, the message will simply be moved to the receiving account's **Deleted Items** folder
- 2 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Email: Mark as Read/Unread

Mark a message retrieved via the **GET EMAIL MESSAGES** command as read or unread.



### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.

## Using the EMAIL: MARK AS READ/UNREAD command

- 1 Choose whether you would like the email to be marked as read or unread
- 2 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Email: Save Attachments

Save the attachments of a message retrieved via the **GET EMAIL MESSAGES** command and place the saved file names into a new or existing variable.



### NOTE

This command can be used only within a **GET EMAIL MESSAGES** container.

## Using the EMAIL: SAVE ATTACHMENTS command

- 1 Enter the **full path** of the folder to which you would like to save the attachments; **and** indicate whether the saved files should replace (overwrite) existing file(s) with the same filename(s)
- 2 (Optional) Enter a filter if you wish to save only attachments matching a certain filename pattern
  - Use an asterisk as a wildcard for one or more characters within the filename, for example:
    - The filter `*.docx` will save only attachments with a `*.docx` extension (such as `invoice1.docx` and `premium notice.docx`)
    - The filter `*invoice.*` will save only attachments with the word `invoice` in the filename (such as `december 2017 invoice.xlsx` and `invoice122017.pdf`)

- Enter multiple comma-separated filters to save attachments matching one or more of them (i.e., attachments matching **any** of the specified filters will be saved)

3

Enter the name of the variable into which you'd like to place the name(s) of the saved files

- Multiple filenames will be returned in the variable separated by commas

4

Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 14: External Program Commands

In this chapter:

Run Program .....	221
Run .NET Plugin Method .....	223
Call Web Service Method .....	224
Run Script .....	225
Run curl Command .....	227
Get IE Page HTML .....	228
Run JavaScript on Page .....	229

## Run Program

Launch an application directly from its executable (.exe) file (including command line parameters as required).



### TIP

#### What are command line parameters?

Make this command truly powerful by using command line parameters.

What are they? Codes (called "arguments" or "switches") that tell a program how to behave when it starts up. Some examples:

- Tell a browser what website to open
- Tell an application what file to open
- Tell a program window whether to open maximized or minimized

Available command line parameters vary by application and are usually documented by the program's developer. For example, click [here](#) to see what Microsoft has to say about the [command line parameters for Excel](#).

## Using the RUN PROGRAM command

The screenshot shows the 'Run program' dialog box with the following fields and options:

- Program:** A text input field with a 'Browse...' button and an information icon. Callout 1.
- Command line parameters:** A text input field with an information icon. Callout 2.
- Return result in variable:** A dropdown menu with the text 'Type a variable name'. Callout 3.
- Hide program window
- Wait for the program to finish. Callout 4.
- Return program window handle in variable:  
Type a variable name (Max. wait [ ] ms.)

At the bottom, there are 'OK' and 'Cancel' buttons and an information icon.

- 1 Select the .exe file to launch
- 2 Specify any **command line parameters**
- 3 Enter the name of the variable into which you'd like to place the returned result  
**Note:** This field is relevant only for an executable program that returns a result
- 4 Indicate additional options for running the program:
  - Whether to hide the window
  - Whether to wait for the program to finish before the wizard moves on
  - Whether to return the program's **window handle** in a variable



### TIP

#### Internet Explorer at your command

No need to look for Internet Explorer's .exe file. In step 1 above, just type `iexplore`. And if you want it to start with a website already open, just type the site's URL in step 2.

## Run .NET Plugin Method

Run a plugin that was developed specifically to extend Leo's capabilities. For additional information see the *Leo Plugin Development* section of the Leo Studio User Guide.

**Run .net plugin method**

Embedded plugin:

Local plugin:

Runtime folder:

Method to Run:

Parameters:

Name	Type	Direction	Value

Return result in variable:

▼ Error handling

## Call Web Service Method

Retrieve data from a web service and place it into a new or existing variable.



### NOTE

#### What is a web service?

A web service allows an **application** to talk to a web page, instead of using a browser to open it. The application is able to either retrieve information from or submit information to some resource. Some examples:

- Financial websites providing a method for your system to retrieve stock quotes and currency exchange rates
- Shipping companies providing a method for your shipping application to request quotes and tracking information

Call web service method
✕

Enter values manually or use the Service Discovery tool Discover Services...

Web Service URL:

Call Method
  Send SOAP xml

Protocol:

Service:

Method:

Enable user authorization

Parameters:

	Name	Type	Value
+			Enter new parameter name

Return result in variable:

Service timeout:  seconds

▼ Error handling ?

OK
Cancel



## Run Script

Instruct Leo to execute a script (including parameters as required). Scripts can be written in any of the following languages:

- VBScript
- JScript
- Perl
- PScript
- Python

### Using the RUN SCRIPT command

**Run script**

Script Code: ⓘ

Load from file...

Parameters:  ⓘ 1

Language:  2

Timeout:  seconds 3

Return result in variable 4

5

Use WScript.StdOut.Write("My return value") to return value from script

**Information:**  
Script parameters are space separated

ⓘ

- 1 Enter the script code or load it from a file
- 2 Specify any parameters for running the script (separated by spaces)
- 3 Select the language in which the script is written
- 4 Specify a timeout (i.e., if the script does not begin to run within this time frame, Leo will move on)
- 5 Enter the name of the variable into which you'd like to place the returned result
  - Languages requiring specific code to return a value:
    - **VBScript:** `WScript.StdOut.Write("My return value")`
    - **JScript:** `WSH.StdOut.WriteLine("My return value")`
    - **Python:** `print([returnvalue])`
  - **Note:** This field is relevant only for a script that is designed to return a result

## Run curl Command

Execute a curl command to transfer data from or to a server.

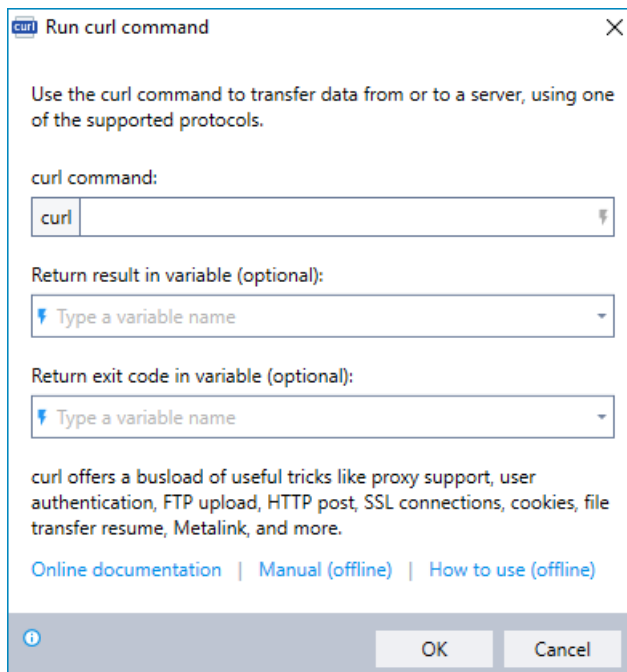


### NOTE

#### What is curl?

curl is a tool used to transfer data from or to a server without user interaction, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP).

You can use curl to download or upload physical web pages, images, documents and files.



## Get IE Page HTML

Retrieve the HTML code of the web page currently active in Internet Explorer and place it into a new or existing variable.



### NOTE

**Make sure it's Internet Explorer!**

This command supports only the Internet Explorer browser, so make sure that's the one that's open when the wizard is run.

## Using the GET IE PAGE HTML command

- 1 Enter the name of the variable into which you'd like to place the HTML code
- 2 Indicate whether to append HTML for embedded `<iframe>` elements on the page. If yes:
  - Specify how many embedded levels to include
  - Enter the delimiter to use to separate the code for each `<iframe>`
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Run JavaScript on Page

Instruct Leo to execute JavaScript on a web page.



### NOTE

**Make sure it's the right browser!**

This command supports only Chrome and Internet Explorer, so make sure it's one of those that's open when the wizard is run.

## Using the RUN JAVASCRIPT ON PAGE command

The screenshot shows a dialog box titled "Run javascript on page" with a close button (X) in the top right corner. The dialog is divided into three main sections:

- Section 1:** A text area labeled "Javascript Code:" with a search icon (magnifying glass) in the top right corner. Below the text area is a "Load from file..." link. A blue circle with the number "1" is positioned to the right of the text area.
- Section 2:** A greyed-out text box containing the instruction: "To return a result from the javascript code, explicitly call `LeoSetOutputVariableValue()`". Below this is a "+Add function call" link. A blue circle with the number "2" is positioned to the right of the text box.
- Section 3:** A section labeled "Set the result in variable:" with a dropdown menu below it. The dropdown menu contains the text "Type a variable name" and a search icon (magnifying glass) on the left. A blue circle with the number "3" is positioned to the right of the dropdown menu.

At the bottom of the dialog, there is an information icon (i) on the left and "OK" and "Cancel" buttons on the right.

- 1 Enter the script code or load it from a file
- 2 To return a result, explicitly call the `LeoSetOutputVariableValue` function with the output value
  - Click [+Add function call](#) to quickly add it to your code
- 3 Enter the name of the variable into which you'd like to place the returned result



#### NOTE

Steps [2](#) and [3](#) are relevant only for a script that is designed to return a result.

# CHAPTER 15: Database Commands

In this chapter:

Execute SQL Query .....	232
Monitor Database Changes .....	234

## Execute SQL Query

Perform an SQL query against a specified database and place the results into a variable. The query may be either predefined in Leo Admin or defined within the Advanced Command itself.

### Using the EXECUTE SQL QUERY command

#### Predefined query

The screenshot shows the 'Execute SQL query' dialog box with the following details:

- Query name:** Select users
- Query content:**

```
SELECT TOP $_num_num_$ [UserID]
,[UserName]
,[Password]

FROM [Leo].[dbo].[LeoUsers]
```
- Parameters:**

Name	Value	Type
_num_num_		Automatic
- Return result in variable:** Type a variable name
- Query output:** All rows
- Row delimiter:** Line break
- Column delimiter:** Comma

- 1 Select the query you would like to perform from the available predefined queries
- 2 **Query Content** and **Parameters** will be populated based upon the query you have selected
- 3 Specify **Value** and **Type** for defined parameters as required
- 4 Enter the name of the variable into which you'd like to place the results
- 5 Indicate whether you would like Leo to return all rows of the result or the first row only; *and*  
Enter the delimiters to use to separate each row and column in the returned data



## Custom query

- 1 Define the connection string for the data source
  - (Optional) Click the **BUILDER** link to assist in defining the connection string
- 2 Indicate whether you would like to retrieve database login credentials from the Leo Credentials Vault
- 3 Enter your query using standard SQL syntax
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).
- 5 Enter the name of the variable into which you'd like to place the results
- 6 Indicate whether you would like Leo to return all rows of the result or the first row only; *and* Enter the delimiters to use to separate each row and column in the returned data

## Monitor Database Changes

Monitor a specified database table for changes (insertions, updates, deletions) and place changed data in variables.

### Using the MONITOR DATABASE CHANGES command

- 1 Define the connection string for the data source; and
  - (Optional) Click the **BUILDER** link to assist in defining the connection string
- 2 Indicate whether you would like to retrieve database login credentials from the Leo Credentials Vault
- 3 Select the table to monitor; *and*
  - Indicate the types of changes to monitor (insert/update/delete)
- 4 Identify the following columns:
  - UID – Unique ID column
  - Update Date – The DATETIME column containing update time of each record
  - Is Deleted – Numeric (BOOL) column representing a deleted state for each record
- 5 (Optional) Enter any applicable WHERE clauses to further refine the changes to monitor

- 6** Set the duration of polling cycle in seconds; *and*  
Indicate if you would like Leo to stop monitoring (i.e., timeout) after a certain number of minutes
- 7** Enter the columns to be returned into the variables, separated by commas; *and*  
Indicate the number of rows to fetch, or select to fetch the first changed row only
- 8** Enter the names of the variables into which you'd like to place the results; *and*  
Enter the delimiters to use to separate each row and column in the returned data
- 9** Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 16: Credentials Vault Commands

In this chapter:

Insert Username Into Active Field .....	237
Insert Password Into Active Field .....	238
Generate New Password .....	239
Revert Password .....	240

## Insert Username Into Active Field

Place a username from the Leo Credentials Vault into the active field.



### TIP

Don't forget to make sure the field you need is selected before using this command!

Either <TAB> over to it or click inside it, then you'll be set to go.

## Using the INSERT USERNAME INTO ACTIVE FIELD command

- 1 Select the application for which you'd like to enter a username (the "target" application)
- 2 Indicate which username Leo should retrieve and enter:
  - From application credentials (i.e., the username configured for the target application on the current machine)
  - For a specific user identified by the value stored in a variable; *or*
  - For a specific user selected from a list
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Insert Password Into Active Field

Place a password from the Leo Credentials Vault into the active field.



### TIP

Don't forget to make sure the field you need is selected before using this command!

Either <TAB> over to it or click inside it, then you'll be set to go.

## Using the INSERT PASSWORD INTO ACTIVE FIELD command

- 1 Select the application for which you'd like to enter a password (the "target" application)
- 2 Indicate which password Leo should retrieve and enter:
  - From application credentials (i.e., the password configured for the target application on the current machine)
  - For a specific user identified by the value stored in a variable; *or*
  - For a specific user selected from a list
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Generate New Password

Create a new password in the Leo Credentials Vault, either for a specified application or for a general user (email servers, database, etc.)

### Using the GENERATE NEW PASSWORD command

- 1 Indicate whether you'd like to create a password for a specific application or for a general user (email servers, database, etc.)
  - 1a If for a specific application, select the application (the "target" application)
  - 1b If for a general user, select the user
- 2 If creating a password for a specific application, indicate which password:
  - For application user's credentials (i.e., the password configured for the target application on the current machine)
  - For a specific user selected from a list; **or**
  - For a specific user identified by the value stored in a variable
- 3 Enter the password requirements as defined by the target application or system administrator
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Revert Password

Overwrite the current password in the Leo Credentials Vault with the last saved password, either for a specified application or for a general user (email servers, database, etc.)



### CAUTION

Once performed, this action cannot be reversed.

## Using the REVERT PASSWORD command

**1** Indicate whether you'd like to revert the password for a specific application or for a general user (email servers, database, etc.)

**1a** If for a specific application, select the application (the "target" application)

**1b** If for a general user, select the user



- 2 If reverting the password for a specific application, indicate which password:
  - For application user's credentials (i.e., the password configured for the target application on the current machine)
  - For a specific user selected from a list; **or**
  - For a specific user identified by the value stored in a variable
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 17: Robotic Process Automation Commands

In this chapter:

Add Automation Task to Queue .....	243
Get Automation Task Status .....	246
Get File Trigger Input .....	248
Get Folder Trigger Input .....	249
Get Email Trigger Input .....	250
Get Database Trigger Input .....	251

## Add Automation Task to Queue

Initiate and add a task to the robot task queue.



### NOTES

- This command can be used in both sensors and wizards and run by either attended or unattended robots (i.e., both humans and robots can add tasks to the robot queue).
- The sensor/wizard in which this command is executed continues after the task has been added to the queue. It does **NOT** wait for a robot to complete the assigned task.

## Using the ADD AUTOMATION TASK TO QUEUE command

**Add automation task to queue**

Initiates and adds an automation task to the robots' task queue. The task will be assigned when a robot becomes available. The current wizard/sensor will continue and will not wait for the robot to complete the task.

Task name:

Queue Priority:

**Wizard** Parameters Robot

Select a wizard:

By wizard ID:

By custom ID:

Return initiated task ID in variable:

Add this task to the user's robot task queue viewer  
If initiated from a user's desktop (Leo Player), allow the user to track task status, receive a notification when the task ends and continue the process if necessary.

When task ends, place task's output in a variable (optional):

If ended **successfully**:

If ended **unsuccessfully**:

If **failed**:

Error handling

OK Cancel

1 Enter a task name and set queue priority

2 Enter additional information in 3 tabs:

<p>Wizard Parameters Robot</p> <p><input checked="" type="radio"/> Select a wizard: <input type="text"/> ...</p> <p><input type="radio"/> By wizard ID: <input type="text"/> ⓘ ...</p> <p><input type="radio"/> By custom ID: <input type="text"/> ⓘ</p>	<p><b>Wizard:</b></p> <p>Select the wizard to assign to a robot (the "task wizard"):</p> <ul style="list-style-type: none"> <li>• From the Wizard Catalog</li> <li>• By the wizard ID automatically assigned to the task wizard; <b>or</b></li> <li>• By the custom ID created for the task wizard</li> </ul>				
<p>Wizard Parameters Robot</p> <p>Parameters: <span style="float: right;">+ Add</span></p> <table border="1" style="width: 100%;"> <thead> <tr> <th style="width: 50%;">Variable</th> <th style="width: 50%;">Value</th> </tr> </thead> <tbody> <tr> <td style="height: 100px;"> </td> <td> </td> </tr> </tbody> </table>	Variable	Value			<p><b>Parameters:</b></p> <p>(Optional) Enter parameters (i.e., the initial values of some or all of the task wizard's variables)</p>
Variable	Value				
<p>Wizard Parameters Robot</p> <p><input checked="" type="radio"/> First available robot</p> <p><input type="radio"/> First available robot from the group: <input type="text"/></p> <p><input type="radio"/> A specific robot: <input type="text"/></p>	<p><b>Robot:</b></p> <p>Choose to which robot the task should be assigned:</p> <ul style="list-style-type: none"> <li>• The first available robot</li> <li>• The first available robot from a specific group (select from list); <b>or</b></li> <li>• A specific robot (select from list)</li> </ul>				

3 Enter the name of the variable into which you'd like to place the task ID

- The task ID is created immediately and will be available after this Advanced Command is executed (i.e., it does not require that the task be started or completed by a robot). The task ID can then be used to track the task's status with the **GET AUTOMATION TASK STATUS** command.

#### 4 Hybrid Mode Feature

Indicate if you'd like the task to be added to the (human) end user's robot task queue viewer

- Requires that this Advanced Command be initiated from the user's desktop (via **Leo Attended Robot**, formerly **Leo Player**)
- Allows the user to track task status, receive a notification when the task is complete, and continue the hybrid process if required

#### 5 (Optional) Enter the name of the variable into which you'd like to place the task's output

- Requires that the task wizard includes the `Report wizard output` command. [Learn more.](#)
- This variable cannot be used in messages that display while the current sensor/wizard is running (or other Advanced Commands in the current wizard), but it can be used in messages that display once the task has completed (see step [6](#) below).

#### 6 Select further actions to execute once the task has completed (if necessary):

- No further action
- Display a custom message
- Run the current wizard from a specific step; *or*
- Run another wizard

#### 7 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get Automation Task Status



Retrieve the status of an automation task assigned to the robot queue and place it into a new or existing variable.

### Using the GET AUTOMATION TASK STATUS command

The screenshot shows a dialog box titled "Get automation task status". It has a close button in the top right corner. The dialog is divided into four sections, each with a numbered callout (1-4):  
1. "Get the status of the task (enter task ID):" followed by a text input field and an information icon.  
2. "Return status in variable:" followed by a dropdown menu with the text "Type a variable name".  
3. "Return executing robot (machine name) in variable:" followed by a dropdown menu with the text "Type a variable name".  
4. "Error handling" with a dropdown arrow.  
At the bottom of the dialog are "OK" and "Cancel" buttons, and an information icon on the left.

- 1 Enter the task ID of the task for which you want to retrieve the status
  - The task ID can be set into a variable when the task is added to the queue using the **ADD AUTOMATION TASK TO QUEUE** command
- 2 Enter the name of the variable into which to place the status. The statuses returned are numeric codes, as follows:

Code	Status
0	started
1	stopped
2	ended
3	delayed
4	inactive
5	skipped
6	queued
7	faulty

-  3 Enter the name of the variable into which to place the name of the robot executing (or that executed) the task
  - This data becomes available at the time the robot starts the task
-  4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get File Trigger Input

Retrieve information about the file that triggered the wizard and place it into new or existing variables.



### NOTES

- This command is relevant only for a wizard initiated by a file trigger (as configured in Leo Console)
- The information retrieved is generally used in conjunction with other Advanced Commands (such as [File Commands](#) and [Excel Commands](#))

### Using the GET FILE TRIGGER INPUT command

- 1 Indicate if you wish to retrieve the path of the file that triggered the wizard and enter the name of the variable into which to place it
- 2 Indicate if you wish to retrieve the action executed on the file that triggered the wizard and enter the name of the variable into which to place it. The possible statuses to be returned are **NEW**, **MODIFIED**, and **DELETED**.



## Get Folder Trigger Input

Retrieve information about the folder that triggered the wizard and place it into new or existing variables.



### NOTES

- This command is relevant only for a wizard initiated by a folder trigger (as configured in Leo Console)
- The information retrieved is generally used in conjunction with other Advanced Commands (such as [Folder Commands](#), [File Commands](#), and [Excel Commands](#))

### Using the GET FOLDER TRIGGER INPUT command

Get folder trigger input

If this wizard is initiated by a folder trigger, retrieve the folder information into variables:

Folder path

Type a variable name

Use file/folder commands to explore a folder by its path.

Folder action

Type a variable name

Expected values: NEW / DELETED

OK Cancel

- 1 Indicate if you wish to retrieve the path of the folder that triggered the wizard and enter the name of the variable into which to place it
- 2 Indicate if you wish to retrieve the action executed on the folder that triggered the wizard and enter the name of the variable into which to place it. The possible statuses to be returned are **NEW** and **DELETED**.

## Get Email Trigger Input

Retrieve the message key of the email that triggered the wizard and place it into a new or existing variable.



### NOTES

- This command is relevant only for a wizard initiated by an email trigger (as configured in Leo Console)
- The message key retrieved is used in conjunction with the **GET EMAIL MESSAGES** command

## Using the GET EMAIL TRIGGER INPUT command



Indicate if you wish to retrieve the message key of the email that triggered the wizard and enter the name of the variable into which to place it

## Get Database Trigger Input

Retrieve information about the database changes that triggered the wizard and place it into new or existing variables.



### NOTES

- This command is relevant only for a wizard initiated by a database trigger (as configured in Leo Console)
- The information retrieved is generally used in conjunction with other Advanced Commands (such as [Database Commands](#), [Excel Commands](#), and [File Commands](#))

## Using the GET DATABASE TRIGGER INPUT command

**Get database trigger input**

If this wizard is initiated by a database trigger, retrieve the information into variables:

Changed data  
Type a variable name **1**



Row delimiter  
Type a variable name **2**

Column delimiter  
Type a variable name **3**

Database action  
Type a variable name **4**  
Expected values: INSERT / UPDATE / DELETE.

OK Cancel

- 1** Indicate if you wish to retrieve the data that triggered the wizard (columns and rows as configured in the trigger) and enter the name of the variable into which to place it
- 2** Indicate if you wish to retrieve the delimiter used to separate each row in the changed data and enter the name of the variable into which to place it

-  3 Indicate if you wish to retrieve the delimiter used to separate each column in the changed data and enter the name of the variable into which to place it
-  4 Indicate if you wish to retrieve the action executed on the database that triggered the wizard and enter the name of the variable into which to place it. The possible statuses to be returned are **INSERT**, **UPDATE**, and **DELETE**.

# CHAPTER 18: Excel Commands

In this chapter:

Get Excel Range Values .....	254
Set Excel Range Values .....	256
Excel Worksheet Actions .....	259
Excel Row Actions .....	260
Excel Column Actions .....	262
Convert Excel to CSV .....	264
Create New Excel File .....	265
Query From Excel .....	266
Run Macro .....	268

## Get Excel Range Values

Retrieve specified cell values from an Excel file and place them into a new or existing variable.

### Using the GET EXCEL RANGE VALUES command

The screenshot shows a dialog box titled "Get Excel range values" with a close button (X) in the top right corner. The dialog is divided into several sections:

- Section 1:** "Get the values from Excel file:" with a text input field containing an information icon (i) and a "Browse..." button.
- Section 2:** "Worksheet name:" with a text input field containing an information icon (i).
- Section 3:** Radio button selected: "Range specified by cell index or letter". It contains four input fields: "Row:", "Column:", "- Row:", and "Column:", each with an information icon (i). Below them is a tip: "Tip: Type '0' to automatically identify non-empty cell values".
- Section 4:** Radio button unselected: "Range specified by cell names". It contains a text input field with an information icon (i) and the text "Example: A2:D10".
- Section 5:** "Return values in variable:" with a dropdown menu showing "Type a variable name".
- Section 6:** "Return values as:" with a dropdown menu showing "Actual Values".
- Section 7:** "Column delimiter:" with a text input field containing an information icon (i).
- Section 8:** "Row delimiter:" with a text input field containing an information icon (i).
- Section 9:** "Error handling" with a dropdown arrow and an information icon (i).

At the bottom of the dialog are "OK" and "Cancel" buttons, and a small information icon (i) on the left.

1 Select the Excel file from which you would like to retrieve cell values

2 Enter the worksheet (tab) in which the values are located

### 3 Specify the range in which the values are located

#### RANGE SPECIFIED BY CELL INDEX OR LETTER

- Designate the row and column numbers by numbers or letters (e.g., cell **C5** could be designated as `Row 5, Column 3` or `Row 5, Column C`)
- Use a 0 to specify all non-empty cells
  - To indicate all non-empty rows from the start of the range, place a 0 in the **ROW** field for the end of the range
  - To indicate all non-empty columns from the start of the range, place a 0 in the **COLUMN** field for the end of the range
  - Once Leo reads 20 consecutive empty cells, it will assume that all remaining rows or columns are empty

#### RANGE SPECIFIED BY CELL NAMES

- Designate the range using "standard" Excel format (e.g., the range from **Column B Row 3** to **Column F Row 5** is designated as `B3 : F5`)

### 4 Specify options for returning the data:

- Enter the name of the variable into which you'd like to place the retrieved values
- Select whether you'd like to return the values as **ACTUAL VALUES** or **FORMATTED TEXT**
- Enter the delimiters to use to separate each column and row in the returned data

### 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



#### TIP

##### When should I use **FORMATTED TEXT** option for returning values?

It's important to use the **FORMATTED TEXT** option when the relevant cells are formatted as dates. Otherwise, the values returned will be Excel's serial numbers for the dates. (For example `June 21, 2017 = serial number 42907`.)

**But be careful!** If your column is too narrow to display the full cell contents and you see `####`, Leo will return `####`. (All you've got to do to fix this is make the column wider.)

For reading all other cells, use the **ACTUAL VALUES** option.

## Set Excel Range Values

Write the contents of a variable into an existing Excel file.



### NOTE

**Close your file first!**

The Excel file into which you write values must be closed at the time this command is run.

## Using the SET EXCEL RANGE VALUES command

The screenshot shows the 'Set Excel range values' dialog box with the following elements and callouts:

- 1:** Points to the 'New values:' section, which includes radio buttons for 'Values from array', 'Formulas from array', 'Same value', 'Same formula', and 'Empty'. It also includes a sub-section for 'Array stored in variable:' with a dropdown menu (placeholder: 'Type a variable name'), 'Array column delimiter:', and 'Array row delimiter:' fields.
- 2:** Points to the 'Browse...' button next to the 'Set values in Excel file:' text box.
- 3:** Points to the 'Worksheet name: (will be created if doesn't exist)' text box.
- 4:** Points to the 'Range specified by cell index or letter' section, specifically the 'Row:' and 'Column:' input fields.
- 5:** Points to the 'Error handling' dropdown menu.

At the bottom of the dialog are 'OK' and 'Cancel' buttons.



**1** Specify the data to be written:

<p>New values:</p> <ul style="list-style-type: none"> <li><input checked="" type="radio"/> Values from array</li> <li><input type="radio"/> Formulas from array</li> <li><input type="radio"/> Same value</li> <li><input type="radio"/> Same formula</li> <li><input type="radio"/> Empty</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Array stored in variable:  <input type="text" value="Type a variable name"/></p> <p>Array column delimiter: <input type="text"/> ⓘ</p> <p>Array row delimiter: <input type="text"/> ⓘ</p> </div> <p><b>Note:</b> If the array length exceeds the cell range size, any existing data will be overwritten.</p>	<p><b>Values from array:</b></p> <ul style="list-style-type: none"> <li>• Enter the name of the variable containing the array that will be written to the specified range</li> <li>• Enter the delimiter that separates each column in the array</li> <li>• Enter the delimiter that separates each row in the array</li> </ul>
<p>New values:</p> <ul style="list-style-type: none"> <li><input type="radio"/> Values from array</li> <li><input checked="" type="radio"/> Formulas from array</li> <li><input type="radio"/> Same value</li> <li><input type="radio"/> Same formula</li> <li><input type="radio"/> Empty</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Array stored in variable:  <input type="text" value="Type a variable name"/></p> <p>Array column delimiter: <input type="text"/> ⓘ</p> <p>Array row delimiter: <input type="text"/> ⓘ</p> </div> <p><b>Note:</b> If the array length exceeds the cell range size, any existing data will be overwritten.</p>	<p><b>Formulas from array:</b></p> <ul style="list-style-type: none"> <li>• Enter the name of the variable containing the array that will be written to the specified range</li> <li>• Enter the delimiter that separates each column in the array</li> <li>• Enter the delimiter that separates each row in the array</li> </ul>
<p>New values:</p> <ul style="list-style-type: none"> <li><input type="radio"/> Values from array</li> <li><input type="radio"/> Formulas from array</li> <li><input checked="" type="radio"/> Same value</li> <li><input type="radio"/> Same formula</li> <li><input type="radio"/> Empty</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Value:  <input type="text"/></p> </div>	<p><b>Same value:</b></p> <ul style="list-style-type: none"> <li>• Enter the single value that will be written to each cell in the specified range (can be entered manually or copied from values stored in variables)</li> </ul>

<p>New values:</p> <ul style="list-style-type: none"> <li><input type="radio"/> Values from array</li> <li><input type="radio"/> Formulas from array</li> <li><input type="radio"/> Same value</li> <li><input checked="" type="radio"/> <b>Same formula</b></li> <li><input type="radio"/> Empty</li> </ul> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Formula:</p> <input style="width: 100%; height: 20px;" type="text"/> </div>	<p><b>Same formula:</b></p> <ul style="list-style-type: none"> <li>Enter the single formula that will be written to each cell in the specified range (can be entered manually or copied from values stored in variables)</li> </ul>
<p>New values:</p> <ul style="list-style-type: none"> <li><input type="radio"/> Values from array</li> <li><input type="radio"/> Formulas from array</li> <li><input type="radio"/> Same value</li> <li><input type="radio"/> Same formula</li> <li><input checked="" type="radio"/> <b>Empty</b></li> </ul> <div style="border: 1px solid #ccc; width: 100%; height: 100%; margin-top: 10px;"></div>	<p><b>Empty:</b></p> <ul style="list-style-type: none"> <li>Leo will clear all cell values in the specified range</li> </ul>

- 2 Select the Excel file into which to place the data
- 3 Enter the worksheet (tab) into which to place the data
  - If a worksheet with this name doesn't exist, Leo will create it
- 4 Specify the range into which to place the data. Note that any existing data in the rows/columns specified will be overwritten.

**RANGE SPECIFIED BY CELL INDEX OR LETTER**

- Designate the row and column numbers by numbers or letters (e.g., cell **C5** could be designated as `Row 5, Column 3` or `Row 5, Column C`)

**RANGE SPECIFIED BY CELL NAMES**

- Designate the range using "standard" Excel format (e.g., the range from **Column B Row 3 to Column F Row 5** is designated as `B3 : F5`)

**Caution:** For arrays, Leo will use as many rows /columns as required to write all of the data the array contains. If the array is longer than the range specified, existing data **outside the range** will be overwritten.

- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Excel Worksheet Actions

- Retrieve information about the worksheets in an Excel file and place it into a new or existing variable; *or*
- Perform basic worksheet actions (rename, move, delete, etc.)

### Using the EXCEL WORKSHEET ACTIONS command

- 1 Select the Excel file on which you would like to perform a worksheet action
- 2 Select the worksheet action you would like to perform:
  - **Get worksheet name:** Retrieve the name of the worksheet at a specified position
  - **Get worksheet position:** Retrieve the position of the worksheet with a specified name
  - **Get worksheet count:** Retrieve the total number of worksheets in the file
  - **Insert worksheet:** Insert a blank worksheet in the specified position
  - **Move worksheet to another position:** Move a worksheet from its currently specified position to a new position
  - **Duplicate worksheet:** Duplicate the worksheet at the specified position
  - **Rename worksheet:** Rename the worksheet at the specified position
  - **Delete worksheet:** Delete the worksheet at the specified position
- 3 Provide additional information as required (fields will vary by the worksheet action selected)
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Excel Row Actions

- Retrieve the number of non-empty rows within a specified column and place it into a new or existing variable; *or*
- Insert or delete rows

### Using the EXCEL ROW ACTIONS command

The screenshot shows the 'Excel row actions' dialog box with the following fields and controls:

- Select Excel file:** A text input field with an information icon and a 'Browse...' button.
- Worksheet name:** A radio button and a text input field with an information icon.
- Worksheet position:** A radio button and a text input field with the value '1' and an information icon.
- Action:** A dropdown menu with 'Get row count' selected.
- Column index or letter:** A text input field with the value '1' and an information icon.
- Number of empty cells allowed:** A text input field with the value '1' and an information icon. Below it is the text: 'Set the maximum number of empty cells before Leo stops counting rows. The last non-empty cell will define the row count.'
- Return result in variable:** A dropdown menu with 'Type a variable name' selected.
- Error handling:** A dropdown arrow and an information icon.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

1 Select the Excel file on which you would like to perform a row action

2 Enter the relevant worksheet within the file (identified either by name or position)

- 3 Select the row action you would like to perform:
  - **Get row count:** Retrieve the number of non-empty rows within a specified column
    - Option to specify the number of empty cells before Leo stops counting and assumes all remaining rows are empty
  - **Insert rows:** Insert the specified number of rows at the specified position
  - **Delete rows:** Delete the specified number of rows at the specified position
- 4 Provide additional information as required (fields will vary by the action selected)
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Excel Column Actions

- Retrieve the number of non-empty columns within a specified row and place it into a new or existing variable; *or*
- Insert or delete columns

### Using the EXCEL COLUMN ACTIONS command

The screenshot shows the 'Excel column actions' dialog box. It has a title bar with a close button. The main area is divided into several sections:

- Select Excel file:** A text box followed by a 'Browse...' button.
- Worksheet name:** A radio button and a text box.
- Worksheet position:** A radio button and a text box.
- Action:** A dropdown menu.
- Row index:** A text box.
- Number of empty cells allowed:** A text box.
- Return result in variable:** A dropdown menu.
- Error handling:** A dropdown arrow.

At the bottom, there are 'OK' and 'Cancel' buttons. Numbered callouts 1 through 5 point to the 'Browse...' button, the 'Worksheet name' and 'Worksheet position' text boxes, the 'Action:' dropdown, the 'Row index:' text box, and the 'Error handling' dropdown respectively.

- 1 Select the Excel file on which you would like to perform a column action
- 2 Enter the relevant worksheet within the file (identified either by name or position)

- 3 Select the column action you would like to perform:
  - **Get column count:** Retrieve the number of non-empty columns within a specified row
    - Option to specify the number of empty cells before Leo stops counting and assumes all remaining columns are empty
  - **Insert columns:** Insert the specified number of columns at the specified position
  - **Delete columns:** Delete the specified number of columns at the specified position
- 4 Provide additional information as required (fields will vary by the action selected)
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Convert Excel to CSV

Convert a single worksheet from an Excel file to CSV format and save it in the specified location.

### Using the CONVERT EXCEL TO CSV command

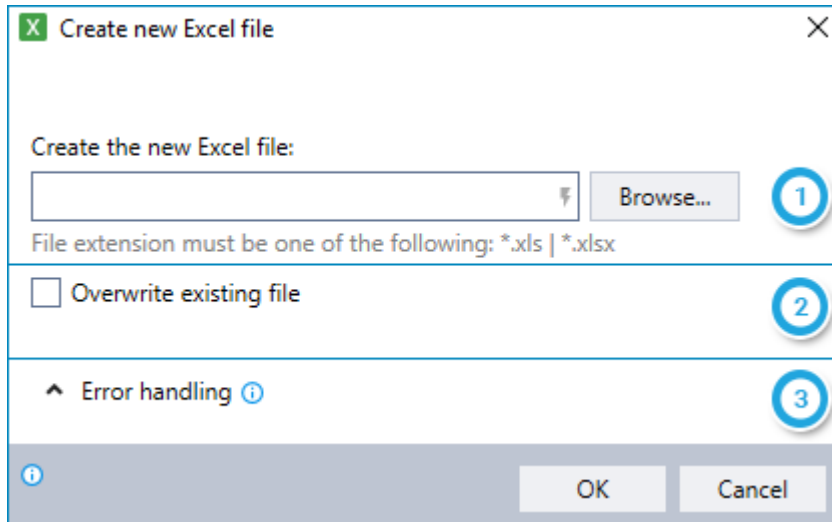
- 1 Select the Excel file that contains the worksheet you would like to convert
- 2 Identify the worksheet to convert (either by name or position)
- 3 Specify the name with which Leo should save the converted file, including the full file path
  - If the file does not exist, when the wizard is run, Leo will create one with the name you entered
  - If the file does exist, it will be overwritten by the new file
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Create New Excel File

Create a new Excel file in the location and with the file name you choose.

### Using the CREATE NEW EXCEL FILE command



- 1 Browse to the location in which you want Leo to create the Excel file; **and** Enter the desired file name (including one of the following file extensions: \*.xls, \*.xlsx)
- 2 Indicate whether the new file should overwrite an existing file of the same name
- 3 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Query From Excel

Query an Excel worksheet using SQL to return specific values from a range or entire sheet.

Command includes options to:

- Return selective data by using a SELECT clause
- Filter data by using a WHERE clause

### Using the QUERY FROM EXCEL command

- 1 Select the Excel file that you would like to query
- 2 Provide information about the table to query:
  - Enter the worksheet (tab) in which the values are located
  - Choose whether the table to query includes the entire worksheet or cells within a specific range
  - Indicate whether the table contains column headers
- 3 Define the parameters for your SQL query, including SELECT and/or WHERE clauses if relevant

- 4 Specify options for returning the data:
  - Enter the name of the variable into which you'd like to place the retrieved values
  - Enter the delimiters to use to separate each column and row in the returned data
- 5 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Run Macro

Run a VBA macro on the specified Excel file. The macro can be stored either in the Excel file itself or written directly into the Advanced Command dialog.



### NOTE

#### Close your file first!

The Excel file on which you run the macro must be closed at the time this command is run.

## Using the RUN MACRO command

The 'Run macro' dialog box contains the following elements:

- 1** Run Macro on file: A text input field with a 'Browse...' button.
- 2** Macro selection: Radio buttons for 'Macro is embedded in file' (selected) and 'Custom macro' (with an 'Edit' link).
- 3** Module name: A text input field.
- 4** Procedure name: A text input field.
- 5** Set the macro function returned result in the variable: A dropdown menu with the placeholder text 'Type a variable name'.
- 4** Save file when macro ends: A checked checkbox.
- 5** Error handling: A dropdown menu.
- Buttons: 'OK' and 'Cancel' at the bottom right.

- 1 Select the Excel file on which you would like to run a macro
- 2 Choose whether the macro is stored in the Excel file or if you will write it in directly the Advanced Command
  - If stored in the Excel file, indicate the module name and procedure name as shown in Excel's VBA Editor
  - If written in the Advanced Command, use the **EDIT** link to write/edit the macro
- 3 Enter the name of the variable into which you'd like to place the returned result of the macro

**Note:** This field is relevant only for a macro written as a function that returns a result
- 4 Indicate if you would like to save the file when the macro ends
- 5 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

# CHAPTER 19: SAP Commands

In this chapter:

Get SAP Object Text .....	271
Get SAP Object Value .....	272
Get SAP Object Location .....	273
Set SAP Object Value .....	275



## NOTE

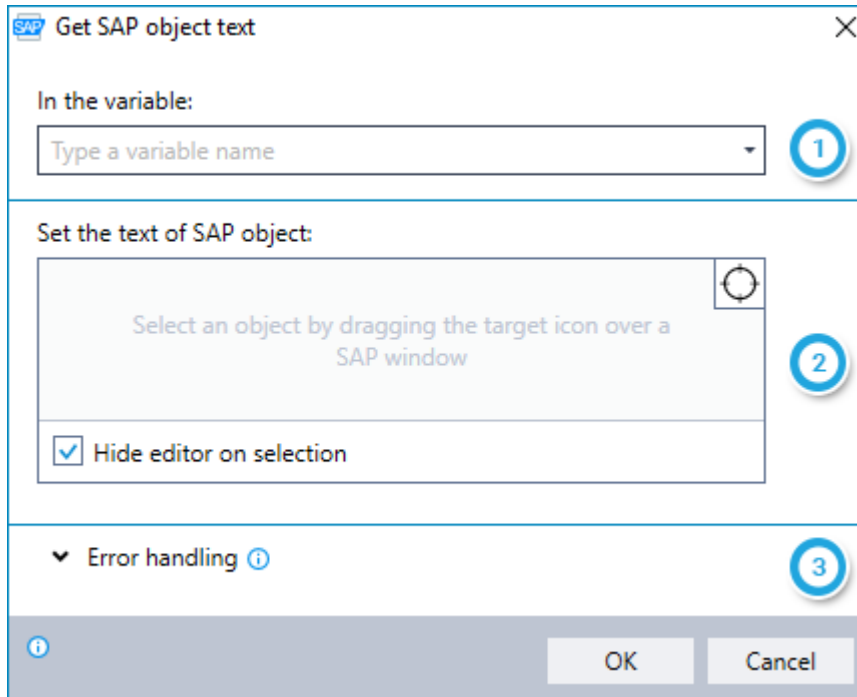
**Use in recorded steps only**





Use **SAP COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

## Get SAP Object Text

Retrieve the text of an object in the active SAP window and place it into a new or existing variable.

### Using the GET SAP OBJECT TEXT command

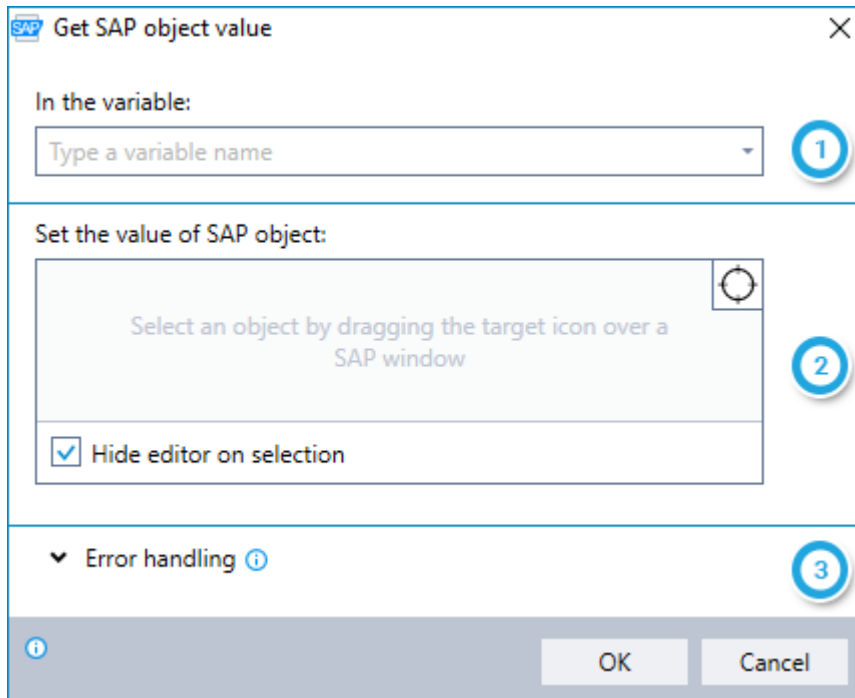




- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in an SAP window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test retrieving the text of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get SAP Object Value

Retrieve the value of an object in the active SAP window and place it into a new or existing variable. This command can be especially useful for checking the value of checkboxes, radio buttons, drop-down lists, etc.

### Using the GET SAP OBJECT VALUE command



- 1 Enter the name of the variable into which you would like to place the value of the selected object
- 2 Select the object whose value you would like to retrieve by dragging the  icon onto the object in an SAP window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - [Info](#) Display additional information about the selected object
    - [Test](#) Test retrieving the value of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Get SAP Object Location

Retrieve the location (in pixels) of an object in the active SAP window and place it into new or existing variables. Choose to retrieve the location represented either by:

- **Rectangle** – with variables for **left**, **top**, **width**, and **height**; *or*
- **Center point** – with variables for **X** and **Y** coordinates



### TIP

#### Choose rectangle or center point first



The variables you need to specify while setting up this command vary based on the method you choose for retrieving the location. So make this selection first and save yourself some time!

## Using the GET SAP OBJECT LOCATION command

The screenshot shows the 'Get SAP object location' dialog box with the following elements and callouts:

- Callout 1:** Points to the 'Set the **rectangle** of selected SAP object' dropdown menu.
- Callout 2:** Points to the four input fields for 'Left', 'Top', 'Width', and 'Height', each containing the placeholder text 'Type a variable name'.
- Callout 3:** Points to the central area containing the instruction 'Select an object by dragging the target icon over a SAP window' and a 'Hide editor on selection' checkbox.
- Callout 4:** Points to the 'Error handling' section, which is currently collapsed.

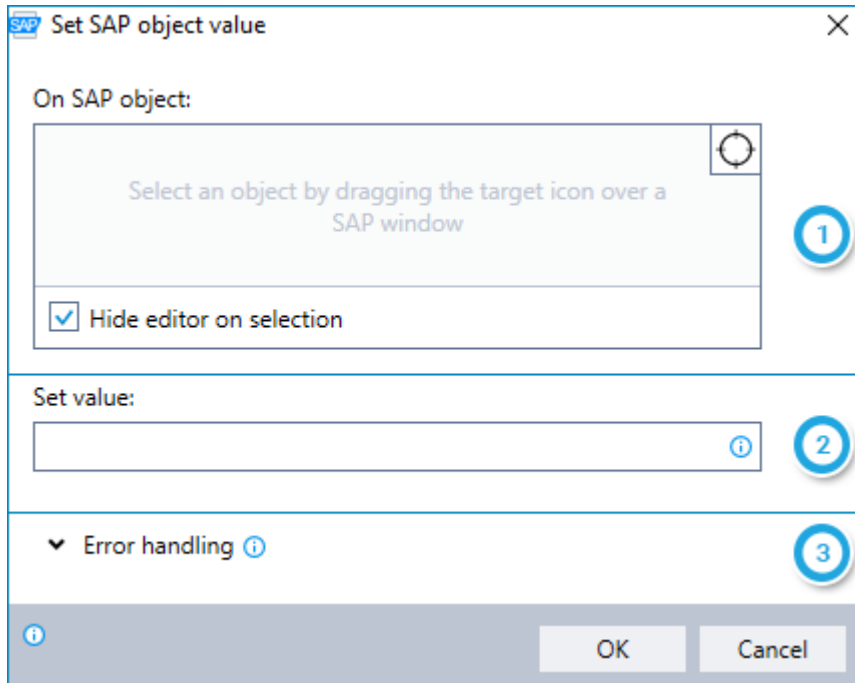
At the bottom of the dialog, there are 'OK' and 'Cancel' buttons, and an information icon on the left.





- 1 Choose whether to retrieve the object location either by rectangle or by center point
- 2 Enter the names of the variables into which you would like Leo to place the location information
- 3 Select the object whose location you would like to retrieve by dragging the  icon onto the object in an SAP window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - Info** Display additional information about the selected object
    - Test** Test retrieving the location of the selected object
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Set SAP Object Value

Place a value into an object in the active SAP window.

### Using the SET SAP OBJECT VALUE command



- 1 Select the object into which you would like to place a value by dragging the  icon onto the object in an SAP window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test placing a value into the selected object
- 2 Enter the value you would like to place (can be free text or copied from values stored in variables)
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 20: UI Automation Commands

In this chapter:

Get UI Object Text .....	277
Get UI Object Value .....	278
Get UI Object Location .....	279
Set UI Object Value .....	281



## NOTE

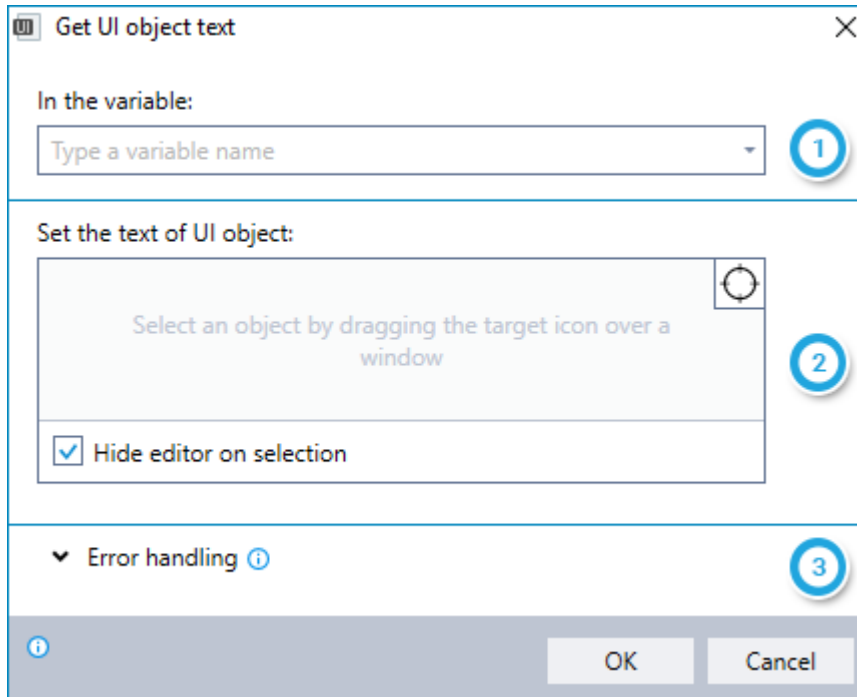
**Use in recorded steps only**





Use **UI AUTOMATION COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

## Get UI Object Text

Retrieve the text of an object in the active application and place it into a new or existing variable.

### Using the GET UI OBJECT TEXT command

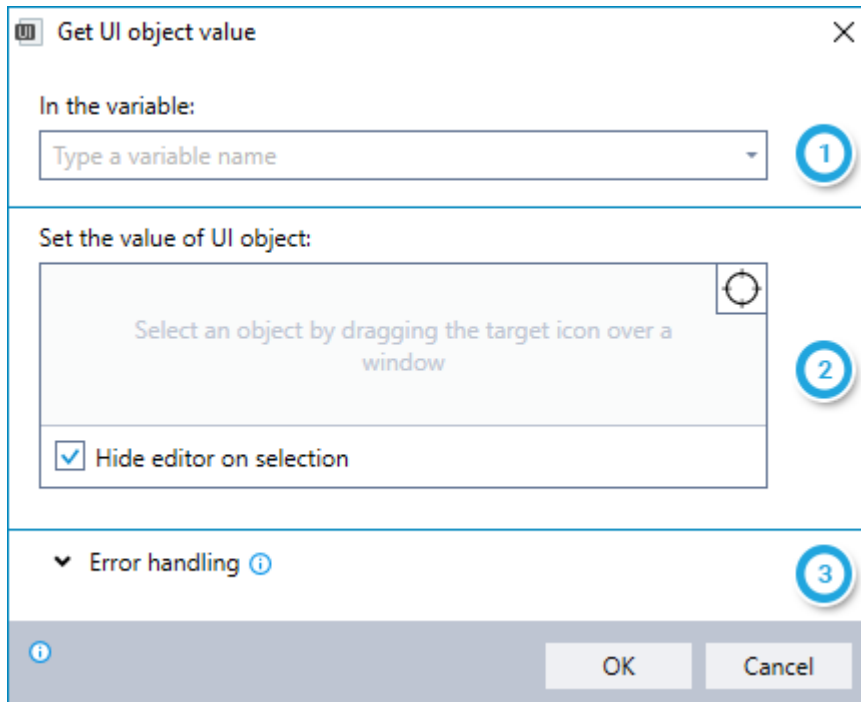






- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in an application window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test retrieving the text of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get UI Object Value

Retrieve the value of an object in the active application and place it into a new or existing variable. This command can be especially useful for checking the value of checkboxes, radio buttons, drop-down lists, etc.

### Using the GET UI OBJECT VALUE command



- 1 Enter the name of the variable into which you would like to place the value of the selected object
- 2 Select the object whose value you would like to retrieve by dragging the  icon onto the object in an application window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test retrieving the value of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get UI Object Location

Retrieve the location (in pixels) of an object in the active application and place it into new or existing variables. Choose to retrieve the location represented either by:

- **Rectangle** – with variables for **left**, **top**, **width**, and **height**; *or*
- **Center point** – with variables for **X** and **Y** coordinates



### TIP

#### Choose rectangle or center point first



The variables you need to specify while setting up this command vary based on the method you choose for retrieving the location. So make this selection first and save yourself some time!

## Using the GET UI OBJECT LOCATION command

The screenshot shows the 'Get UI object location' dialog box with the following elements:

- 1**: A dropdown menu set to 'rectangle' with the text 'Set the **rectangle** of selected UI object'.
- 2**: Four input fields for 'Left', 'Top', 'Width', and 'Height', each containing the placeholder text 'Type a variable name'.
- 3**: A large text area with the instruction 'Select an object by dragging the target icon over a window' and a target icon in the top right corner. Below it is a checked checkbox labeled 'Hide editor on selection'.
- 4**: An expanded 'Error handling' section with an information icon.

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

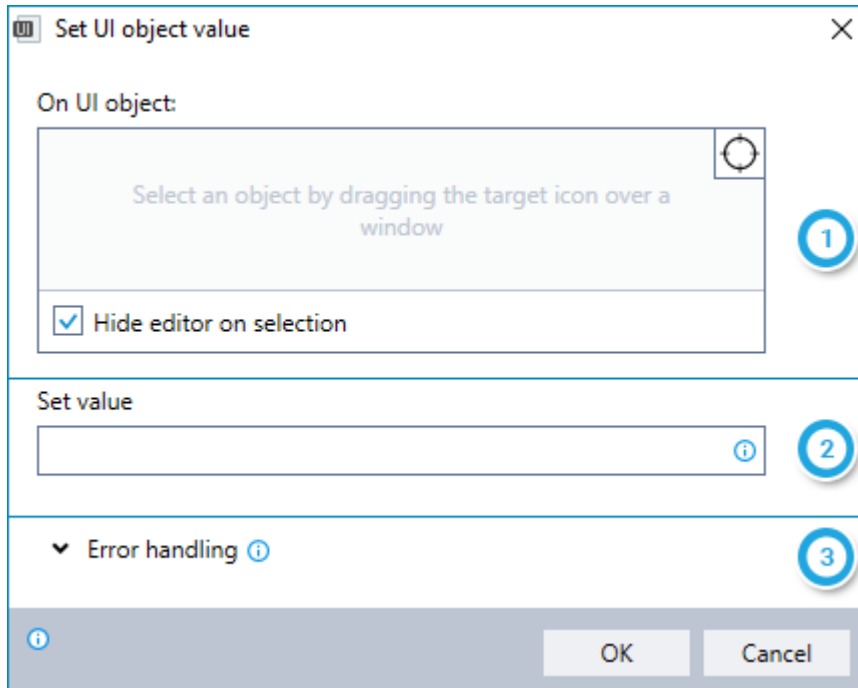
- 1 Choose whether to retrieve the object location either by rectangle or by center point
- 2 Enter the names of the variables into which you would like Leo to place the location information
- 3 Select the object whose location you would like to retrieve by dragging the  icon onto the object in an application window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - Info** Display additional information about the selected object
    - Test** Test retrieving the location of the selected object
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.







## Set UI Object Value

Place a value into an object in the active application.

### Using the SET UI OBJECT VALUE command



1 Select the object into which you would like to place a value by dragging the  icon onto the object in an application window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected object
  -  Test placing a value into the selected object

2 Enter the value you would like to place (can be free text or copied from values stored in variables)

3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 21: HTML Commands

In this chapter:

Get HTML Table .....	284
Get HTML Object Text .....	287
Get HTML Object Value .....	289
Get HTML Object .....	291
Set HTML Object Value .....	293
Does HTML Object Exist .....	295
Click on HTML Object .....	297



## NOTES

### Make sure it's Internet Explorer!

Leo's **HTML COMMANDS** support only the Internet Explorer browser. So, it's important to make sure:

- You are using Internet Explorer when you select objects as part of the wizard creation/editing process; and
- Internet Explorer is the active browser when the wizard is run

### Use in recorded steps only

Use **HTML COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.



## A BIT ABOUT jQuery

Leo's HTML commands include the option to identify an object by entering a jQuery selector.

jQuery selectors allow you to select and manipulate HTML elements based on their names, ids, classes, types, attributes, values of attributes, and more. jQuery utilizes existing CSS selectors (along with some additional selectors of its own).

For example, if the HTML element with which you want Leo to interact is an input box with the id **firstname**, you could use the jQuery selector `$ ("input#firstname")` in order to identify it.

**NOTE:** If your jQuery selector results in the selection of more than one HTML element on the page, Leo will return the error message `OBJECT_NOT_FOUND`.

## Get HTML Table

Retrieve the text of an HTML table in the active **Internet Explorer** window and place it into a new or existing variable. Choose to identify the table by:

- using Leo's drag and drop target; *or*
- entering a jQuery selector

To enable easy parsing, searching, and formatting, the data returned by this command is separated into columns and rows using the delimiters you specify.

## Using the GET HTML TABLE command

**Select an HTML object:**

Select an object by dragging the target icon over a web-browser window

Hide editor on selection

1

Use the field label as a reference for better accuracy

Select an object by dragging the target icon over a web-browser window

Hide editor on selection

2

Use jQuery selector:

\$(  )

[jQuery guide...](#)

3

Return the result (HTML object text) in variable:

Type a variable name

Column delimiter:




Row delimiter:



4




Error handling ?


5



?
OK
Cancel


To select the relevant table using Leo's drag-and-drop target, follow steps  and  below. To select the table using jQuery, skip to Step .

 Select the table whose text you would like to copy by dragging the  icon onto the object in an Internet Explorer window


- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected table
  -  Test retrieving the text of the selected table

 Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.


1. After selecting the table, a field label will appear in the box in section 
  - By default, Leo will select the field label nearest to the table you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

 Enter the jQuery selector for the table whose text you would like to copy. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

 Enter the name of the variable into which you would like to place the text of the selected table; *and*

the delimiters to use to separate each column and row in the returned data

 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get HTML Object Text

Retrieve the text of an HTML object in the active **Internet Explorer** window and place it into a new or existing variable. Choose to identify the relevant object by:

- using Leo's drag and drop target; *or*
- entering a jQuery selector


### Using the GET HTML OBJECT TEXT command


The screenshot shows the 'Get HTML object text' dialog box with the following sections and callouts:

- 1:** A large text area with a target icon in the top right corner. The text inside says 'Select an object by dragging the target icon over a web-browser window'. Below it is a checked checkbox labeled 'Hide editor on selection'.
- 2:** A second, identical text area and checkbox section.
- 3:** A radio button labeled 'Use jQuery selector:' followed by a text input field containing '\$(' and a closing parenthesis ')'. Below the input is a link that says 'jQuery guide...'.
- 4:** A section titled 'Return the result (HTML object text) in variable:' with a dropdown menu containing the text 'Type a variable name'.
- 5:** A section titled 'Error handling' with a downward arrow and an information icon.


At the bottom of the dialog are 'OK' and 'Cancel' buttons, and an information icon on the left.

To select an object using Leo's drag-and-drop target, follow steps [1](#) and [2](#) below. To select an object using jQuery, skip to Step [3](#).

[1](#) Select the object whose text you would like to copy by dragging the  icon onto the object in an Internet Explorer window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  - [Info](#) Display additional information about the selected object
  - [Test](#) Test retrieving the text of the selected object

[2](#) Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object whose text you would like to copy, a field label will appear in the box in section [2](#)
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

[3](#) Enter the jQuery selector for the object whose text you would like to copy. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

[4](#) Enter the name of the variable into which you would like to place the text of the selected object

[5](#) Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



## Get HTML Object Value

Retrieve the value of an HTML object in the active **Internet Explorer** window and place it into a new or existing variable. This command can be especially useful for checking the value of checkboxes, radio buttons, drop-down lists, etc. Choose to identify the relevant object by:

- using Leo's drag and drop target; *or*
- entering a jQuery selector


### Using the GET HTML OBJECT VALUE command


The screenshot shows the 'Get HTML object value' dialog box with the following sections and callouts:

- Select an HTML object:** A large text area with a target icon in the top right corner. Below it is a checkbox labeled 'Hide editor on selection' which is checked.
- Use the field label as a reference for better accuracy:** A checkbox which is unchecked. Below it is another large text area with a target icon in the top right corner. Below that is another checkbox labeled 'Hide editor on selection' which is checked.
- Use jQuery selector:** A radio button which is unselected. Below it is a text input field containing '\$(' and a closing parenthesis ')'. Below the input field is a link labeled 'jQuery guide...'. A small 'jQuery guide...' link is also visible below the input field.
- Return the result (HTML object value) in variable:** A dropdown menu with the text 'Type a variable name' and a downward arrow.
- Error handling:** A dropdown menu with a downward arrow and an information icon.


At the bottom of the dialog are 'OK' and 'Cancel' buttons, and an information icon on the left.

To select an object using Leo's drag-and-drop target, follow steps **1** and **2** below. To select an object using jQuery, skip to Step **3**.

**1** Select the object whose value you would like to retrieve by dragging the  icon onto the object in an Internet Explorer window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  - Info** Display additional information about the selected object
  - Test** Test retrieving the value of the selected object

**2** Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object whose value you would like to retrieve, a field label will appear in the box in section **2**
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

**3** Enter the jQuery selector for the object whose value you would like to retrieve. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

**4** Enter the name of the variable into which you would like to place the value of the selected object

**5** Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get HTML Object

Retrieve the HTML code for a selected object in the active **Internet Explorer** window and place it into a new or existing variable. Choose to identify the relevant object by:




- using Leo's drag and drop target; *or*
- entering a jQuery selector



### Using the GET HTML OBJECT command




The screenshot shows the 'Get HTML object' dialog box with the following sections and callouts:


- 1**: The 'Select an HTML object:' section, which includes a large text area with a target icon and the instruction 'Select an object by dragging the target icon over a web-browser window'. A checkbox for 'Hide editor on selection' is checked.
- 2**: The 'Use the field label as a reference for better accuracy' section, which includes another large text area with a target icon and the same instruction. A checkbox for 'Hide editor on selection' is checked.
- 3**: The 'Use jQuery selector:' section, which features a text input field containing '\$(' and a closing parenthesis ')'. Below the field is a link for 'jQuery guide...'.
- 4**: The 'Return the result (HTML object) in variable:' section, which contains a dropdown menu with the placeholder text 'Type a variable name'.
- 5**: The 'Error handling' section, which is currently collapsed.



At the bottom of the dialog, there are 'OK' and 'Cancel' buttons, and an information icon on the left.


To select an object using Leo's drag-and-drop target, follow steps  and  below. To select an object using jQuery, skip to Step .

 Select the object whose code you would like to copy by dragging the  icon onto the object in an Internet Explorer window


- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected object
  -  Test retrieving the HTML code of the selected object


 Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object whose code you would like to copy, a field label will appear in the box in section 
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

 Enter the jQuery selector for the object whose code you would like to copy. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

 Enter the name of the variable into which you would like to place the HTML code of the selected object

 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Set HTML Object Value

Place a value into an object in the active **Internet Explorer** window. Choose to identify the relevant object by:

- using Leo's drag and drop target; *or*
- entering a jQuery selector


### Using the SET HTML OBJECT VALUE command




The screenshot shows the 'Set HTML object value' dialog box with the following sections and callouts:

- 1**: The 'Select an HTML object:' section, which includes a large text area with a target icon and the instruction 'Select an object by dragging the target icon over a web-browser window'. A checkbox labeled 'Hide editor on selection' is checked.
- 2**: The 'Use the field label as a reference for better accuracy' section, which includes another large text area with a target icon and the same instruction. A checkbox labeled 'Hide editor on selection' is checked.
- 3**: The 'Use jQuery selector:' section, which includes a text input field containing '\$(' and a closing parenthesis ')'. Below the field is a link labeled 'jQuery guide...'. A 'jQuery guide...' link is also present below the input field.
- 4**: The 'Set the value:' section, which includes a text input field.
- 5**: The 'Error handling' section, which includes a dropdown menu.


At the bottom of the dialog box, there are 'OK' and 'Cancel' buttons, and an information icon on the left.

To select an object using Leo's drag-and-drop target, follow steps **1** and **2** below. To select an object using jQuery, skip to Step **3**.

**1** Select the object into which you would like to place a value by dragging the  icon onto the object in an Internet Explorer window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected object
  -  Test placing a value into the selected object

**2** Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object into which you would like to place a value, a field label will appear in the box in section **2**
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

**3** Enter the jQuery selector for the object into which you would like to place a value. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

**4** Enter the value you would like to place (can be free text or copied from values stored in variables)

**5** Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Does HTML Object Exist

Check to see if an object in the active **Internet Explorer** window exists and place the result of the check (TRUE/FALSE) into a variable. Choose to identify the relevant object by:

- using Leo's drag and drop target; *or*
- entering a jQuery selector

### Using the DOES HTML OBJECT EXIST command

**Does HTML object exist**

**Select an HTML object:**

Select an object by dragging the target icon over a web-browser window

Hide editor on selection

Use the field label as a reference for better accuracy

Select an object by dragging the target icon over a web-browser window

Hide editor on selection

**Use jQuery selector:**




\$( )



[jQuery guide...](#)




Return the result (TRUE/FALSE) in variable:


Type a variable name



OK Cancel


To select an object using Leo's drag-and-drop target, follow steps  and  below. To select an object using jQuery, skip to Step .

 Select the object whose existence you would like to check by dragging the  icon onto the object in an Internet Explorer window


- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected object
  -  Test checking the existence of the selected object

 Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object whose existence you would like to check, a field label will appear in the box in section 
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

 Enter the jQuery selector for the object whose existence you would like to check. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

 Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)



## Click on HTML Object

Click the mouse on an object in the active **Internet Explorer** window. Choose to identify the relevant object by:




- using Leo's drag and drop target; *or*
- entering a jQuery selector



### Using the **CLICK ON HTML OBJECT** command




The screenshot shows the 'Click on HTML object' dialog box with the following sections and callouts:


- 1**: The first section, 'Select an HTML object:', contains a large text area with a target icon in the top right corner and the text 'Select an object by dragging the target icon over a web-browser window'. Below the text area is a checkbox labeled 'Hide editor on selection' which is checked.
- 2**: The second section, 'Use the field label as a reference for better accuracy', contains a similar text area and target icon. Below it is a checkbox labeled 'Hide editor on selection' which is checked.
- 3**: The third section, 'Use jQuery selector:', contains a text input field with '\$(' and ')' characters and a 'jQuery guide...' link below it.
- 4**: The fourth section, 'Asynchronous mode', contains a checkbox and the text: 'Use this option if Leo is blocked due to the click action. This usually happens when a system dialog window is open. Note: Error handling is not available in this mode'.
- 5**: The fifth section, 'Error handling', is expanded to show an information icon.



At the bottom of the dialog are 'OK' and 'Cancel' buttons.


To select an object using Leo's drag-and-drop target, follow steps  and  below. To select an object using jQuery, skip to Step .

 Select the object on which you would like to click by dragging the  icon onto the object in an Internet Explorer window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  -  Display additional information about the selected object
  -  Test clicking the selected object


 Indicate whether you would like to use the field label as a reference. Use this option to improve accuracy when selection of a non-unique object is required.

1. After selecting the object on which you would like to click, a field label will appear in the box in section 
  - By default, Leo will select the field label nearest to the object you selected
2. If you would like to use the default field label as a reference, indicate so by ticking the checkbox
3. If you would like to use a different field label as a reference, tick the checkbox, then select the desired field label by dragging the  icon onto it in the Internet Explorer window

 Enter the jQuery selector for the object on which you would like to click. (No need to type opening and closing syntax... it's already there for you.) To learn more about jQuery, see [A BIT ABOUT jQuery](#).

Note that this method does not support iframes.

 Indicate whether you would like to use Asynchronous mode

 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

# CHAPTER 22: .NET Automation Commands

In this chapter:

Get .NET Object Text .....	300
Get .NET Object Value .....	301
Get .NET Object Location .....	302
Set .NET Object Value .....	304



## NOTE

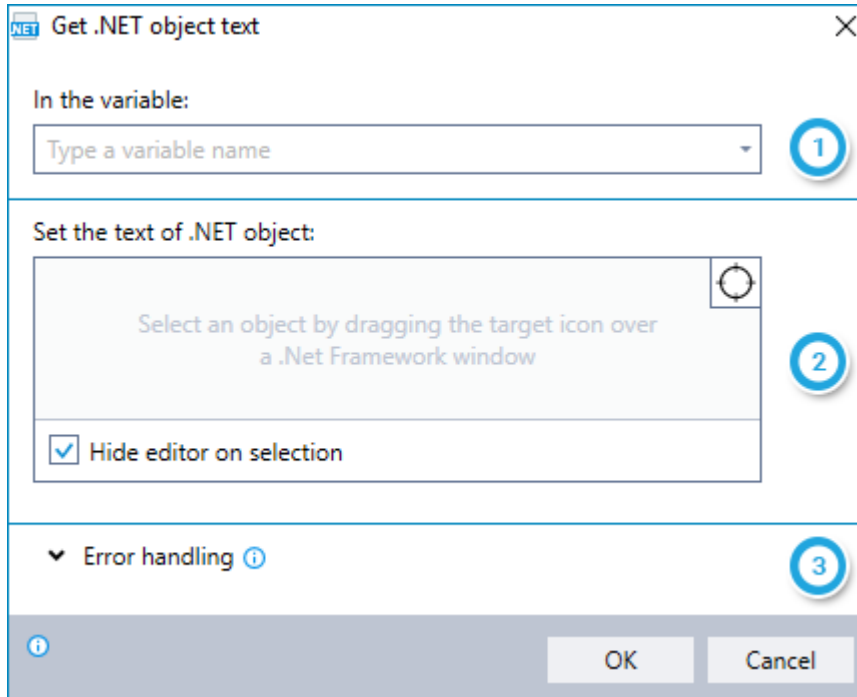
**Use in recorded steps only**



Use **.NET AUTOMATION COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

## Get .NET Object Text

Retrieve the text of an object in the active .NET Framework window and place it into a new or existing variable.

### Using the GET .NET OBJECT TEXT command





- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in a .NET Framework window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - [Info](#) Display additional information about the selected object
    - [Test](#) Test retrieving the text of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get .NET Object Value

Retrieve the value of an object in the active .NET Framework window and place it into a new or existing variable. This command can be especially useful for checking the value of checkboxes, radio buttons, drop-down lists, etc.

### Using the GET .NET OBJECT VALUE command

- 1 Enter the name of the variable into which you would like to place the value of the selected object
- 2 Choose whether to retrieve the object value by text or by index
- 3 Select the object whose value you would like to retrieve by dragging the  icon onto the object in a .NET Framework window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - Info** Display additional information about the selected object
    - Test** Test retrieving the value of the selected object
- 4 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get .NET Object Location

Retrieve the location (in pixels) of an object in the active .NET Framework window and place it into new or existing variables. Choose to retrieve the location represented either by:

- **Rectangle** – with variables for **left**, **top**, **width**, and **height**; *or*
- **Center point** – with variables for **X** and **Y** coordinates



### TIP

#### Choose rectangle or center point first



The variables you need to specify while setting up this command vary based on the method you choose for retrieving the location. So make this selection first and save yourself some time!

## Using the GET .NET OBJECT LOCATION command

The screenshot shows the 'Get .NET object location' dialog box with the following elements:

- 1**: A dropdown menu set to 'rectangle' with the text 'Set the **rectangle** of selected .NET object'.
- 2**: Four input fields for 'Left', 'Top', 'Width', and 'Height', each containing the placeholder text 'Type a variable name'.
- 3**: A large text area containing the instruction 'Select an object by dragging the target icon over a .Net Framework window' and a checkbox labeled 'Hide editor on selection' which is checked.
- 4**: An expanded 'Error handling' section with an information icon.

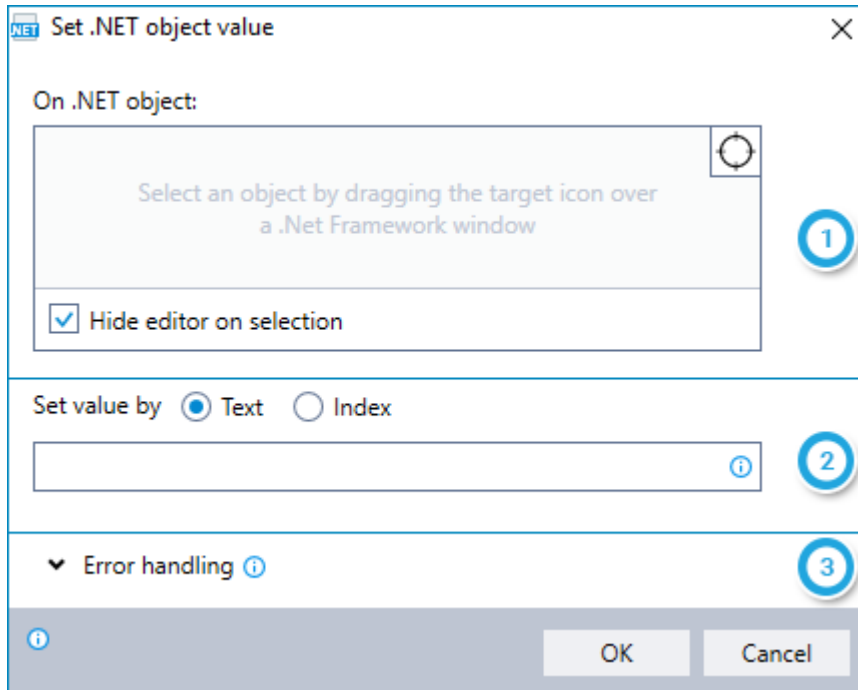
At the bottom of the dialog are 'OK' and 'Cancel' buttons.





- 1 Choose whether to retrieve the object location either by rectangle or by center point
- 2 Enter the names of the variables into which you would like Leo to place the location information
- 3 Select the object whose location you would like to retrieve by dragging the  icon onto the object in a .NET Framework window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - Info** Display additional information about the selected object
    - Test** Test retrieving the location of the selected object
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.

## Set .NET Object Value

Place a value into an object in the active .NET Framework window.

### Using the SET .NET OBJECT VALUE command



- 1 Select the object into which you would like to place a value by dragging the  icon onto the object in a .NET window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test placing a value into the selected object
- 2 Choose whether to place the value by text or index; **and** enter the value you would like to place (can be free text or copied from values stored in variables)
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



# CHAPTER 23: Java Automation Commands

In this chapter:

Get Java Object Text .....	307
Get Java Object Value .....	308
Get Java Object Location .....	310
Set Java Object Value .....	312



## NOTES

### Install the Leo Java bridge first

Use of **JAVA AUTOMATION COMMANDS** requires installation of the Leo Java bridge on: (i) your robots; and (ii) the machine(s) on which Leo Studio is installed. For additional information, contact your Leo Support team.

### Use in recorded steps only

Use **JAVA AUTOMATION COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

### Supported Java runtime versions

- 1.4.2
- 1.5.0 and above (beta)

### Supported Java GUI framework:

- SWING (get & set values)
- AWT (get values)

### Supported controls:

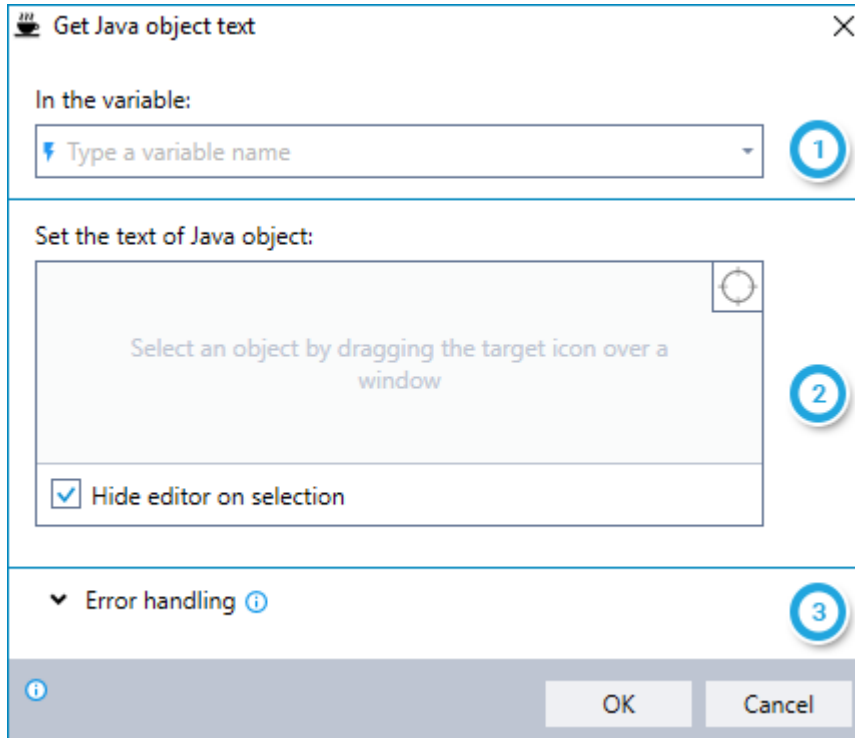
- Text box
- Button
- Label



- Radio button
- Checkbox
- Combo box
- List box

## Get Java Object Text

Retrieve the text of an object in the active Java window and place it into a new or existing variable.

### Using the GET JAVA OBJECT TEXT command







- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in a Java window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - [Info](#) Display additional information about the selected object
    - [Test](#) Test retrieving the text of the selected object
- 3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get Java Object Value

Retrieve the value of an object in the active Java window and place it into a new or existing variable. This command can be especially useful for checking the value of checkboxes, radio buttons, drop-down lists, etc.

### Using the GET JAVA OBJECT VALUE command

- 1 Enter the name of the variable into which you would like to place the value of the selected object
- 2 Choose whether to retrieve the object value by text or by index
- 3 Select the object whose value you would like to retrieve by dragging the  icon onto the object in a Java window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    -  Display additional information about the selected object
    -  Test retrieving the value of the selected object



Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).

## Get Java Object Location

Retrieve the location (in pixels) of an object in the active Java window and place it into new or existing variables. Choose to retrieve the location represented either by:

- **Rectangle** – with variables for `left`, `top`, `width`, and `height`; *or*
- **Center point** – with variables for `X` and `Y` coordinates





### TIP

#### Choose rectangle or center point first

The variables you need to specify while setting up this command vary based on the method you choose for retrieving the location. So make this selection first and save yourself some time!

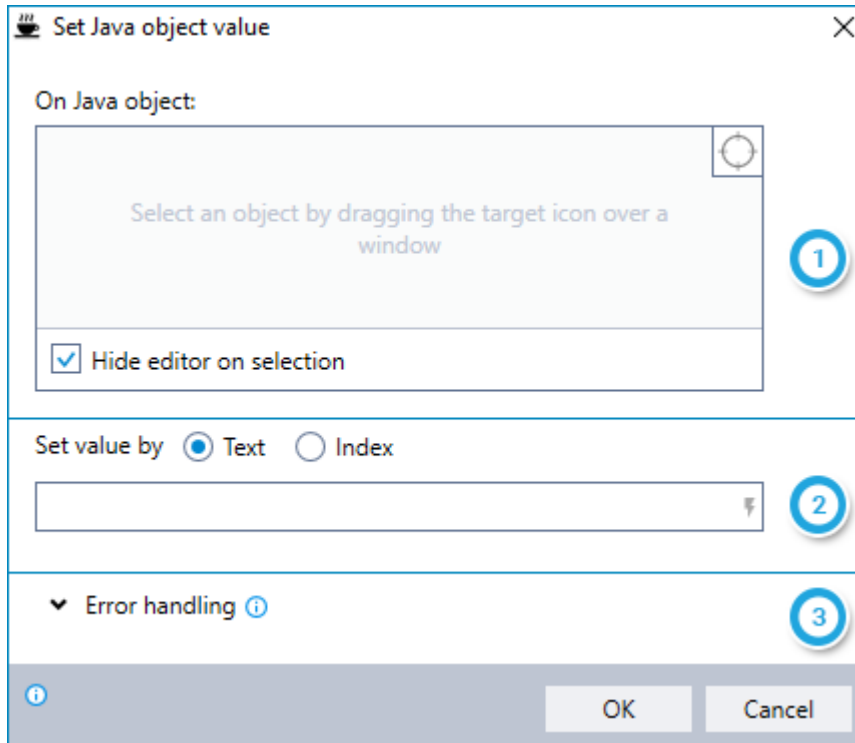
## Using the GET JAVA OBJECT LOCATION command


- 1 Choose whether to retrieve the object location either by rectangle or by center point
- 2 Enter the names of the variables into which you would like Leo to place the location information
- 3 Select the object whose location you would like to retrieve by dragging the  icon onto the object in a Java window
  - Indicate whether you would like to hide Leo Studio while you are selecting the object
  - After selecting the object, the following additional options will become available:
    -  Configure available additional settings
    - Info** Display additional information about the selected object
    - Test** Test retrieving the location of the selected object
- 4 Instruct Leo how to handle any errors encountered. Read more about **ERROR HANDLING**.


## Set Java Object Value

Place a value into an object in the active Java window.

### Using the SET JAVA OBJECT VALUE command



1 Select the object into which you would like to place a value by dragging the  icon onto the object in a Java window

- Indicate whether you would like to hide Leo Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
  -  Configure available additional settings
  - [Info](#) Display additional information about the selected object
  - [Test](#) Test placing a value into the selected object

2 Choose whether to place the value by text or index; **and** Enter the value you would like to place (can be free text or copied from values stored in variables)

3 Instruct Leo how to handle any errors encountered. Read more about [ERROR HANDLING](#).



# CHAPTER 24: Global Variable Commands

In this chapter:

Set Global Variable .....	314
Get Global Variable .....	315
Delete Global Variable .....	316

## Set Global Variable

Place a value into a variable that is available for use in other wizards and sensors (a "global variable"), by either:

- Creating a new global variable and setting its value; or
- Setting the value of an global existing variable

### Using the SET GLOBAL VARIABLE command

- 1 Enter the name of the global variable (new or existing) to which you want to assign a value
  - If you want to create a new global variable, type the name of the new variable
  - If the global variable already exists, choose its name from the drop-down list
- 2 Set the value of the global variable you have specified
  - You can include free text and/or values copied from different variables
  - <Enter> <Space> and/or <Tab> can be used



#### TIP

##### Turn something standard into something global...

Very often, the value set into a global variable is actually the value of one of the current wizard's "standard" variables. Just type the standard variable's name

between dollar signs (e.g., \$MyVar\$) in step 2 to make this happen.

## Get Global Variable

Retrieve the value of a global variable (previously created in a different wizard using the **SET GLOBAL VARIABLE** command) and place it into a standard variable in the current wizard.



### NOTE

In order to use a global variable in the current wizard, its value must first be placed into a standard variable.

## Using the GET GLOBAL VARIABLE command

The screenshot shows a dialog box titled "Get global variable" with a close button (X) in the top right corner. The dialog is divided into two main sections. The first section is labeled "In the variable:" and contains a dropdown menu with the placeholder text "Type a variable name". A blue circle with the number "1" is positioned to the right of this dropdown. The second section is labeled "Set the global variable value:" and also contains a dropdown menu with the placeholder text "Type a variable name". A blue circle with the number "2" is positioned to the right of this dropdown. At the bottom of the dialog, there is a grey bar containing an information icon (i) on the left, and "OK" and "Cancel" buttons on the right.

- 1 Enter the name of the standard variable into which you would like to place the value of a global variable
- 2 Enter the name of the existing global variable whose value you want to place into the specified standard variable

## Delete Global Variable

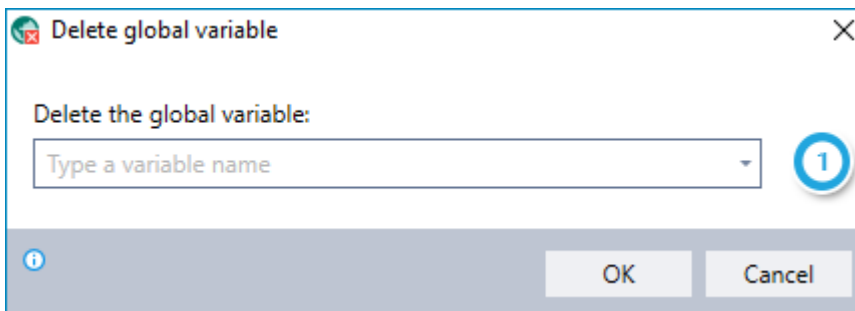
Delete a global variable (previously created in this or a different wizard using the **SET GLOBAL VARIABLE** command) so that it is no longer available for use.



### CAUTION

Deleting a global variable does not merely clear its value; it deletes the variable itself.

## Using the DELETE GLOBAL VARIABLE command



Enter the name of the global variable you would like to delete

# CHAPTER 25: Scripting Commands

In this chapter:

Note .....	318
Notes for Mobile/Web Access .....	319
View Variable List .....	320
Show Debug Message .....	322

## Note

Enter an internal note to appear in the Editor Pane of the Advanced Commands view.



### TIP

#### Just do it.

Anyone who has ever developed code knows how important it is to document it internally. But all too often, in the rush of getting things done, this crucial practice is overlooked. Don't let it happen to you!

Enter notes to document various sections of the wizard and its logical flow. You'll be surprised how many hours and headaches it will save when you (or someone else) is revising, updating, or debugging.

A few examples:

- *Fallback for when the date is empty*
- *This is the end point of a loop*
- *Ensures that the wizard will continue even if the file can't be found*

## Using the NOTE command

**1** Enter the text of the note as you want it to appear

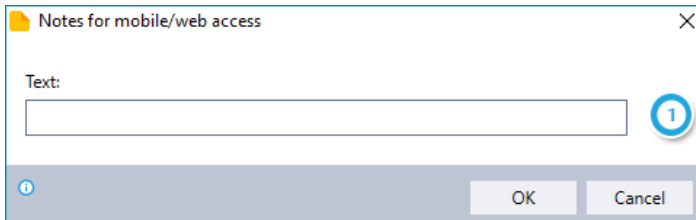
- The note will appear only in the Advanced Commands Editor, so end users will never see it when the wizard/sensor is run

## Notes for Mobile/Web Access

Enter an internal note to appear on the mobile/webpage summary of the wizard.

- Applicable only if Leo Mobile/Web Access has been deployed and configured for your organization

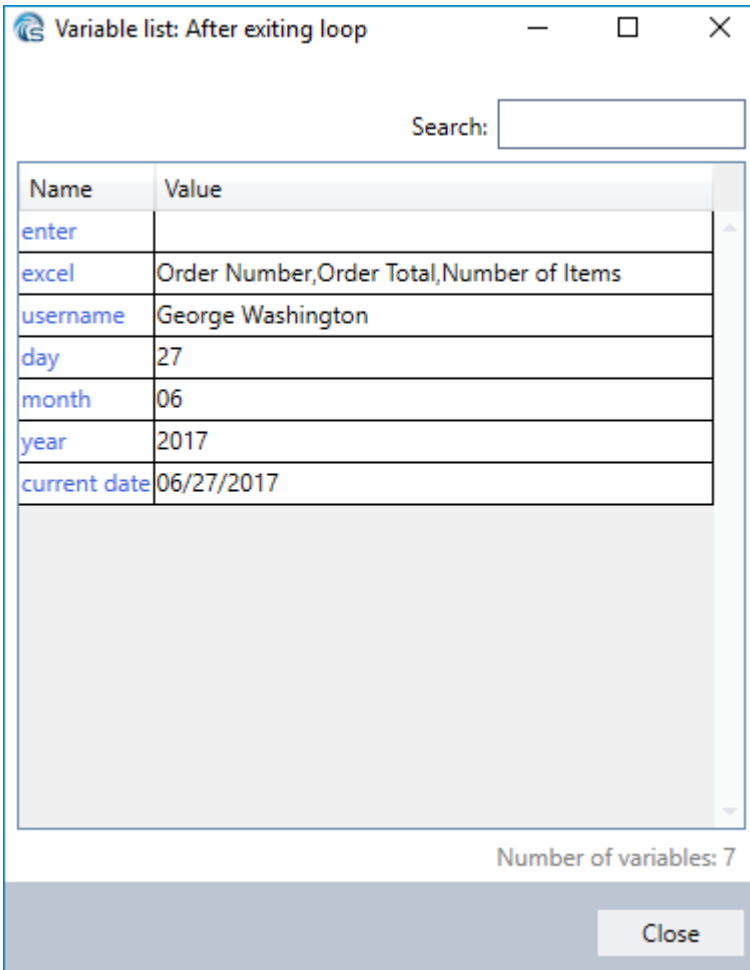
### Using the NOTES FOR MOBILE/WEB ACCESS command



- 1 Enter the text of the note as you want it to appear on the mobile/web summary page

## View Variable List

Display a list of variables and their values as they would stand at any specific point during execution of the wizard. Here's a sample:



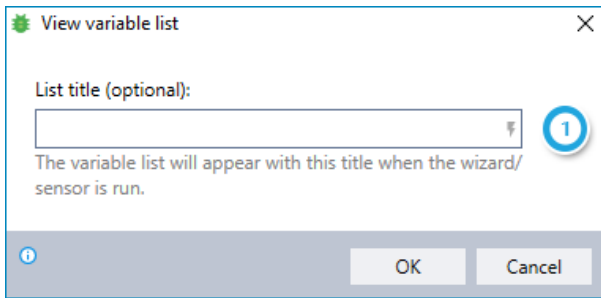
### CAUTION

**Be sure to remove these before publishing to production!**

Variable lists can be extremely valuable to help you know where things stand as you are developing a wizard's logic. But be sure to remove (or disable) any **VIEW VARIABLE LIST** commands before you release the wizard to production. No need for anyone see these when the wizard is run.



## Using the VIEW VARIABLE LIST command



**1** (Optional) Enter a list title that will appear when the wizard is run

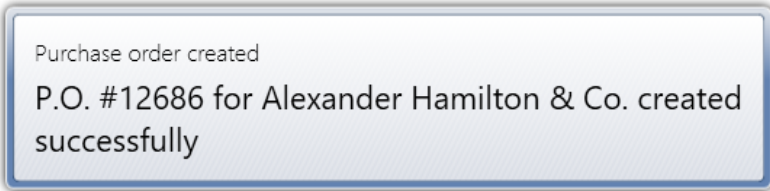


### TIP

Give your list a title that will help you identify the point at which it was created.

## Show Debug Message

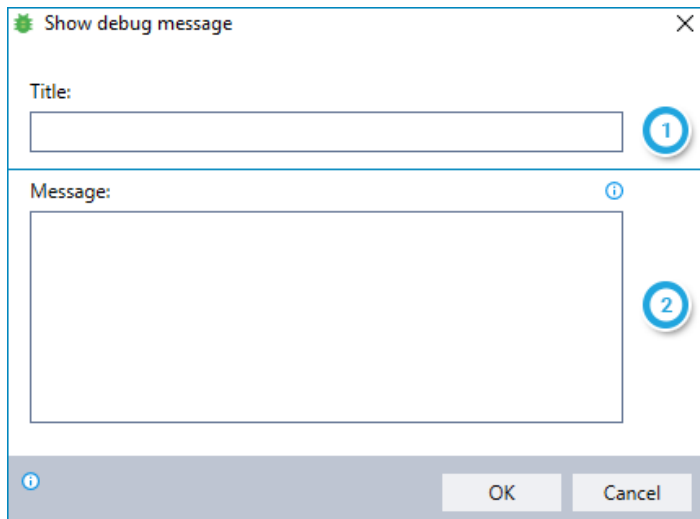
Display a text message to the Leo Studio user when the wizard is run in debug mode. Here's a sample:



### NOTES

- Messages like these help you know when the wizard has reached a certain point in its logic (and if it was reached successfully)
- Debug messages only appear when the wizard is run in debug mode, so there's no need to remove them before the wizard is released to production (i.e., the end user won't see them when the wizard/sensor is run normally)

## Using the SHOW DEBUG MESSAGE command



Enter the title of the message



Enter the text of the message as you want it to appear

- To incorporate a variable value within the text, type the variable's name between dollar signs (e.g., \$MyVar\$)

# APPENDIX A: Error Handling

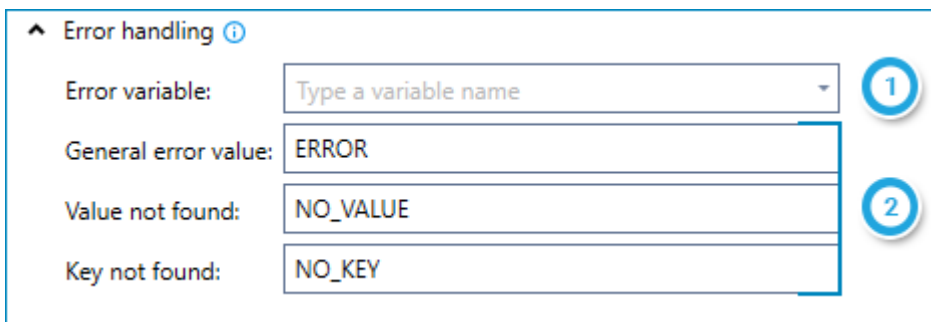
In a perfect world, every computer system would run without freezing. Every file would be found. A 0 (zero) would never be mistaken for a capital O.

But the truth is we live in the real world, and that's why many of Leo's Advanced Commands include a section dedicated to **ERROR HANDLING**... so that even when errors happen (as they inevitably do), they are easier to track and correct.

## Using ERROR HANDLING options

You can specify how Leo should report errors in any Advanced Command in which you see

options:



- 1** Enter the name of the variable in which an error message should be placed
  - This option is generally only available for Advanced Commands that:
    - Do not have a variable in which a regular value is returned (a "**return variable**"); *or*
    - Have more than one return variable
  - For commands with **one** return variable, an error message will be placed in the return variable
- 2** Customize default error messages for the various types of errors that might occur during execution of the command
  - This option is available for all Advanced Commands that offer **ERROR HANDLING**