



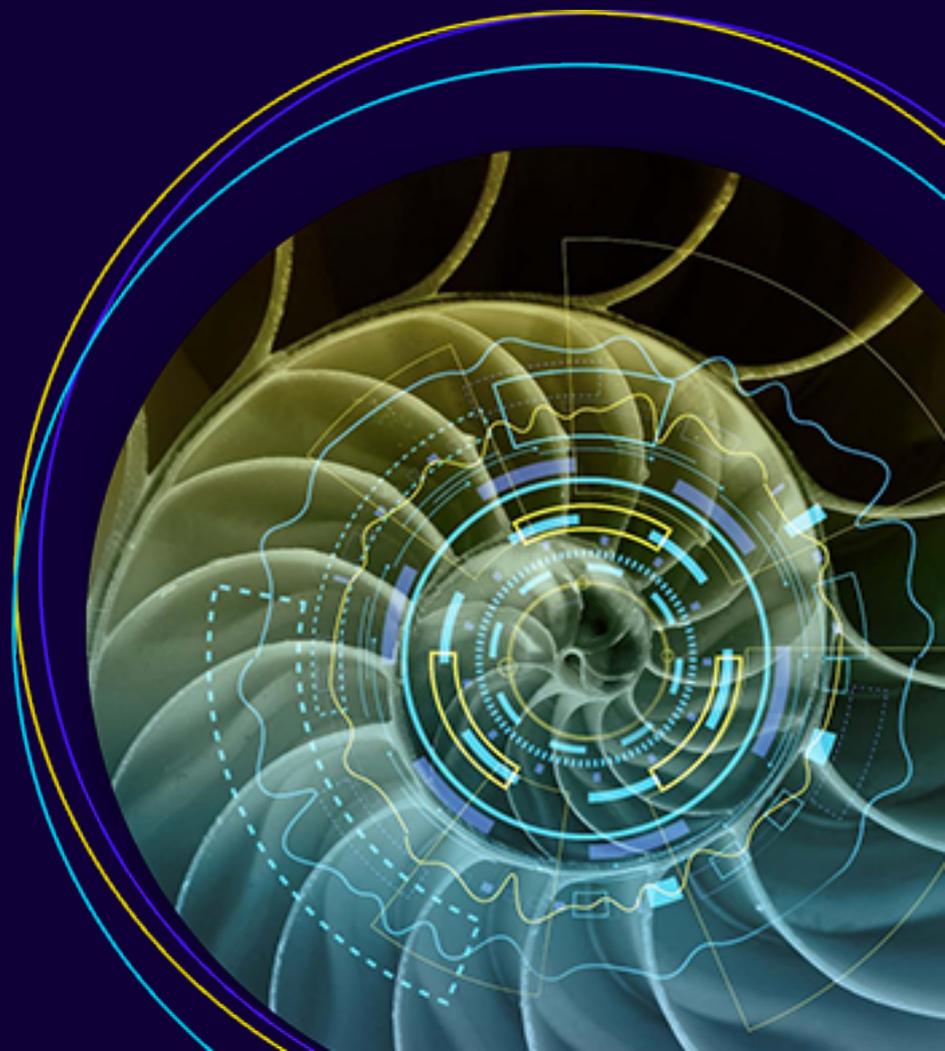
Advanced Commands Reference Guide

Kryon Studio v19.5.1

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

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

Document revision: 30-Jan-2020



Contents

CHAPTER 1: Variable Commands

Set Value	11
Find	13
Replace	21
Mathematics	23
Evaluate Expression	24
Split	29
Get Array Data	33
Get Table Data	37
Table Lookup	42
Get ASCII Character	47
Remove Blank Spaces	49
Change Letter Case	50
Check Type	51
Get Random Number	52
Reverse	53
Get Length	54
Extract Numeric Values	55

CHAPTER 2: Flow Commands

If Else	59
Complex If Else	63
Multi-Value If Else	65
Loop	67
Loop: Break	69
Loop: Restart	70
Loop Table	71
Loop Items	73
Pause	75
Group	76

CHAPTER 3: Wizard Commands

Continue Wizard	80
-----------------------	----

End Wizard	81
Go To Step	82
Get Step Data	84
Get Wizard Data	85
Check Application	86
Get User Data	87
Check Run Mode	88
Check Video Recording Mode	89
Set User Interrupt Mode	90
Show Message	91
Raise Wizard Error	93
Get Last Failure Type	94
Resume Error	95
Report Wizard Output	96

CHAPTER 4: Mouse and Keyboard Commands

Use Keyboard Shortcut	98
Input Text	99
Get Mouse Position	100
Drag & Drop	101
Mouse Click	103
Wait for Busy Cursor	104
Set Caps Lock State	105
Get Caps Lock State	106

CHAPTER 5: Block Commands

Allow Last Stored Action	107
Deny Last Stored Action	107
Remove All Blocks	108

CHAPTER 6: Date and Time Commands

Get Current Date	111
Validate Date	112
Get Day of Month	113
Compare Dates	115

Add/Subtract Date	117
Calculate Date Range	118
Check Day of Week	120
Format Date	122
Get Current Time	125
Compare Time	126
Add/Subtract Time	128
Calculate Time Range	129
Convert Between Time Zones	131

CHAPTER 7: Window Commands

Get Step Window Handle	133
Find Matching Window Handles	134
Get Active Application	136
Control Window State	137
Check Window State	138
Get Active Window/Web Page	139
Capture Step Window Image	140

CHAPTER 8: File Commands

Create a Text File	142
Read From Text File	144
Write to Text File	145
Does File Exist	146
Copy a File	147
Move a File	148
Rename a File	149
Delete a File	150
Delete File(s)	151
Monitor File Changes	153

CHAPTER 9: Folder Commands

Create a Folder	156
Get Folder Location	157
Get Files	158

Does Folder Exist	160
Is Folder Empty	161
Copy a Folder	162
Move a Folder	163
Rename a Folder	164
Delete a Folder	165
Monitor Folder Changes	166

CHAPTER 10: AI-Powered Document and Text Analysis Commands

What's Required?	169
Azure Licenses	171
OCR: Documents	172
Get Text (SmartScan)	176
Get Value	177
Get Selection Element State	179
Does Word Exist (SmartScan)	181
Get Date/Time	182
Save as Image	184
Save to Excel	185
OCR: Printed and Handwritten Text	187
Get Text (MS Azure)	190
Does Word Exist (MS Azure)	191
Form Recognizer	192
Get Receipt Data	195
Text Analytics: Analyze Sentiment	197
Text Analytics: Detect Language	198
Text Analytics: Identify Key Phrases	199
Convert to Text (SmartScan+)	200
Convert to Text (Tesseract)	201

CHAPTER 11: Digital PDF Analysis Commands

Analyze Digital PDF File	203
Page: Get Text	206

CHAPTER 12: External Data Commands

Get From Clipboard	208
Place in Clipboard	209
Copy Active Field Value	210
Get OS Version	211
Get Input Language	212
Read From Registry	213
Remove From Registry	215
Write to Registry	216
Get Environment Variable	217
Get Windows Event Log Data	218
Set BI Field	220
Log an Action	221
Query XML	222
Query JSON	224
Call REST API Method	226

CHAPTER 13: Email Commands

Send Email Message	229
Get Email Messages	236
Email: Get Data	243
Email: Move to Folder	244
Email: Forward	246
Email: Reply	248
Email: Delete	249
Email: Mark as Read/Unread	250
Email: Save Attachments	251
Email: Save Message	253

CHAPTER 14: External Program Commands

Run Program	255
Run .NET Plugin Method	257
Call Web Service Method	259
Run Script	260

Open URL Link (beta)	262
Run curl Command	263
Get Web Page HTML	264
Run JavaScript on Page	265
CHAPTER 15: Database Commands	
Execute SQL Query	268
Monitor Database Changes	271
CHAPTER 16: Credentials Vault Commands	
Insert Username Into Active Field	274
Insert Password Into Active Field	275
Generate New Password	276
Revert Password	277
CHAPTER 17: Robotic Process Automation Commands	
Add Automation Task to Queue	279
Get Automation Task Status	282
Get File Trigger Input	284
Get Folder Trigger Input	285
Get Email Trigger Input	286
Get Database Trigger Input	287
CHAPTER 18: Excel Commands	
Copy from Excel	290
Paste to Excel	294
Delete from Excel	298
Excel Worksheet Actions	301
Excel Row Actions	302
Excel Column Actions	304
Convert Excel to CSV	306
Create New Excel File	307
Query From Excel	308
Run Macro	310
CHAPTER 19: SAP Commands	
Get SAP Object Text	313

Get SAP Object Value	314
Get SAP Object Location	315
Set SAP Object Value	317

CHAPTER 20: UI Automation Commands

Get UI Object Text	319
Get UI Object Value	320
Get UI Object Location	321
Set UI Object Value	323

CHAPTER 21: HTML Commands

HTML Object Selector	326
Get HTML Table	331
Get HTML Object Text	333
Get HTML Object Value	334
Get HTML Object	335
Set HTML Object Value	336
Does HTML Object Exist	337
Click on HTML Object	338
Scroll to HTML Object	339
Extract from HTML Table/List	340

CHAPTER 22: .NET Automation Commands

Get .NET Object Text	351
Get .NET Object Value	352
Get .NET Object Location	353
Set .NET Object Value	355

CHAPTER 23: Java Automation Commands

Get Java Object Text	358
Get Java Object Value	359
Get Java Object Location	360
Set Java Object Value	362

CHAPTER 24: Global Variable Commands

Set Global Variable	364
Get Global Variable	365

Delete Global Variable	366
CHAPTER 25: Scripting Commands	
Note	368
Notes for Mobile/Web Access	369
View Variable List	370
Show Debug Message	372
APPENDIX A: Error Handling	
Using ERROR HANDLING options	373
APPENDIX B: Kryon Connector Chrome Extension	

CHAPTER 1: Variable Commands

In this chapter:

Set Value	11
Find	13
Replace	21
Mathematics	23
Evaluate Expression	24
Split	29
Get Array Data	33
Get Table Data	37
Table Lookup	42
Get ASCII Character	47
Remove Blank Spaces	49
Change Letter Case	50
Check Type	51
Get Random Number	52
Reverse	53
Get Length	54
Extract Numeric Values	55

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). Below the title bar, there are two sections: 'In the variable:' with a dropdown menu showing 'separator', and 'Set the value:' with a text input field containing the character '/'. At the bottom of the dialog, there are 'OK' and 'Cancel' buttons, and a small information icon (i) on the left.

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). Below the title bar, there are two sections: 'In the variable:' with a dropdown menu 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 'OK' and 'Cancel' buttons, and a small information icon (i) on the left.

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.

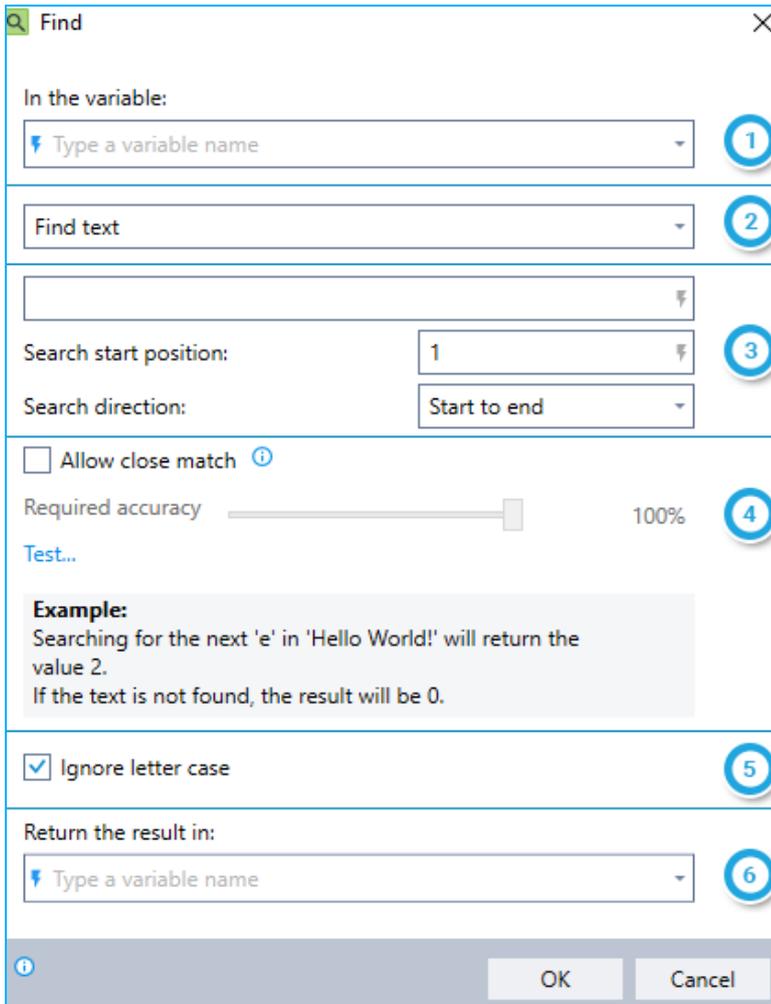
- To learn more about regular expressions, see [this article](#) on the Microsoft Developer Network.
- For an online regex tester and reference, check out this website: [regular expressions 101](#).

Using the FIND command

To find specific text

When you search for the location of a specific text, the wizard 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, the location of the first character in the string will be returned
- If the text you are searching for is not found, the value 0 will be returned



- 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 letter case should be ignored when identifying matching text
 - If unchecked, only text in the same case as the text entered is considered a match
 - When **close match** is allowed, letter case is always ignored
- 6 Enter the name of the variable into which 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:

- The wizard would likely accept the word `c1ose` as matching the word `close` because the number 1 is highly visually similar to a lowercase L.
- The wizard 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.

Find

In the variable:
quote

Find text:
th

Search start position: 1

Search direction: Start to end

Allow close match ⓘ

Required accuracy: 100%

Test...

Example:
Searching for the next 'e' in 'Hello World!' will return the value 2.
If the text is not found, the result will be 0.

Ignore letter case

Return the result in:
find result

OK Cancel

Result: `find result` = 3

To find text matching a regular expression

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

The image shows a 'Find' dialog box with the following fields and callouts:

- 1**: 'In the variable:' dropdown menu with 'Select a variable'.
- 2**: Search type dropdown menu with 'Find text matching a regular expression'.
- 3**: Search type dropdown menu with 'Full match'.
- 4**: 'Pattern:' text input field.
- 5**: 'Match delimiter:' text input field.
- 6**: 'Test...' button.
- 7**: 'Ignore letter case' checkbox (checked).
- 8**: 'Return the result in:' dropdown menu with 'Select a variable'.

Below the fields is an 'Example' box containing the text: 'Finding the regular expression `(?:\d*\.)?\d+` will return all decimal numbers in the given variable.'

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

1 Enter the name of the variable in which you would like to search

2 Select **FIND TEXT MATCHING A REGULAR EXPRESSION**

3 Choose the search type to use:

- **FULL MATCH** is the standard search type: Each text matching the regex pattern is returned, separated by a delimiter. To learn more, see the **FULL MATCH** example below.
- **CAPTURE GROUPS** is a more complex search type: The regex pattern defines a group, which can be further subdivided into individual elements
 - Each group matching the regex pattern is returned, separated by the **GROUP DELIMITER**
 - The individual elements within each group are separated by the **MATCH DELIMITER**

To learn more, see the **CAPTURE GROUPS** example below.

4 Enter the regular expression pattern for which you would like to find matching text

- The regular expression pattern can include:
 - free text and/or variables
 - line breaks

5 Enter the delimiter to separate each matching text found

- If you are using the **CAPTURE GROUPS** search type, you must specify both the **GROUP DELIMITER** and the **MATCH DELIMITER**

Group delimiter:	<input type="text"/>
Match delimiter:	<input type="text"/>

6 **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.

7 Indicate whether letter case should be ignored when identifying matching text

- If unchecked, only text with the same case as that entered will be considered a match

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



EXAMPLE

FULL MATCH search type

Let's say you have copied a large block of text from a web page, and you want 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://regexpr.com/foo.html?q=bar
                https://mediatemple.net
```

The screenshot shows a 'Find' dialog box with the following configuration:

- In the variable:** regex text
- Find text matching a regular expression**
- Search type:** Full match
- Pattern:** \b\d{3}[-]?d{3}[-]?d{4}\b
- Match delimiter:** ;
- Test..** (button)
- Example:** Finding the regular expression (?:\d*\.)?\d+ will return all decimal numbers in the given variable.
- Ignore letter case
- Return the result in:** regex result
- Buttons:** OK, Cancel

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



EXAMPLE

CAPTURE GROUPS search type

Let's say you have copied the HTML code of a table from a web page. Each row represents a different item available for sale. Within each row is a column for the item name and a column for quantity on hand.

You want to extract the data from each row (excluding the table header) as a complete row. You also want to parse the data from each row so that the item name is separated from the quantity.

```
⚡ regex text = <table>
<tr><th>Item</th><th>Quantity</th></tr>
<tr><td>laptop</td><td>15</td></tr>
<tr><td>keyboard</td><td>12</td></tr>
<tr><td>mouse</td><td>5</td></tr>
<tr><td>LCD</td><td>8</td></tr>
</table>
```

The screenshot shows a 'Find' dialog box with the following settings:

- In the variable:** regex text
- Find text matching a regular expression**
- Search type:** Capture groups
- Pattern:** <td>(w*)</td> <td>(d*)</td>
- Group delimiter:** %
- Match delimiter:** ^^
- Test...** (button)
- Example:** Finding the regular expression (?\d*\)\?\d+ will return all decimal numbers in the given variable.
- Ignore letter case
- Return the result in:** regex result
- Buttons:** OK, Cancel

Result:

```
⚡ regex result = laptop^^15%keyboard^^12%mouse^^5%LCD^^8
```

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 [i]

Replace the text matching the regular expression: Test...
[]

Ignore letter case

With:
breathe [i]

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 to place the result



NOTE

If one or both of the values entered is non-numeric, the wizard 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 [Close]

Evaluate the expression:

[Input field] [Info] **1**

Return the result in variable:

[Dropdown: Type a variable name] [Info] **2**

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').

[Info] [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'd like 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 'Fran*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 & 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 21 when ⚡ name = 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 Franklin D. Roosevelt when ⚡ year = 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 Theodore Roosevelt 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	SUBSTRING(<i>expression</i> , <i>start</i> , <i>length</i>)
Arguments	<i>expression</i> = the source string <i>start</i> = the numbered position that the substring starts (within the source string) <i>length</i> = the length of the desired substring
Example	SUBSTRING(\$trivia fact\$, 13, 28) returns Roosevelt died in April 1945 when <code>⚡ trivia fact</code> = Franklin D. Roosevelt died in April 1945, before the end of WWII.

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 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

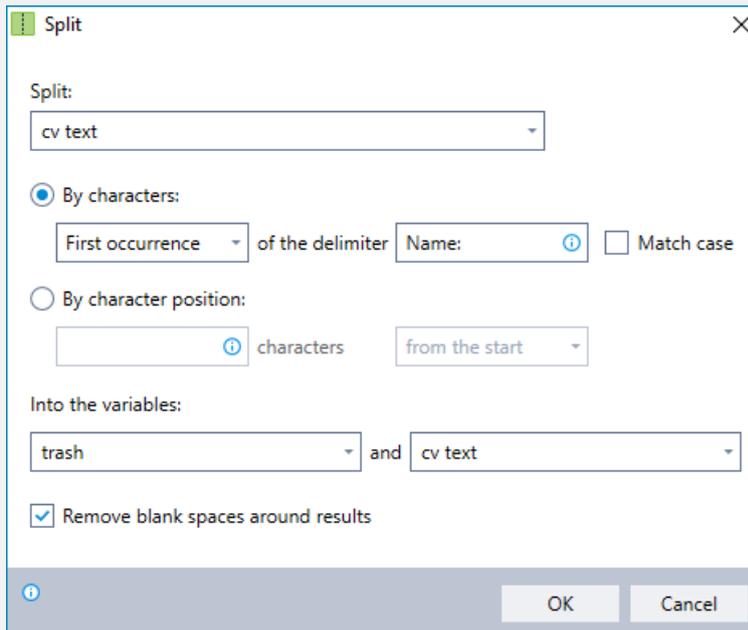
**EXAMPLE****Extracting information from a long text file**

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 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.
```

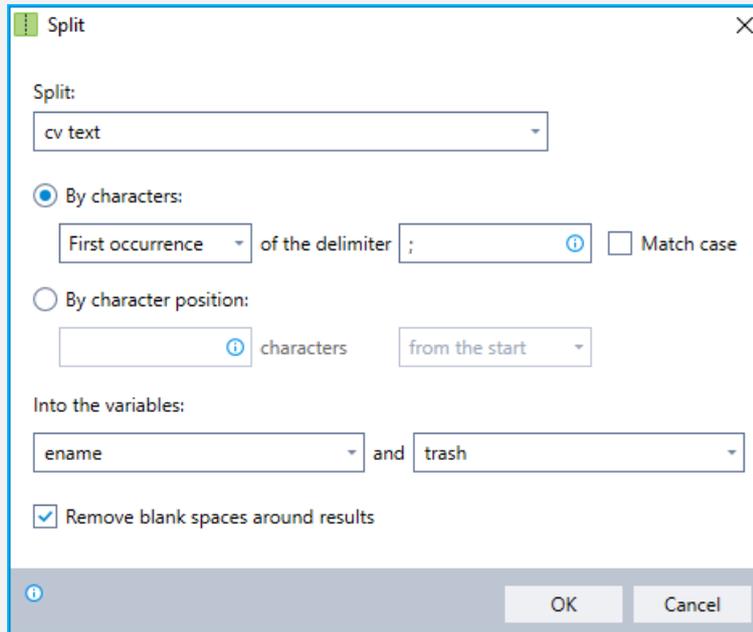


Result:

```
⚡ 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 text input field containing "Type a variable name".
- Section 2:** Three radio button options:
 - "Where items are delimited by:" followed by a text input field.
 - "Where items are preceeded 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 text input field containing "Type a variable name".
- Section 5:** "Example:" followed by text: "For the array *first,second,third,fourth*, where items are delimited by commas, the total item number is 4 and the value of item #2 is *second*." Below this is a dropdown menu for "Error handling".

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

- 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 to place the data obtained
- 5 Instruct the wizard 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, the wizard 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

The screenshot shows a dialog box titled "Get array data" with the following configuration:

- From the array stored in:** dropdown menu containing "daily orders".
- Where items are delimited by:** radio button selected, with a text input field containing ";" and an information icon.
- Where items are preceeded by:** radio button unselected, with an empty text input field and an information icon.
- and followed by:** radio button unselected, with an empty text input field and an information icon.
- Get the value of item #:** radio button unselected, with an empty text input field and an information icon.
- Get the total number of items:** radio button selected.
- Get the maximum value:** radio button unselected.
- Return the result in variable:** dropdown menu containing "number of orders".

At the bottom of the dialog are "OK" and "Cancel" buttons, along with an information icon.

Result: `⚡ number of orders` = 7

Get array data

From the array stored in:
daily orders

Where items are delimited by: ;

Where items are preceded 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 preceded 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
- All the values in a specific row
- All the values in a specific column



TIP

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

Using the GET TABLE DATA command

1 Enter the name of the variable in which the table is stored

2 Enter the delimiters that separate each column and row



Select the type of data to retrieve and provide additional information as required (varies by the type of data to retrieve)

<p>Data to get:</p> <p>Single value</p> <p>Column number: <input type="text"/></p> <p>Row number: <input type="text"/></p>	<p>Single value:</p> <p>The wizard will retrieve the value of the item at a specific location</p> <ul style="list-style-type: none"> Enter the column number and row number of the item to be retrieved
<p>Data to get:</p> <p>Number of items</p>	<p>Number of items:</p> <p>The wizard will retrieve the total number of items in the table</p> <ul style="list-style-type: none"> No additional information is required
<p>Data to get:</p> <p>Number of rows</p>	<p>Number of rows:</p> <p>The wizard will retrieve the total number of rows in the table</p> <ul style="list-style-type: none"> No additional information is required
<p>Data to get:</p> <p>Number of columns</p>	<p>Number of columns:</p> <p>The wizard will retrieve the total number of columns in the table</p> <ul style="list-style-type: none"> No additional information is required
<p>Data to get:</p> <p>Entire row</p> <p>Row number: <input type="text"/></p>	<p>Entire row:</p> <p>The wizard will retrieve all the values in a specific row</p> <ul style="list-style-type: none"> Specify the row number of the values to be retrieved NOTE: The values returned will be separated by the column delimiter you specified in
<p>Data to get:</p> <p>Entire column</p> <p>Column number: <input type="text"/></p>	<p>Entire column:</p> <p>The wizard will retrieve all the values in a specific column</p>

	<ul style="list-style-type: none"> Specify the column number of the values to be retrieved NOTE: The values returned will be separated by the row delimiter you specified in 2
--	---

4 Enter the name of the variable into which 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.

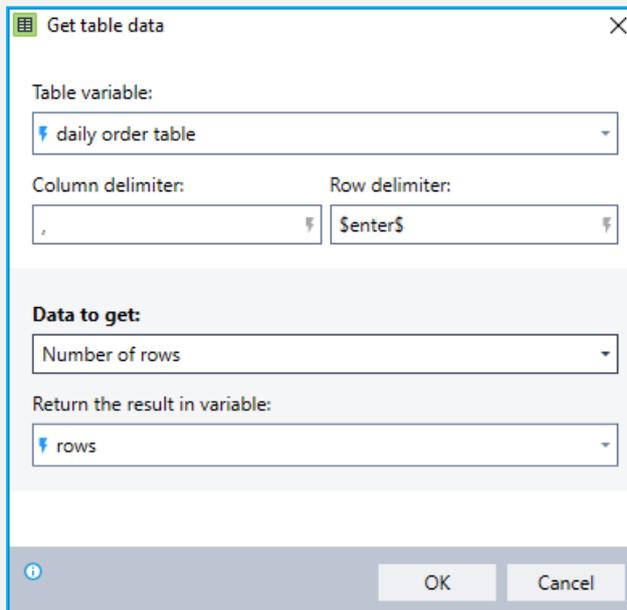
Here is how the data would look formatted as a table:

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
```

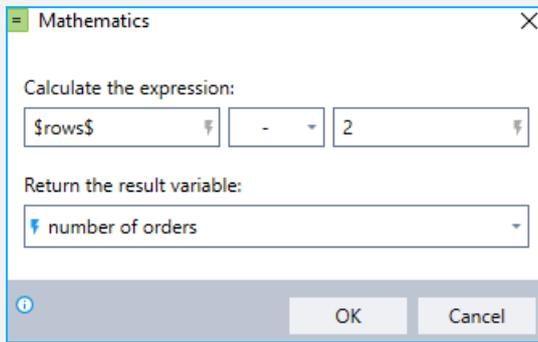
1. Obtain the total number of rows in the table



Result: ⚡ 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

Table Lookup

Search for a value in a table (the "lookup value") and retrieve a corresponding value. You can choose to retrieve the following types of values:

- The location of the lookup value (identified by its row and column numbers)
- All the values in the lookup value's row
- All the values in the lookup value's column
- A single value from a specific column in the lookup value's row
- A single value from a specific row in the lookup value's column

Using the TABLE LOOKUP command

The screenshot shows the 'Table lookup' dialog box with the following fields and callouts:

- 1**: 'From the table stored in:' dropdown menu.
- 2**: 'Column delimiter:' and 'Row delimiter:' input fields.
- 3**: 'Find:' dropdown menu.
- 4**: 'equals (ignore case)' dropdown menu.
- 5**: Input field for the lookup value.
- 6**: Radio buttons for 'Search the entire table', 'Search column:', and 'Search row:'.
- 7**: 'Return:' dropdown menu and 'Column #:' and 'Row #:' input fields.
- 8**: 'Error handling' dropdown menu.

- 1 Enter the name of the variable in which the table is stored
- 2 Enter the delimiters that separate each column and row
- 3 Choose whether to look for:
 - a. A single value matching the lookup criteria specified in **4**, **5** & **6**
 - b. All values that match the lookup criteria specified in **4**, **5** & **6**

NOTE: If you select option **a** (a single value) and more than one matching value is found, the **multiple lookup values found** error will be returned in the **error variable** and the output variable will be returned as empty.

4

Select the condition for finding the lookup value:

- equals (with options to ignore letter case/use wildcards/**allow close match**)
- contains (with option to ignore letter case/**allow close match**)
- 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

5

Enter the lookup value

6

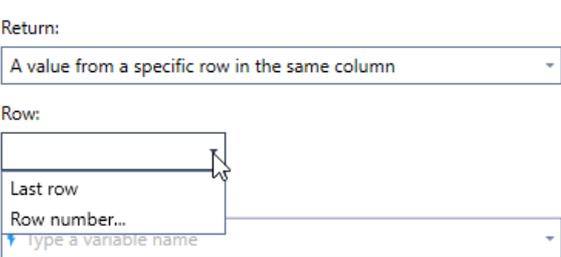
Select whether to search for the lookup value in the entire table, in a specific column, or in a specific row; **and**

Provide the column/row number in which to search (where relevant)



Select the value to return and provide additional information as required (varies by the type of value to return)

<p>Return:</p> <p>Address (column & row)</p> <p>Column #: <input type="text" value="Type a variable name"/></p> <p>Row #: <input type="text" value="Type a variable name"/></p> <p>Return:</p> <p>Entire row</p> <p>Output variable: <input type="text" value="Type a variable name"/></p> <p>Return:</p> <p>Entire column</p> <p>Output variable: <input type="text" value="Type a variable name"/></p> <p>Return:</p> <p>A value from a specific column in the same row</p> <p>Column:</p> <p><input type="text" value="Last column"/></p> <p><input type="text" value="Column number..."/></p> <p><input type="text" value="Type a variable name"/></p>	<p>Address (column & row):</p> <p>The wizard will return the location of the lookup value</p> <ul style="list-style-type: none"> Enter the name(s) of the variable(s) in which to return the value(s) retrieved <ul style="list-style-type: none"> If you have selected option a in 3 (a single matching value), specify separate variables for returning column and row numbers If you have selected option b in 3 (multiple matching values), specify a single output variable <p>Entire row:</p> <p>The wizard will return all the values in the lookup value's row</p> <ul style="list-style-type: none"> Enter the name of the variable in which to return the values retrieved <p>Entire column:</p> <p>The wizard will return all the values in the lookup value's column</p> <ul style="list-style-type: none"> Enter the name of the variable in which to return the values retrieved <p>A value from a specific column in the same row:</p> <ul style="list-style-type: none"> Select whether you want the wizard to return: <ul style="list-style-type: none"> a. The value from the last column in the lookup value's row; or
---	--

 <p>Return:</p> <p>A value from a specific row in the same column</p> <p>Row:</p> <p>Last row</p> <p>Row number...</p> <p>type a variable name</p>	<ul style="list-style-type: none">b. The value from a specific column in the lookup value's row (and enter the column number) NOTE: This should be the actual column number, NOT the column's position relative to the lookup value <ul style="list-style-type: none">• Enter the name of the variable in which to return the value(s) retrieved <p>A value from a specific row in the same column:</p> <ul style="list-style-type: none">• Select whether you want the wizard to return:<ul style="list-style-type: none">a. The value from the last row in the lookup value's column; orb. The value from a specific row in the lookup value's column (and enter the row number) NOTE: This should be the actual row number, NOT the row's position relative to the lookup value <ul style="list-style-type: none">• Enter the name of the variable in which to return the value(s) retrieved
---	---

<p>Return: <input type="text" value="The matching value itself"/></p> <p>Output variable: <input type="text" value="Select a variable"/></p>	<p>The matching value itself:</p> <p>The wizard will return the matching value(s) that meet the lookup criteria</p> <ul style="list-style-type: none">• Enter the name of the variable in which to return the values retrieved <p>EXAMPLE: When would you use this option?</p> <p>Returning the matching value itself is useful when you want to retrieve:</p> <ul style="list-style-type: none">• all values that match the lookup condition; and• the lookup condition is something other than <code>equals</code> <p>Say, for example, your table contains transaction data, and you want to find the amounts all transactions greater than \$10,000. You could set this command to:</p> <ul style="list-style-type: none">• find all matching values greater than \$10,000; and• return the matching value itself
--	---

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



NOTE

When relevant, values retrieved will be separated using the row and column delimiters you specified in .

Get ASCII Character

There are times when you need a wizard 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

- 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
 - Note that extended ASCII codes are not supported
- 2 Enter the name of the variable into which you'd like to place the character



EXAMPLE

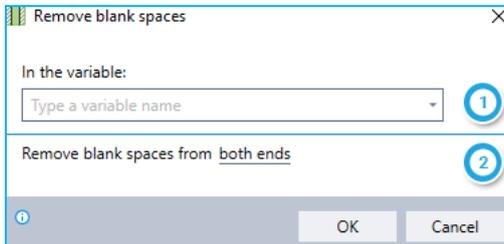
At the end of each day, you download data from a web application for further analysis. The web application uses <Page Break> to separate individual records, so you also need to separate the data at each <Page Break>. How can you make this happen?

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



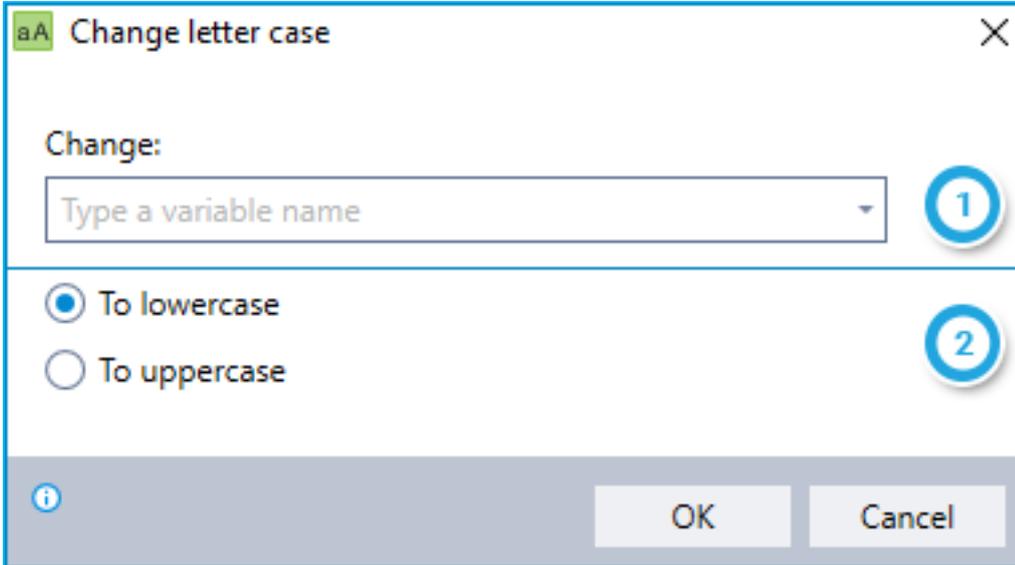
TIP

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 Choose 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

The screenshot shows a dialog box titled "Check type" with a close button (X) in the top right corner. The dialog is divided into two main sections. The first section is labeled "Check if" and contains a dropdown menu with the text "Type a variable name" and a small downward arrow, followed by the word "is", and another dropdown menu with the text "numeric" and a small downward arrow. A blue circle with the number "1" is positioned to the right of this section. The second section is labeled "Return the result (TRUE/FALSE) in variable:" and contains a dropdown menu with the text "Type a variable name" and a small downward arrow. A blue circle with the number "2" is positioned to the right of this section. At the bottom of the dialog, there is an information icon (i) on the left, and two buttons labeled "OK" and "Cancel" on the right.

- 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 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

Get random number

Get a random number between: and 1

Return the result in variable:

2

OK Cancel

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

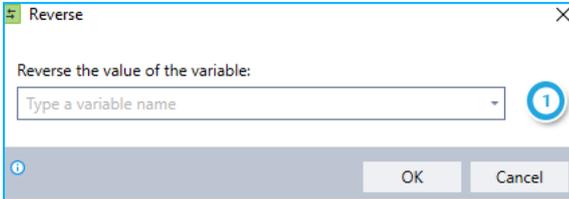
Creating single-use login codes for other applications

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

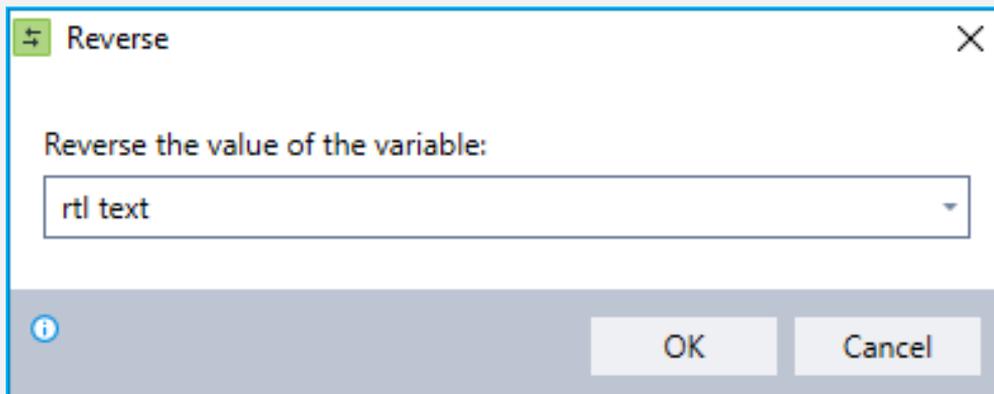


- 1 Enter the name of the variable whose value you'd like to reverse



EXAMPLE

⚡ rtl text = ydenneK .F nhoJ

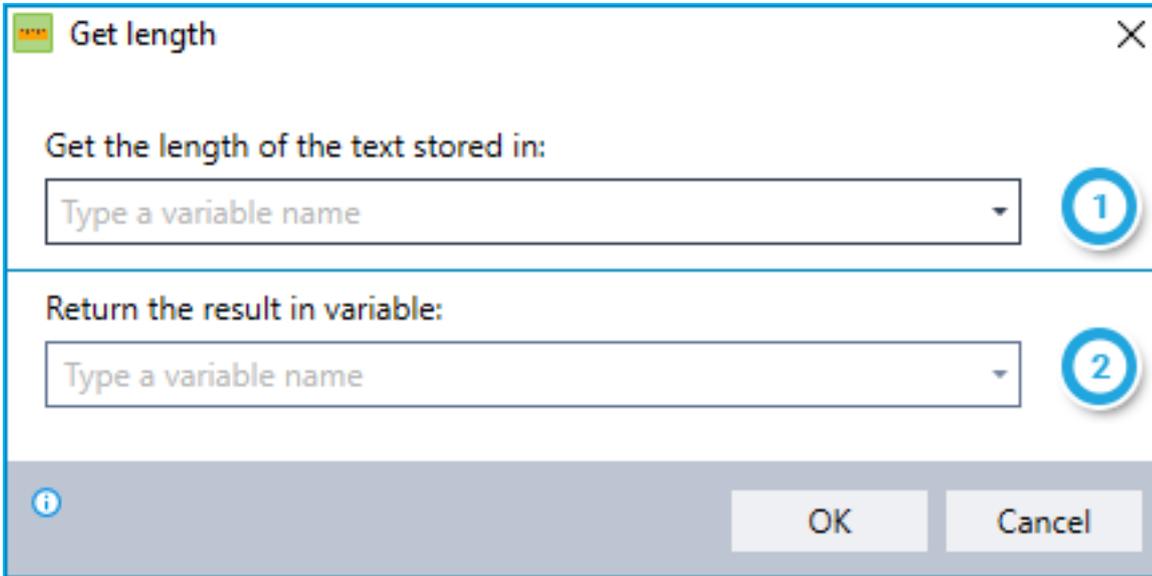


Result: **⚡** rtl text = John F. Kennedy

Get Length

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

Using the GET LENGTH command



The screenshot shows a dialog box titled "Get length" with a close button in the top right corner. The dialog is divided into two main sections. The first section is labeled "Get the length of the text stored in:" 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 "Return the result in variable:" 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 on the left, and "OK" and "Cancel" buttons on the right.

- 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 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 to place the extracted numbers
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

**NOTE****How exactly are numbers extracted?**

The wizard 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 (|)

The wizard 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, the wizard 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	59
Complex If Else	63
Multi-Value If Else	65
Loop	67
Loop: Break	69
Loop: Restart	70
Loop Table	71
Loop Items	73
Pause	75
Group	76

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

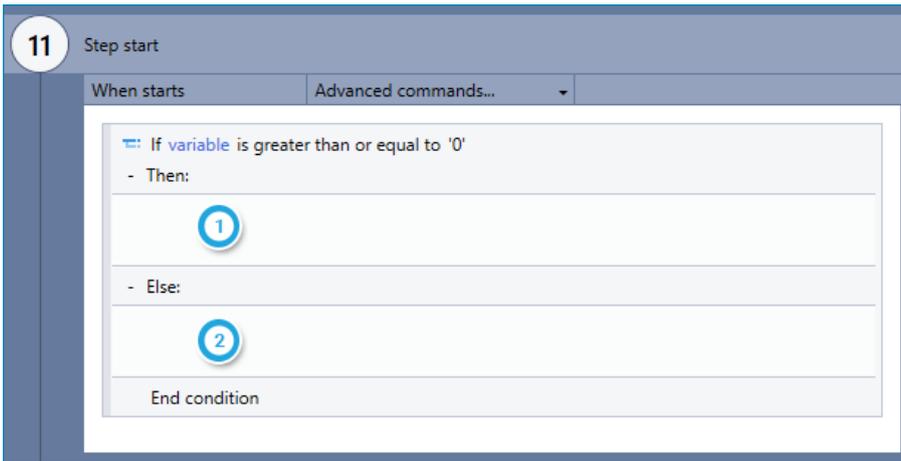
The first step in using the **IF ELSE** command is to define the condition (i.e., set up a comparison).

- 1 Enter the name of the variable whose value you wish to compare with another value
- 2 Select the type of comparison to perform:
 - equals (with options to ignore letter case/use wildcards/allow close match)
 - contains (with option to ignore letter case/allow close match)
 - 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



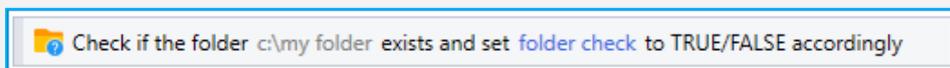
EXAMPLE

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

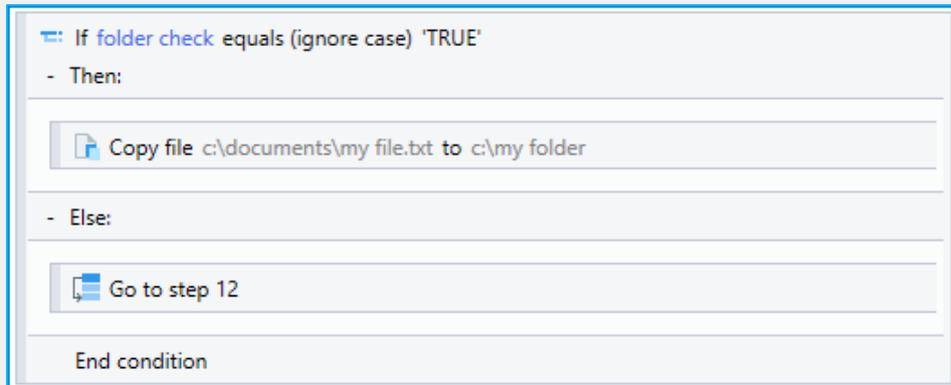
- If it does (the condition is TRUE), you want to copy a file to it
- If it doesn't (the condition is FALSE), you want to skip to the next step

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



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



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

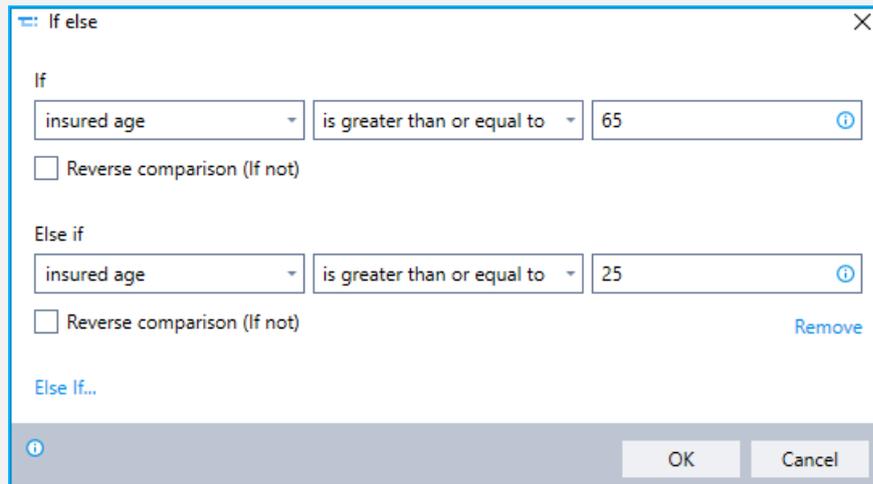


EXAMPLE

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

- 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).

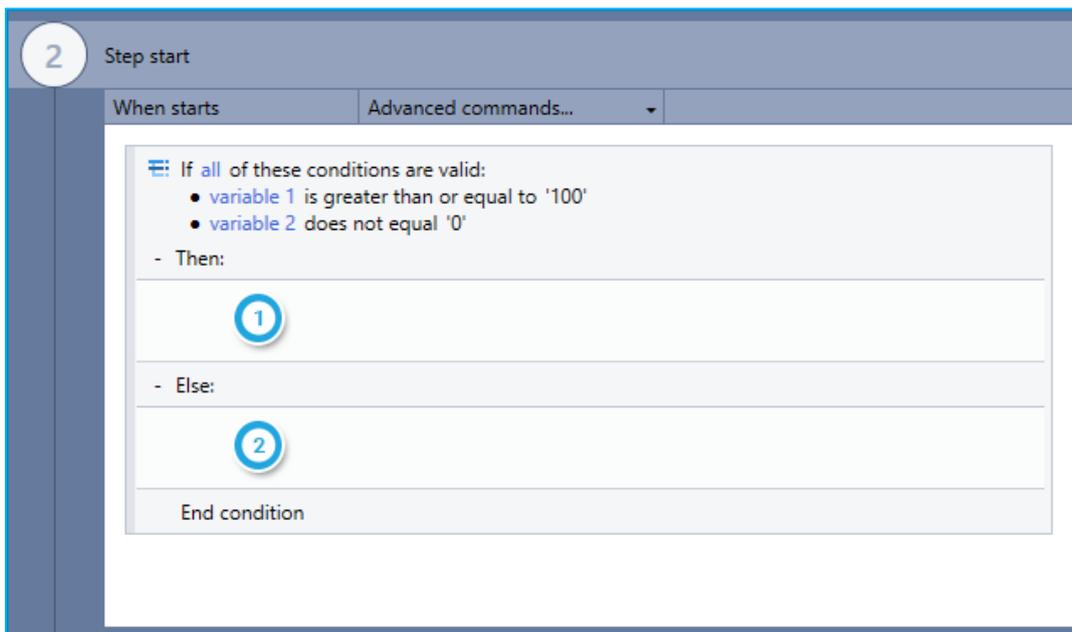
- 1 Choose whether **any** or **all** of the conditions must be true in order for the overall outcome to be TRUE
- 2 For each condition:
Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
 - equals (with options to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

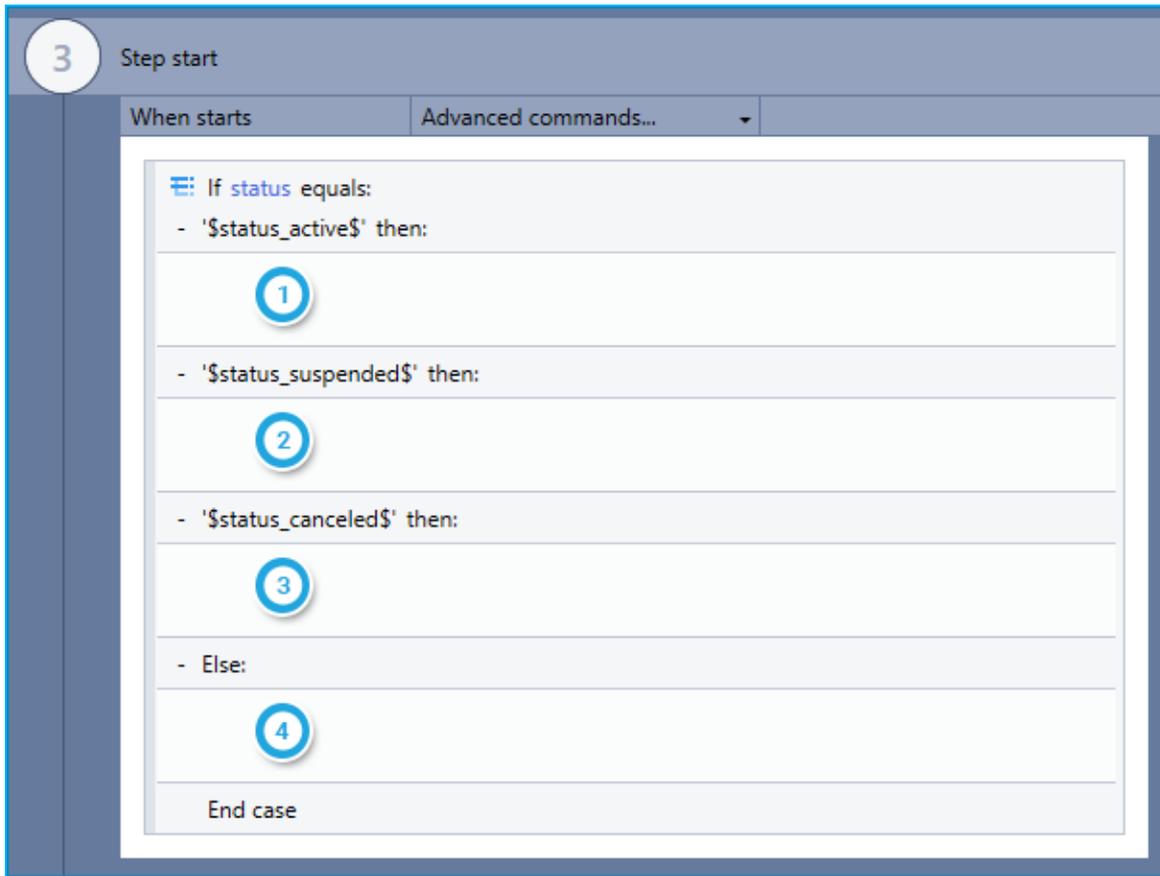
Step #1 - Define the conditions

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
 - 3a Add...
 - 3b Remove

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).

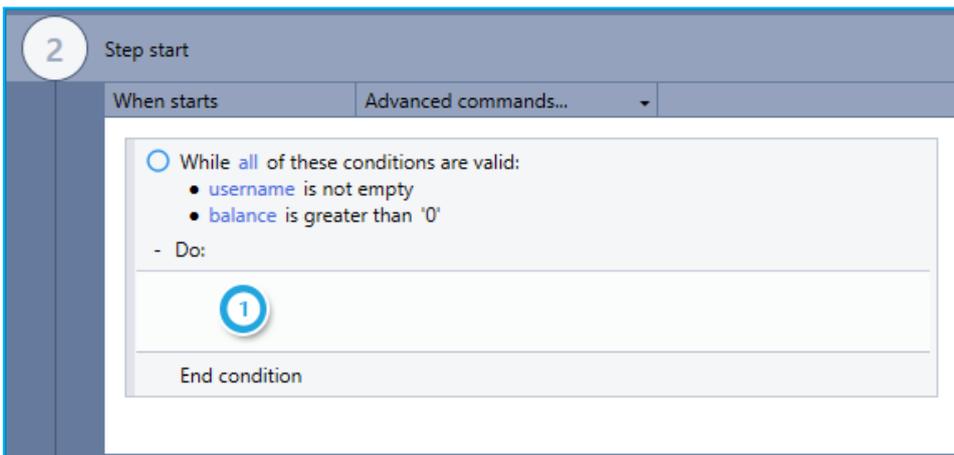
- 1 Choose whether **any** or **all** of the conditions must be true in order for the loop to continue
- 2 For each condition:
 - Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
 - equals (with options to ignore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



TIP

Break that loop!

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

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).

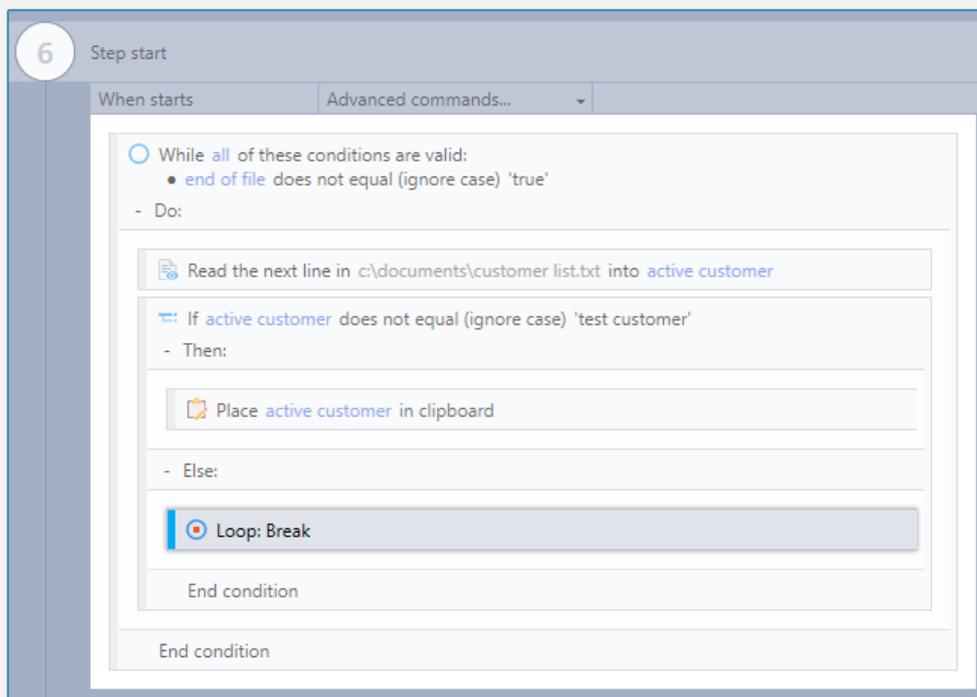


NOTE

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

- **LOOP**
- **LOOP TABLE**
- **LOOP ITEMS**
- **GET EMAIL MESSAGES**
- **ANALYZE DIGITAL PDF FILE**
- **OCR: DOCUMENTS**

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.

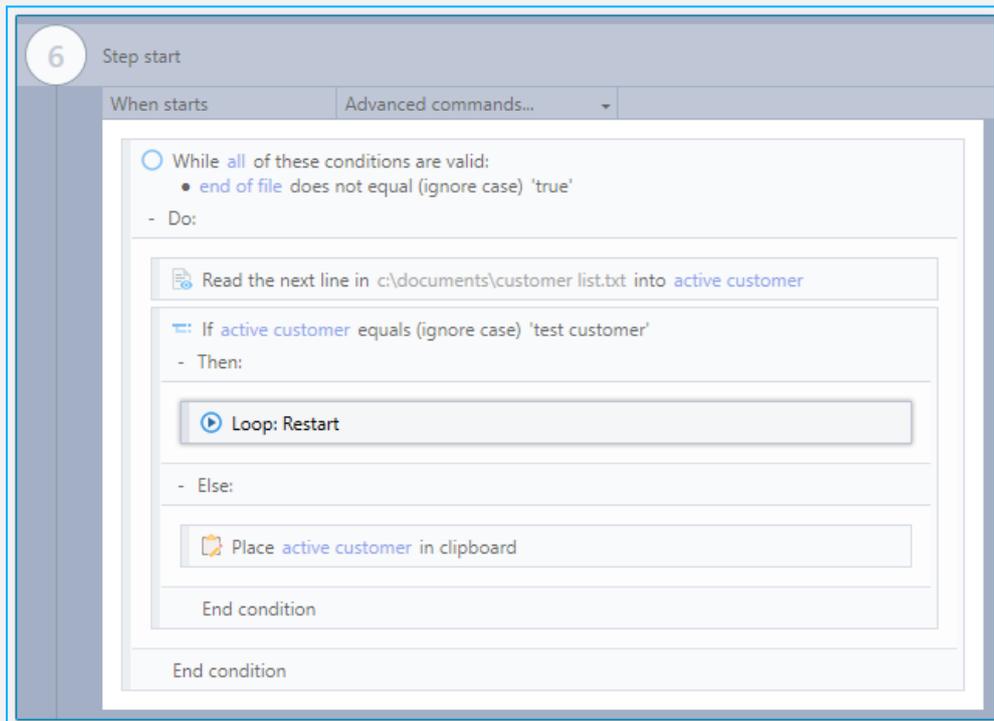


NOTE

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

- **LOOP**
- **LOOP TABLE**
- **LOOP ITEMS**
- **GET EMAIL MESSAGES**
- **ANALYZE DIGITAL PDF FILE**
- **OCR: DOCUMENTS**

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 Table

For each row of a table stored in a variable:

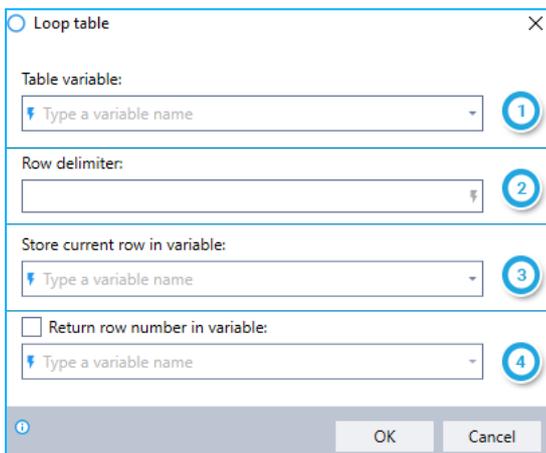
- Place the current row into a new or existing variable
- Optionally place the row number into a new or existing variable; **and**
- Perform a specified action or series of actions

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

Using the LOOP TABLE command

Step #1 - Identify the table

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



The screenshot shows a dialog box titled "Loop table" with a close button (X) in the top right corner. It contains four input fields, each with a numbered callout (1-4) to its right:

- 1. "Table variable:" followed by a dropdown menu containing "Type a variable name".
- 2. "Row delimiter:" followed by a text input field.
- 3. "Store current row in variable:" followed by a dropdown menu containing "Type a variable name".
- 4. "Return row number in variable:" (with an unchecked checkbox) followed by a dropdown menu containing "Type a variable name".

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

1 Enter the name of the variable in which the table is stored

2 Enter the delimiter that separates each row of the table

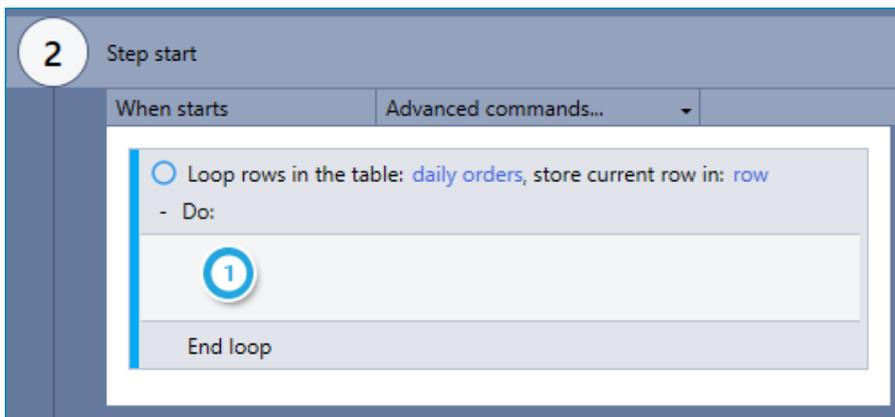
3 Enter the name of the variable into which you'd like to place the row

Why? Since the action(s) in the loop will be performed on each row of the table (one at a time), it makes sense to first place the row into a variable, then perform the defined actions on the value of that variable.

4 Indicate whether to place the current row number in a variable; **and** if so, enter the name of the variable in which to place it

Step #2 - Define the actions

Upon adding the **LOOP TABLE** 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 row in the table

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



TIP

Break that loop!

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

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:
Type a variable name 1

Where items are delimited by:
2

Set each item in the variable:
Type a variable name 3

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

OK Cancel

1 Enter the name of the variable in which the array is stored

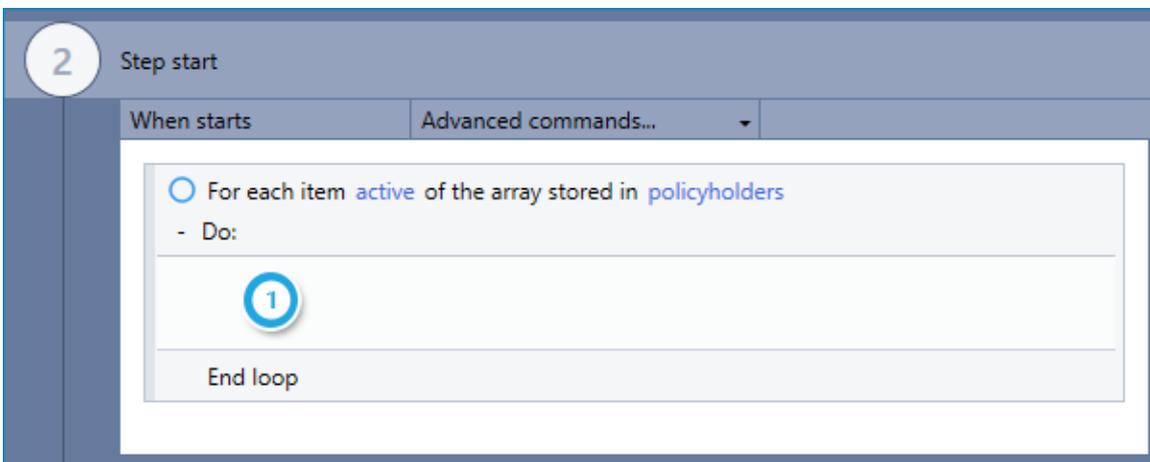
2 Enter the delimiter that separates each item in the array

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



TIP

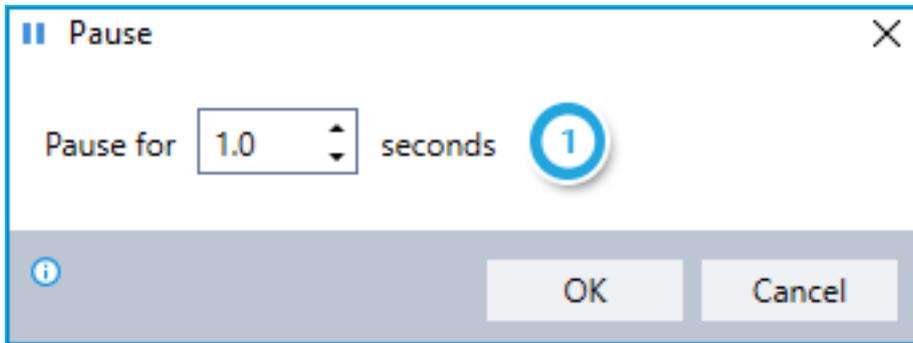
Break that loop!

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

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)

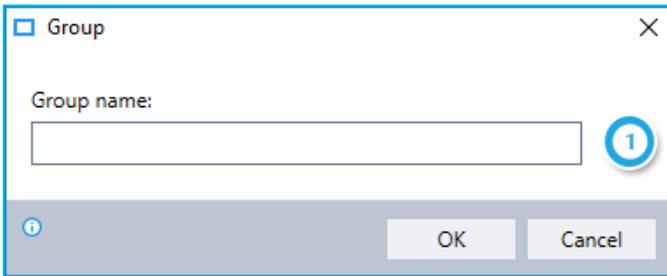
Group

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

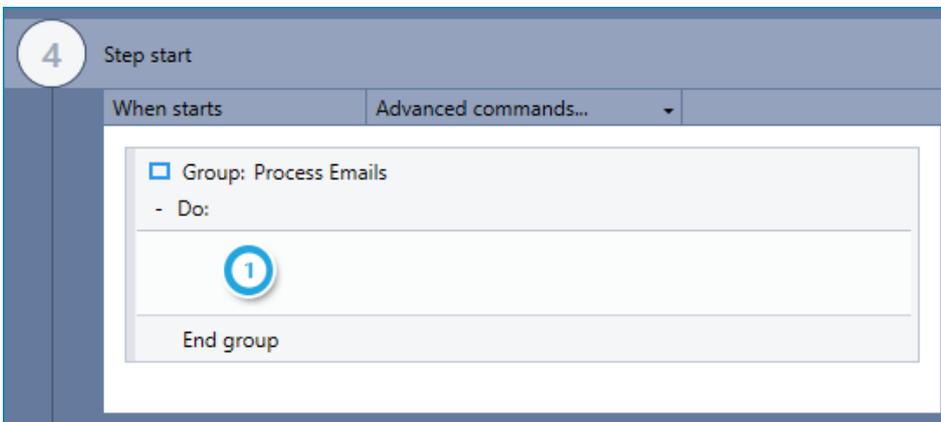
Step #1 - Name the group



- 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.

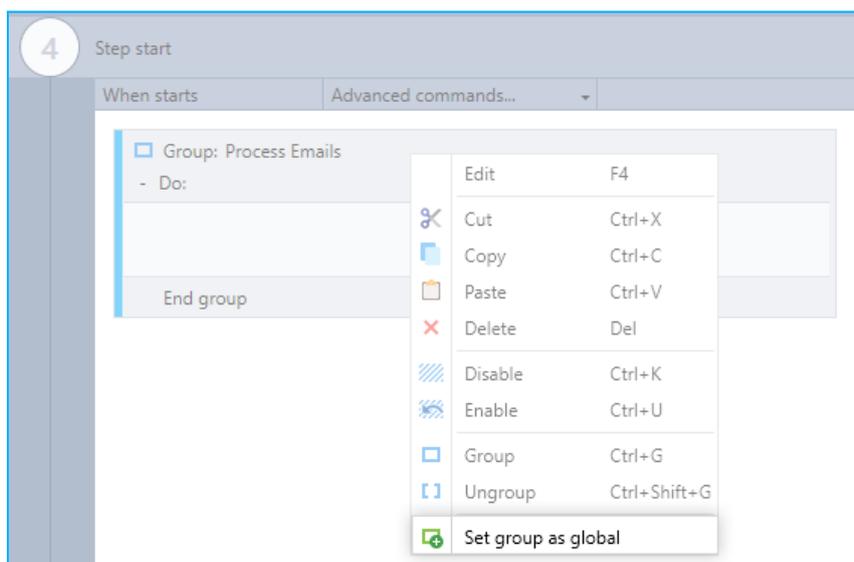


NOTE

A global group is available within 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 make changes to 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	80
End Wizard	81
Go To Step	82
Get Step Data	84
Get Wizard Data	85
Check Application	86
Get User Data	87
Check Run Mode	88
Check Video Recording Mode	89
Set User Interrupt Mode	90
Show Message	91
Raise Wizard Error	93
Get Last Failure Type	94
Resume Error	95
Report Wizard Output	96

Continue Wizard

Exit the current group of commands, the logical flow, or the step section, and continue playing the wizard.

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

How does it work?

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

- Group of commands within another group of commands → the wizard will skip outside of the inner group to the outer group
- Group of commands → the wizard will skip outside of the group to the same step
- 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.

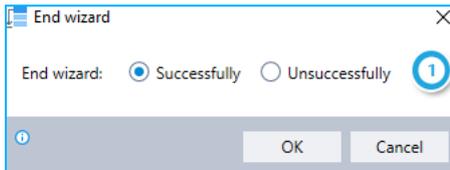
End Wizard

End the wizard immediately. The wizard will skip:

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

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:

1. **Global Actions: WIZARD END** actions can be defined differently based on whether the wizard ends successfully or unsuccessfully
2. **Kryon Report Generator:** Successful/unsuccessful wizard end is differentiated 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 the wizard how to handle any errors encountered. Read more about [Error Handling](#).



CAUTION

Watch the variable value!

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.



CAUTION

Going to steps marked *Do not play*

Note the behavior of the **GO TO STEP** command when the **destination step** is marked *Do not play*:

- When the destination step is identified by **step number/name**, the wizard will end
 - Wizard status will show that wizard ended successfully
- When the destination step is identified by **variable**, the wizard will proceed to that step, skip it (as instructed), and then continue by proceeding to the next step in the flow

Best practice, of course, is to ensure that **GO TO STEP** commands do not point to destination steps marked *Do not play*.

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



CAUTION

Take care with embedded wizards!

Note that the result of this advanced command will differ a bit when it is used for a step in which the core action is an embedded wizard:

- When the command is used in the **Step start** section of the step, the result returned will be the step number/name of the **containing wizard**
- When the command is used in the **Step end** section of the step, the result returned will be the step number/name of the last executed step of the **embedded wizard**

Example:

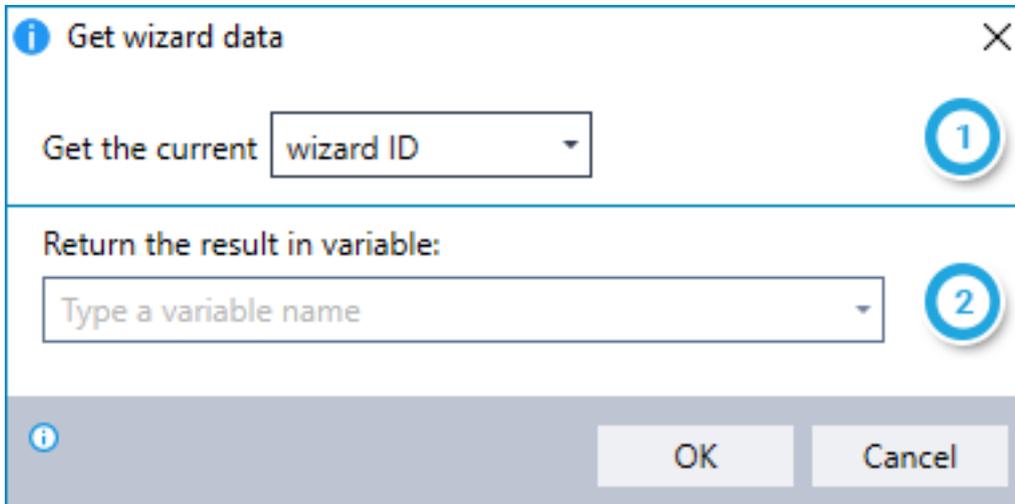
The core action of Step 3 of your wizard (let's call it *Main Wizard*) is an embedded wizard (let's call it *Sub Wizard*). *Sub Wizard* contains 6 steps. You place **GET STEP DATA** Step 3 of *Main Wizard*.

- If you place the command in **Step start**, the result will be 3
- If you place the command in **Step end**, the result will be 6 (assuming Step 6 of *Sub Wizard* was actually executed)

Get Wizard Data

Retrieve the ID or name of the current wizard and place the result into a new or existing variable.

Using the GET WIZARD DATA command



- 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 Application

Check whether the current wizard is being run from Kryon Robot or Kryon Studio.

Using the CHECK APPLICATION command

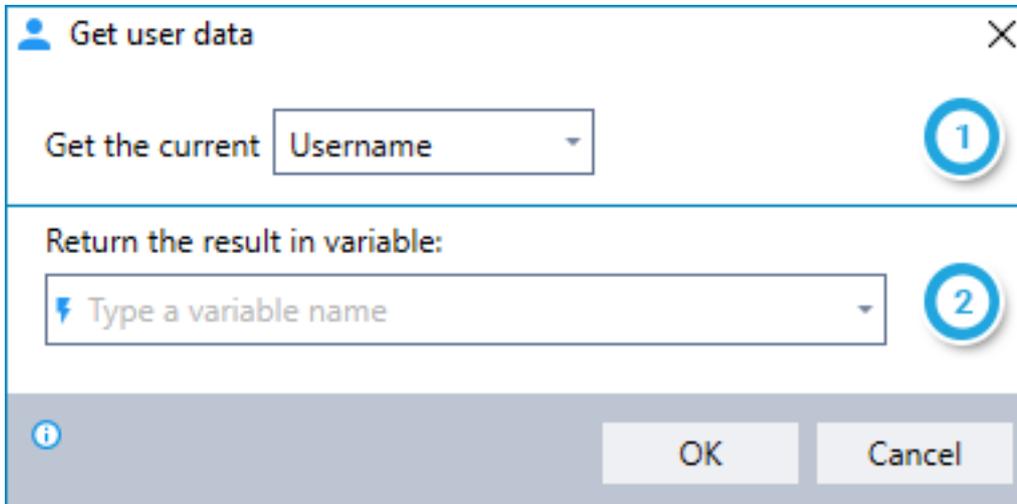
The screenshot shows the 'Call web service method' dialog box. It has a title bar with a close button. Below the title bar is a header area with the text 'Enter values manually or use the Service Discovery tool' and a 'Discover Services...' button. Below this is a 'Web Service URL:' text box. Underneath are radio buttons for 'Call Method' (selected) and 'Send SOAP xml'. To the left are dropdown menus for 'Protocol:' (set to 'HttpPost'), 'Service:', and 'Method:'. Below these are radio buttons for 'Credentials': 'No credentials' (selected), 'Manual', and 'From vault'. To the right is a 'Parameters:' table with columns 'Name', 'Type', and 'Value'. The table has one row with a '+' icon and the text 'Enter new parameter name'. Below the table is a 'Return result in variable:' dropdown menu with the text 'Type a variable name'. To the right of this is a 'Service timeout:' spinner set to '30' seconds. At the bottom left is an 'Error handling' section with a dropdown arrow and an info icon. At the bottom right are 'OK' and 'Cancel' buttons. Four blue circles with numbers 1 through 4 are overlaid on the dialog: 1 is on the 'Discover Services...' button, 2 is on the 'Web Service URL:' text box, 3 is on the 'Service:' dropdown menu, and 4 is on the 'Parameters:' table.

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

Get 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 USER DATA command



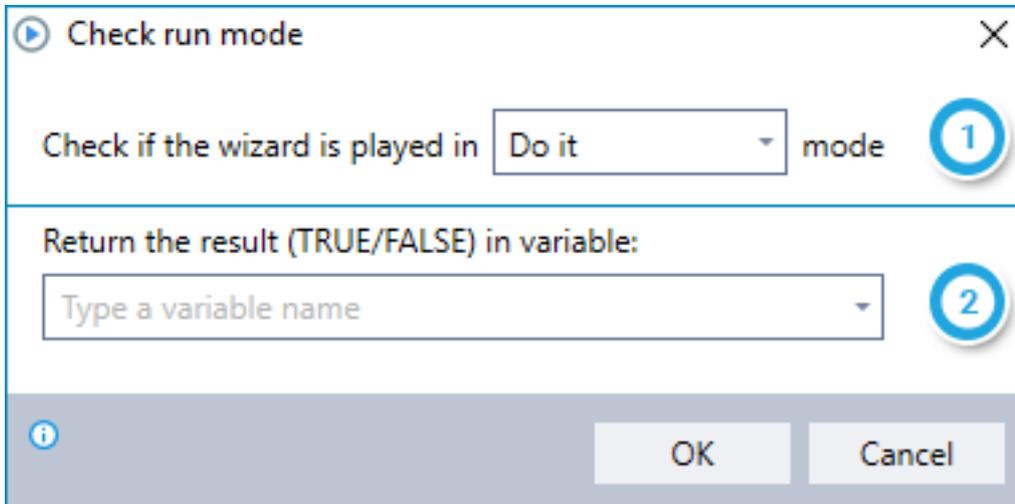
The screenshot shows a dialog box titled "Get user data". It has a close button in the top right corner. The dialog is divided into two main sections. The first section is labeled "Get the current" and contains a dropdown menu with "Username" selected. A blue circle with the number "1" is positioned to the right of this dropdown. The second section is labeled "Return the result in variable:" and contains a text input field with the placeholder text "Type a variable name". A blue circle with the number "2" is positioned to the right of this input field. At the bottom of the dialog, there is an information icon on the left and "OK" and "Cancel" buttons on the right.

- 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 to place the result

Check Run Mode

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

Using the CHECK RUN MODE command



Check run mode

Check if the wizard is played in mode **1**

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

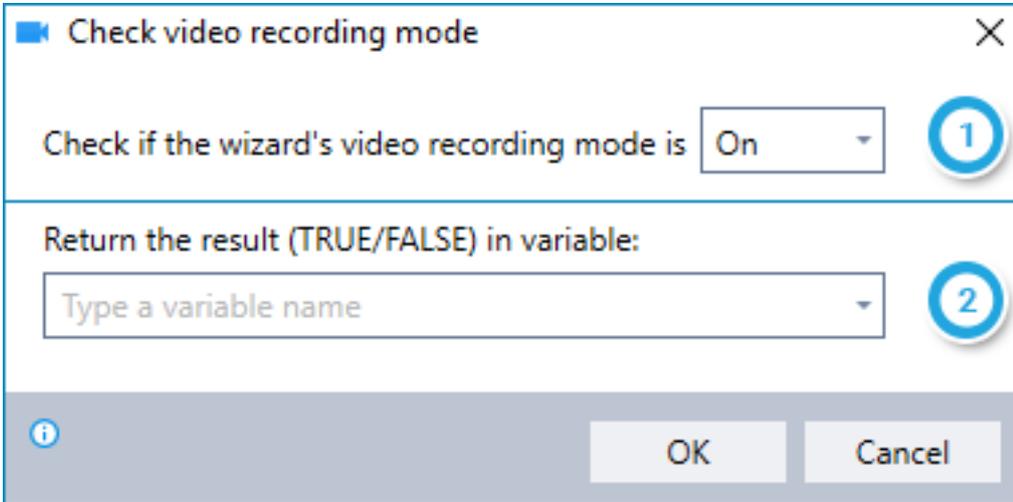
i OK Cancel

- 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 Kryon'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

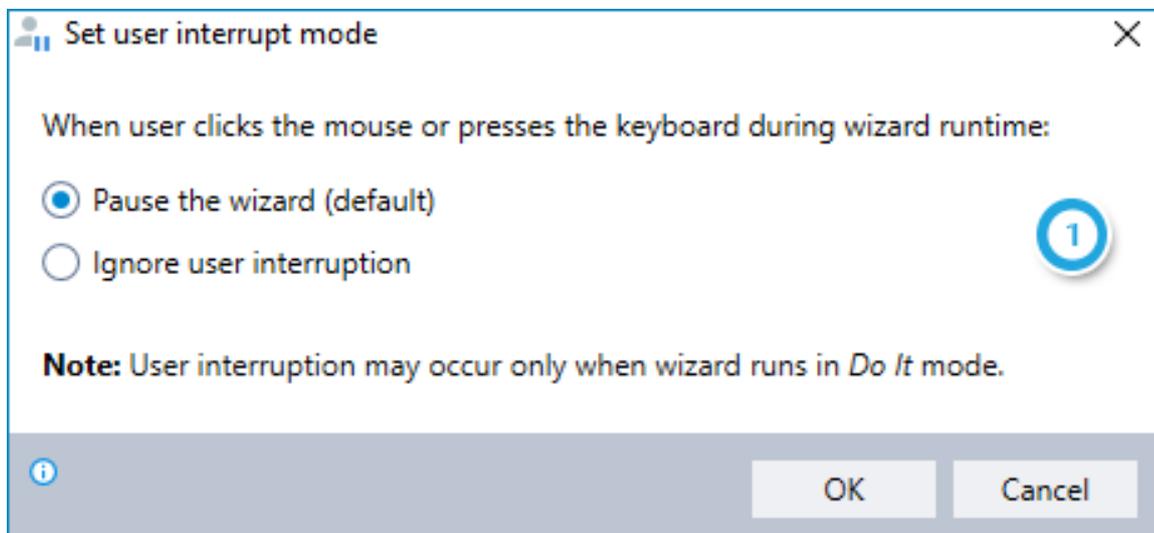
Select whether or not to pause a wizard when the end user clicks the mouse or uses the keyboard while it's running.



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

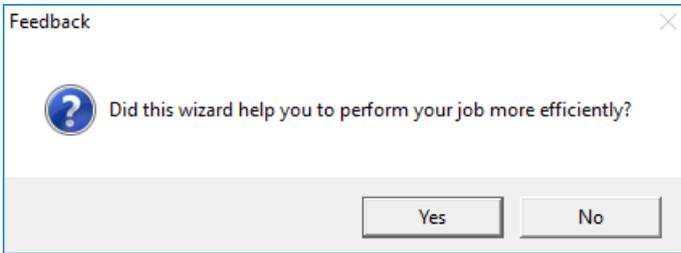


Choose what to do if the user clicks the mouse or keyboard while the wizard is running, either:

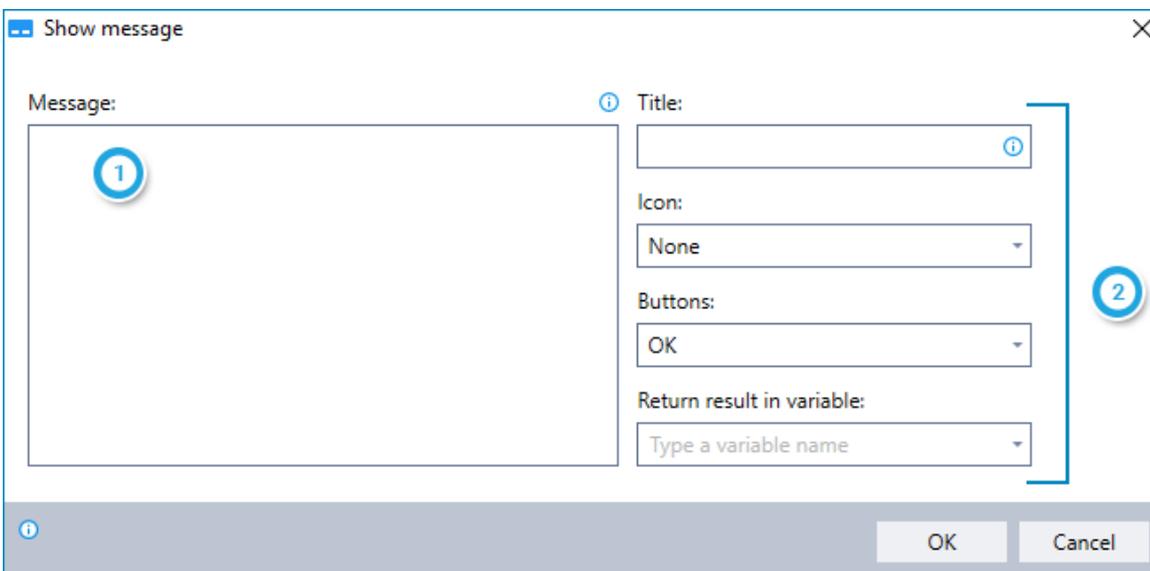
- Pause the wizard (allow user interruption); or
- Continue playing the wizard (ignore user interruption)

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?

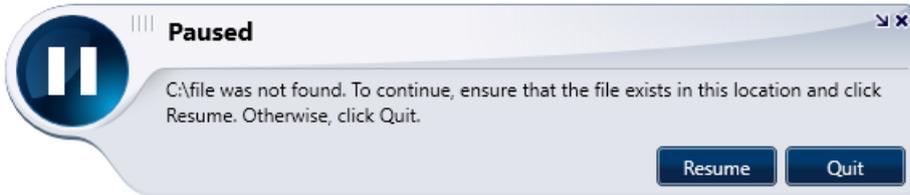
While placing the result of **SHOW MESSAGE** into a variable is optional, there are times when it can be particularly useful:

- 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

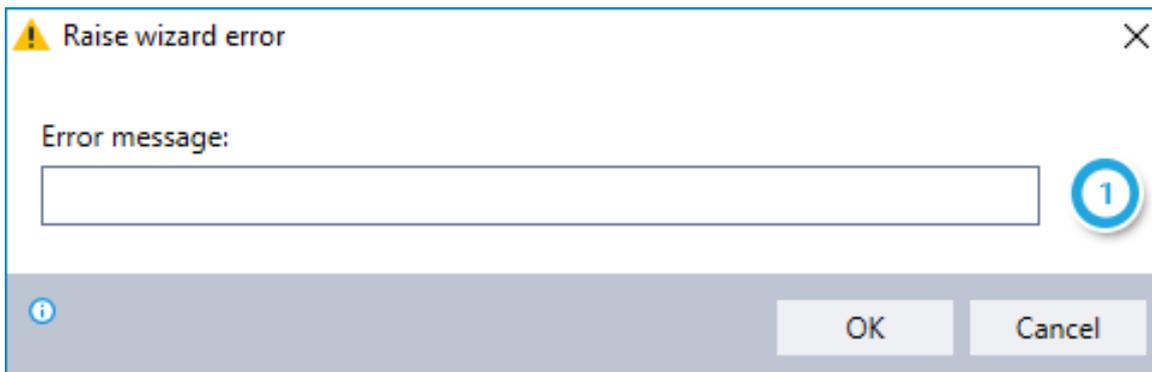
- Pause the currently running wizard;
- Display a short customized message to the end user; **and**
- Give the user the option to resume or quit the wizard



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.

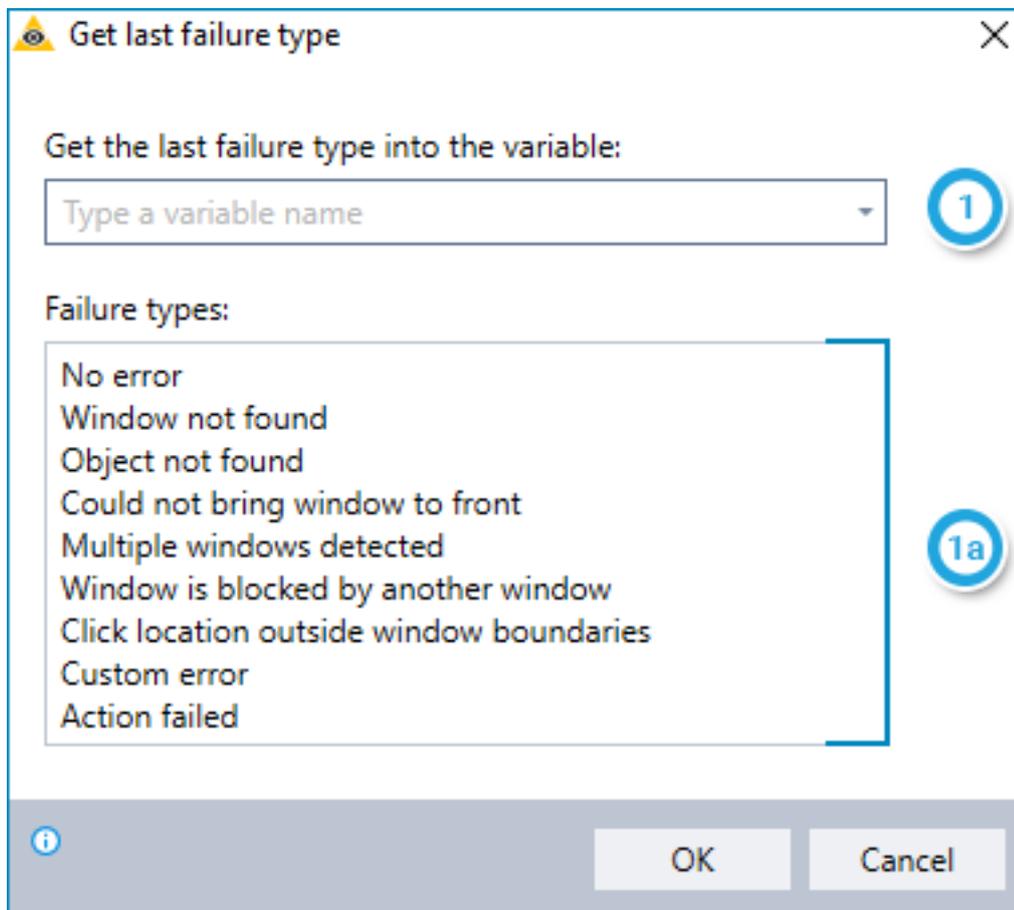
Recommended: The message should instruct the user how to correct the error before he clicks **RESUME**.

Get Last Failure Type

Designed to work in conjunction with Kryon'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 Kryon Studio User Guide.
- For more information about global error handling, see the *Global Actions* section of the Kryon 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 values that could be returned. (No need for you to do anything with this section... it's there just for your information.)

Resume Error

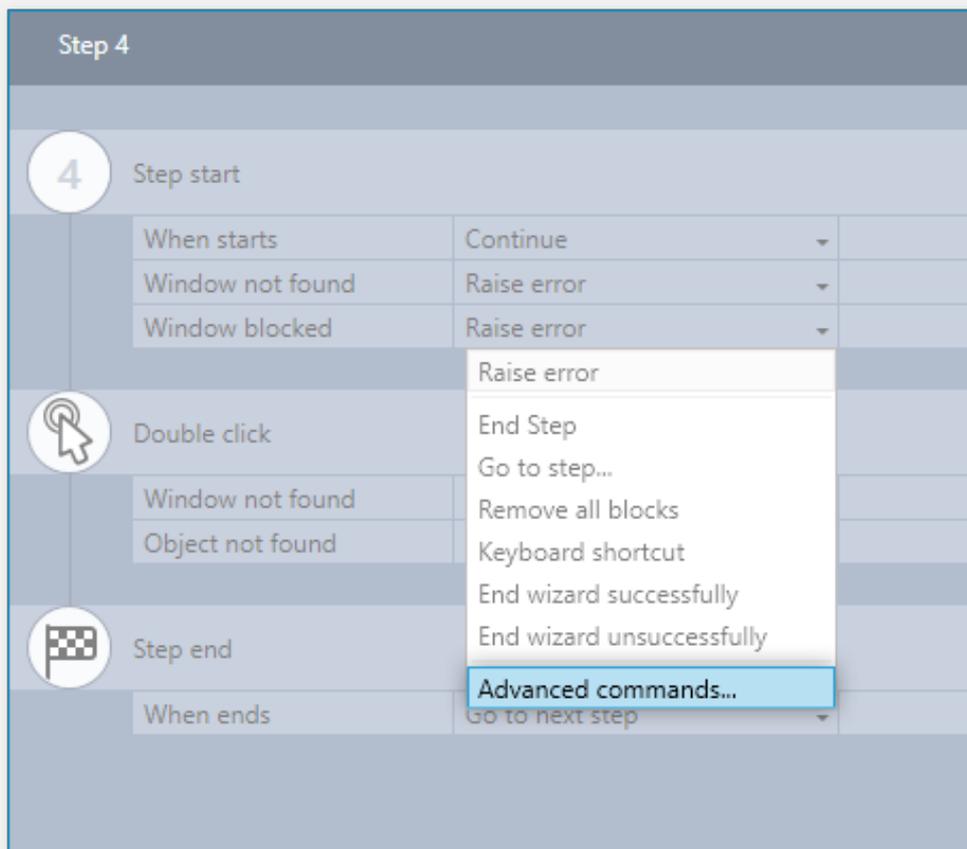
Designed to work in conjunction with Kryon's fallback features (i.e., an action or series of actions you have defined for when an error occurs), this command instructs the wizard to return to the default procedures for handling an error after performing the fallback procedures you have defined. For more information on Fallbacks, see the *Fallbacks Tab* section of the Kryon 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: **User Guide - Kryon Web Service API** (Get Status:WebResponse Parameters)

Using the REPORT WIZARD OUTPUT command

1 Choose whether the reported output should be: (1) the value of a variable; or (2) 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	98
Input Text	99
Get Mouse Position	100
Drag & Drop	101
Mouse Click	103
Wait for Busy Cursor	104
Set Caps Lock State	105
Get Caps Lock State	106

Use Keyboard Shortcut

Enter predefined keystrokes into the active application. This command 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 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 the keystrokes 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 the keystrokes should appear on the end user's screen while the wizard is running

Input Text

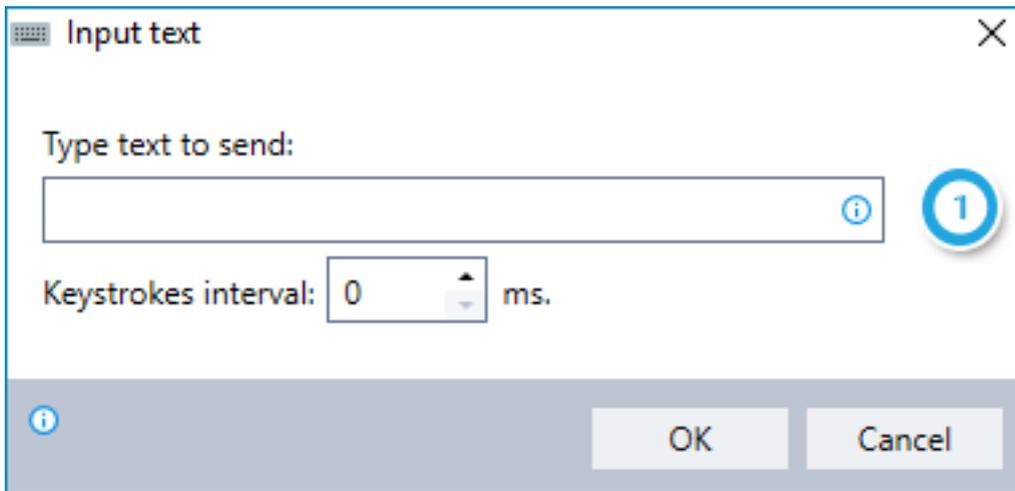
Send the text you've entered (or the text stored in a variable) to the active application.



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 to send (can include free text and/or values copied from different variables); **and**

Select the speed at which the keystrokes will be sent

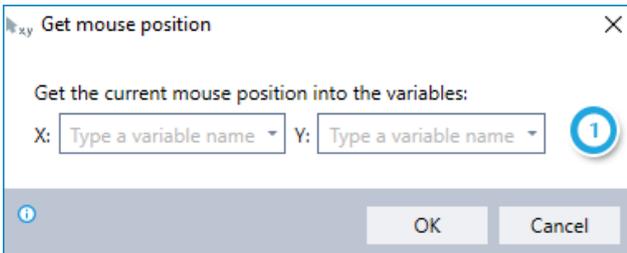
- The longer the interval selected, the slower the keystrokes are sent
- Applications/configurations can vary widely in how quickly they are able to accept data, so finding the best interval may require a bit of experimentation

Get Mouse Position

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

This command 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 in which to place the **x** coordinate; **and**
Enter the name of the variable in which to place the **y** coordinate

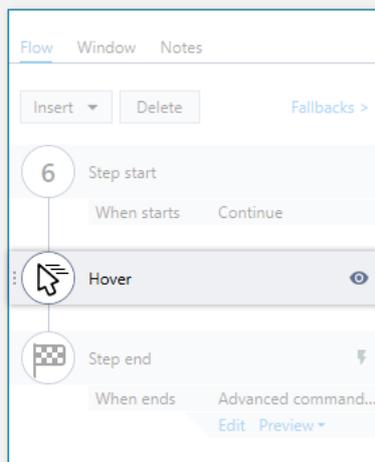


TIP

Getting the mouse where it needs to be

Before using the **GET MOUSE POSITION** command, you need to instruct the wizard to place the mouse in the location 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. Finally, 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 Where you want to drag the object from:

Identify the source location either as: (1) the current mouse position; or (2) a location identified by **x** and **y** coordinates stored in variables

- If using **x** and **y** coordinates, enter the names of the variables in which the coordinates are stored

2 Where you want the object to be dropped:

Enter the names of the variables in which the **x** and **y** coordinates of the target location are stored



Specify additional details about the desired drag & drop operation:

- Optional delay before drag and/or drop
- Whether the mouse movement should appear on the end user's screen while the wizard is running
- Whether the drag & drop should occur in combination with the <CTRL> or <SHIFT> keys



TIP

How do I get the coordinates of the mouse position?

See [GET MOUSE POSITION](#).

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 either: (1) at a location identified by **x** and **y** coordinates stored in variables; or (2) at the current mouse position

- If using **x** and **y** coordinates, enter the names of the variables in which the coordinates are stored

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

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

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.



TIP

How do I get the coordinates of the mouse position?

See [GET MOUSE POSITION](#).

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 the wizard to wait for the cursor to stop spinning before performing the next step.

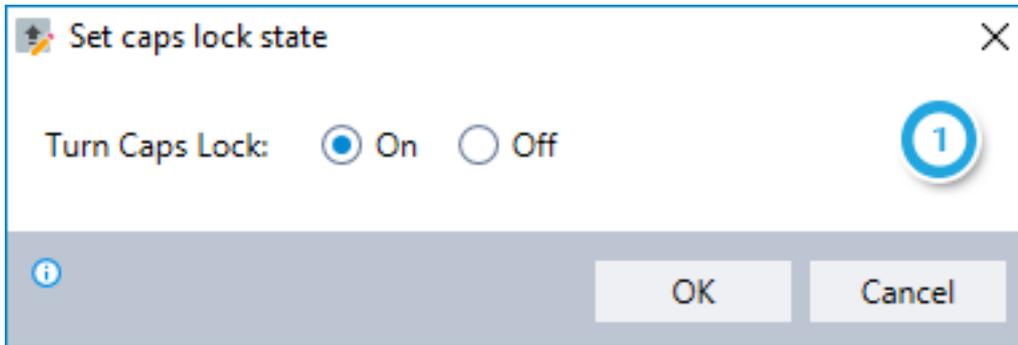
The **WAIT FOR BUSY CURSOR** command has no configurable options and can be added to a wizard simply by dragging it into the Editor Pane of the Advanced Commands view.

Set Caps Lock State

Set **Caps Lock** to **On** or **Off**.

This command is often used after checking the current state with the [GET CAPS LOCK STATE](#) command.

Using the SET CAPS LOCK STATE command



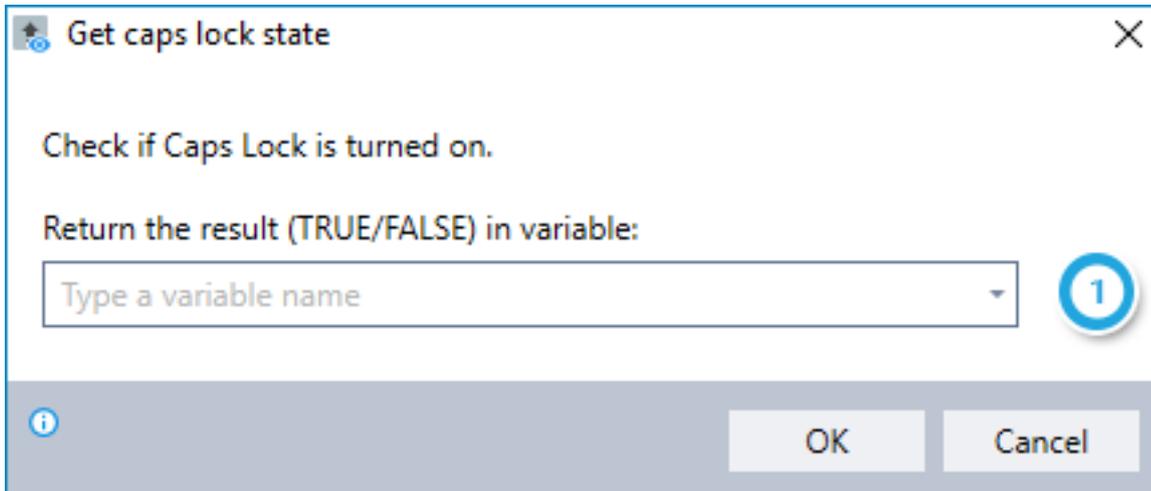
- 1 Choose whether to set **Caps Lock** to **On** or **Off**

Get Caps Lock State

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

This command 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 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 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 Kryon 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), the wizard first stores the blocked action and then proceeds according to the logic you have defined.

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 wizard run.

Allow Last Stored Action

Instruct the wizard 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.

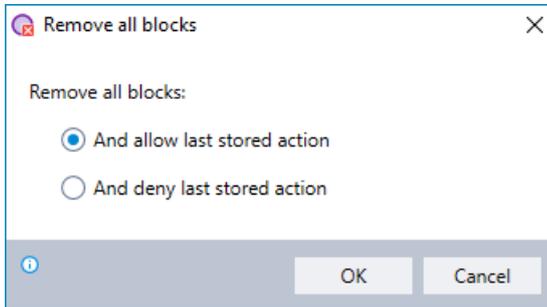
Deny Last Stored Action

Instruct the wizard **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 the wizard 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

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	111
Validate Date	112
Get Day of Month	113
Compare Dates	115
Add/Subtract Date	117
Calculate Date Range	118
Check Day of Week	120
Format Date	122
Get Current Time	125
Compare Time	126
Add/Subtract Time	128
Calculate Time Range	129
Convert Between Time Zones	131



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 Kryon Robot is running) and place the result into a new or existing variable.

Using the GET CURRENT DATE command

The screenshot shows a dialog box titled "Get current date". It has a close button in the top right corner. The main content area contains a dropdown menu with "day" selected, followed by the text "into the variable:". Below this is a text input field with the placeholder "Type a variable name". At the bottom are "OK" and "Cancel" buttons. A blue circle with the number "1" is next to the dropdown, and a blue circle with the number "2" is next to the text input field.

- 1 Choose which date parameter to retrieve: day, month, or year
- 2 Enter the name of the variable into which 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 the wizard 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 the wizard 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

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

- Get the First day of:** This section contains two dropdown menus. The first is labeled "Month:" and the second is labeled "Year:". Both dropdowns currently display "Type a variable name". A blue circle with the number "1" is positioned to the right of these dropdowns.
- Count working days only:** This section has a checkbox that is currently unchecked. Below the checkbox are seven radio button options for the days of the week: "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", and "Sun". A blue circle with the number "2" is positioned to the right of these options.
- Return the result in variable:** This section contains a single dropdown menu that currently displays "Type a variable name". A blue circle with the number "3" is positioned to the right of this dropdown.
- Error handling:** This section is collapsed, indicated by a downward-pointing chevron icon. It contains a label "Error handling" followed by an information icon (i). A blue circle with the number "4" is positioned to the right of this section.

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

- 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 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 the wizard 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

- 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 the wizard 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 the wizard 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

Calculate date range

Calculate the number of days:

From Date	To Date
Day: <input type="text" value="Type a variable name"/> Variable value: 1 - number of days in month	Day: <input type="text" value="Type a variable name"/>
Month: <input type="text" value="Type a variable name"/> Variable value: 1 - 12	Month: <input type="text" value="Type a variable name"/>
Year: <input type="text" value="Type a variable name"/> Variable value: 1 - 9999	Year: <input type="text" value="Type a variable name"/>

Count working days only

Mon Tue Wed Thu Fri Sat Sur

Return the result in variable:

▼ Error handling

OK Cancel

- 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 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 the wizard 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, 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 the wizard 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 output formats](#))

Using the FORMAT DATE command

The screenshot shows a 'Format date' dialog box with the following sections and callouts:

- Input date:** A dropdown menu set to 'Current date' with callout 1.
- Output format:** Radio buttons for 'Long date' (selected), 'Short date', and 'Custom format'. Below 'Custom format' is an empty text field with a dropdown arrow and the example 'dd/MM/yyyy'. Callout 2 is next to the radio buttons.
- Output region:** Radio buttons for 'Use robot's regional format' (selected) and 'Use a different regional format:'. Below is an empty dropdown menu. Callout 3 is next to the radio buttons.
- Return the result in this variable:** A dropdown menu with the placeholder text 'Type a variable name'. Callout 4 is next to the dropdown.

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

- 1 Select the input type for the date to be formatted and provide additional information as required (varies by input type)

<p>Input date:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 10px;">Current date ▾</div> <p>Input date:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 10px;">Use day/month/year variables ▾</div> <p>Day: Type a variable name ▾</p> <p>Month: Type a variable name ▾</p> <p>Year: Type a variable name ▾</p> <p>Input date:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 10px;">Custom format... ▾</div> <div style="border: 1px solid #ccc; height: 20px; width: 100%;"></div>	<p>Current date:</p> <p>The wizard will format the current date (according to the system clock of the machine on which the robot is running)</p> <ul style="list-style-type: none"> • No additional information is required <p>Use day/month/year variables:</p> <ul style="list-style-type: none"> • Enter the names of the variables that contain the day, month, and year of the date to be formatted <p>Custom format:</p> <p>Enter the date the wizard should format</p> <ul style="list-style-type: none"> • Can be free text and/or values copied from different variables • The wizard will interpret the order of month/day according to the robot's regional format (as configured in Windows) <ul style="list-style-type: none"> ◦ In other words, whether 6/5/2017 is interpreted as June 5, 2017, or May 6, 2017, depends on the standard format of the robot's region • See A NOTE ABOUT FORMATTING for additional information about correctly entering this date
--	---

- 2 Choose the format to which the specified date should be converted

- 3 Choose the regional format to be used for the output
- 4 Enter the name of the variable into which to place the result

Specifying custom output formats

To display	Use this code **case-sensitive**
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 Kryon Robot 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

The screenshot shows a dialog box titled "Compare time". It has a close button (X) in the top right corner. The dialog is divided into several sections:

- Check if:** This section is divided into two columns. Column 1 (labeled 1) contains three dropdown menus for "Hour:", "Minute:", and "Second:". Column 2 (labeled 2) also contains three dropdown menus for "Hour:", "Minute:", and "Second:". Between these two columns is a dropdown menu (labeled 3) with the text "is before".
- Return the result (TRUE/FALSE) in variable:** This section contains a single dropdown menu (labeled 4) with the text "Type a variable name".
- Error handling:** This section contains a dropdown menu (labeled 5) with a downward arrow and a question mark icon.

At the bottom of the dialog are two buttons: "OK" and "Cancel".

- 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 the wizard 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 the wizard 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
- 3 Instruct the wizard 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:

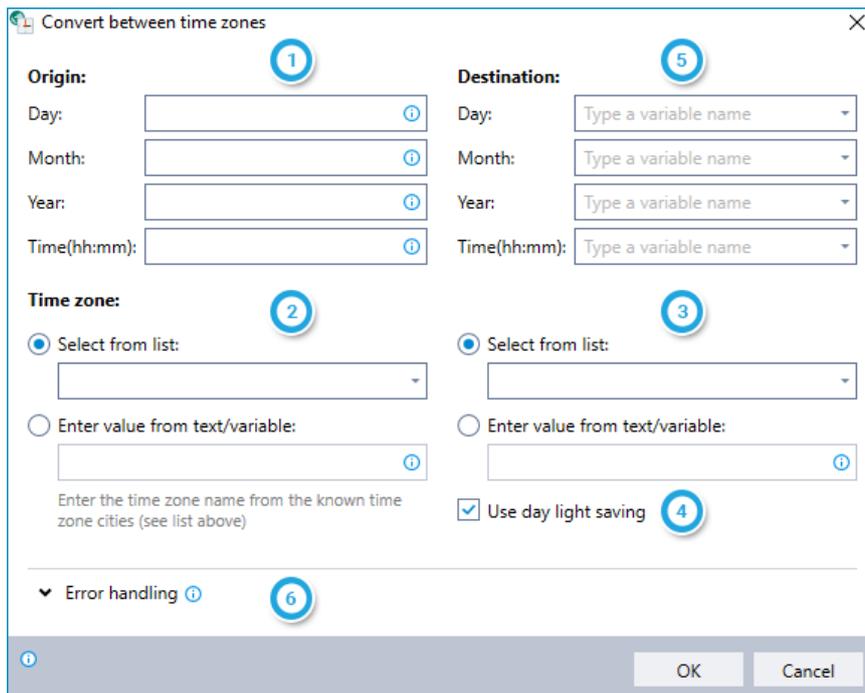
1. Use the **COMPARE TIME** command to compare whether the *shift start time* is before the *clock-in time* for each employee
2. For the employees for whom this comparison is TRUE, use the **CALCULATE TIME RANGE** command to determine the amount of time by which they were late

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

CHAPTER 7: Window Commands

In this chapter:

Get Step Window Handle	133
Find Matching Window Handles	134
Get Active Application	136
Control Window State	137
Check Window State	138
Get Active Window/Web Page	139
Capture Step Window Image	140

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, or 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

Get step window handle

Get the current step's window handle into variable:

Type a variable name

Error handling

OK Cancel

- 1 Enter the name of the variable into which you'd like to place the handle
- 2 Instruct the wizard 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 the wizard identifies the window on which the action is performed, see the *Window Detection* section of the Kryon 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 tabs for 'Flow', 'Window', and 'Notes'. Under the 'Window' tab, there are 'Insert' and 'Delete' buttons. The 'Window data' section is expanded, showing two checked properties: 'Class name' set to 'Equals XLMAIN' and 'Caption' set to 'Contains 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.

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

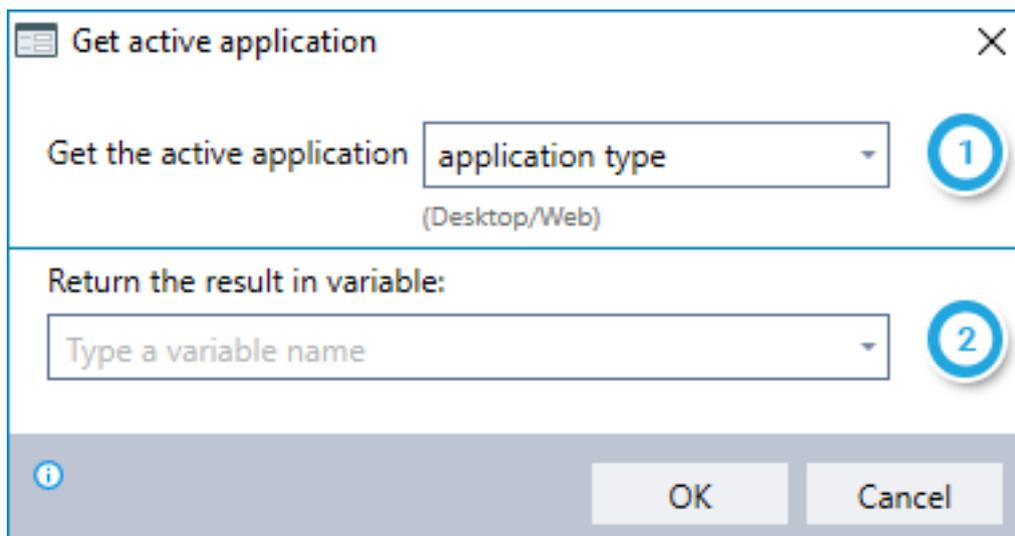
Retrieve information about the application on which the current wizard is being run and place it into a new or existing variable.

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



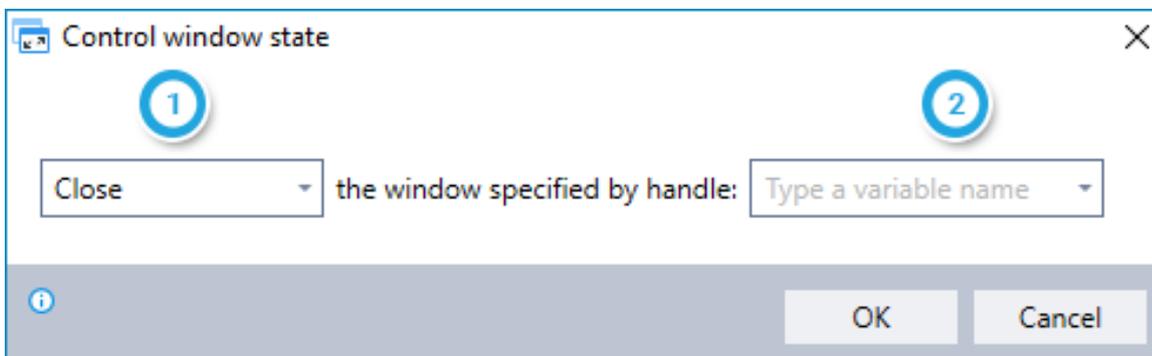
- 1 Choose the type of information you want to retrieve
- 2 Enter the name of the variable into which you'd like to place the result

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

The screenshot shows a dialog box titled "Get active window/web page". It contains a dropdown menu with "Window caption or web page URL" selected, followed by the text "into the variable:". A circled "1" is next to this dropdown. Below that is another dropdown menu with the placeholder text "Type a variable name". A circled "2" is next to this dropdown. At the bottom, there is an information icon, an "OK" button, and a "Cancel" button.

- 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 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 8: File Commands

In this chapter:

Create a Text File	142
Read From Text File	144
Write to Text File	145
Does File Exist	146
Copy a File	147
Move a File	148
Rename a File	149
Delete a File	150
Delete File(s)	151
Monitor File Changes	153

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 the should 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 the wizard 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 from which to read the contents
- 2 Enter the name of the variable into which 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 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 the wizard exits the loop when reaching the end of the file
- 4 Instruct the wizard 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

The screenshot shows a dialog box titled "Write to text file". It has a close button (X) in the top right corner. The dialog is divided into several sections:

- In the file:** A text input field with an information icon (i) to its right and a "Browse..." button. A blue circle with the number 1 is overlaid on the right side of this section.
- Write the contents of the variable:** A dropdown menu with the text "Type a variable name" and a downward arrow. A blue circle with the number 2 is overlaid on the right side of this section.
- Append text / Overwrite text:** Two radio buttons. The "Append text" radio button is selected. A blue circle with the number 3 is overlaid on the right side of this section.
- Error handling:** A section with a downward arrow and the text "Error handling" followed by an information icon (i). A blue circle with the number 4 is overlaid on the right side of this section.

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 **full file path and file name** of the text file in which to write
 - If the file does not exist, the wizard 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 the wizard 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". The first section is labeled "Check if the file" followed by a text input field and 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:" followed by 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, there are "OK" and "Cancel" buttons. A small information icon is visible in the bottom left corner of the dialog.

- 1 Enter the **full file path and file name** of the file for which to check
- 2 Enter the name of the variable into which 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

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

- File to copy:** A text input field with an information icon (i) on the right. A blue circle with the number 1 is overlaid on the right side of this field.
- Target folder:** A text input field with an information icon (i) on the right. A blue circle with the number 2 is overlaid on the right side of this field.
- Override already existing files in target folder:** A checkbox followed by the text "Override already existing files in target folder". A blue circle with the number 3 is overlaid on the right side of this section.
- Error handling:** A dropdown menu with a downward arrow and the text "Error handling" followed by an information icon (i). A blue circle with the number 4 is overlaid on the right side of this section.

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

- 1 Enter the **full file path and file name** of the file to copy
- 2 Enter the **full path** of the folder to which to copy the file
- 3 Indicate whether or not to overwrite any existing file of the same name in this location
- 4 Instruct the wizard 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 'Move a file' dialog box. It has a title bar with a close button. The main area is divided into three sections. The first section is 'File to move:' with a text input field and an info icon. The second section is 'Target folder:' with a text input field and an info icon. The third section is 'Error handling' with a dropdown arrow, an info icon, and a numbered callout '3'. At the bottom, there is a grey bar with an info icon, 'OK', and 'Cancel' buttons. Numbered callouts '1', '2', and '3' are placed to the right of the input fields and the error handling section respectively.

- 1 Enter the **full file path and file name** of the file to move
- 2 Enter the **full path** of the folder to which 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Rename a File

Rename an existing file.

Using the RENAME A FILE command

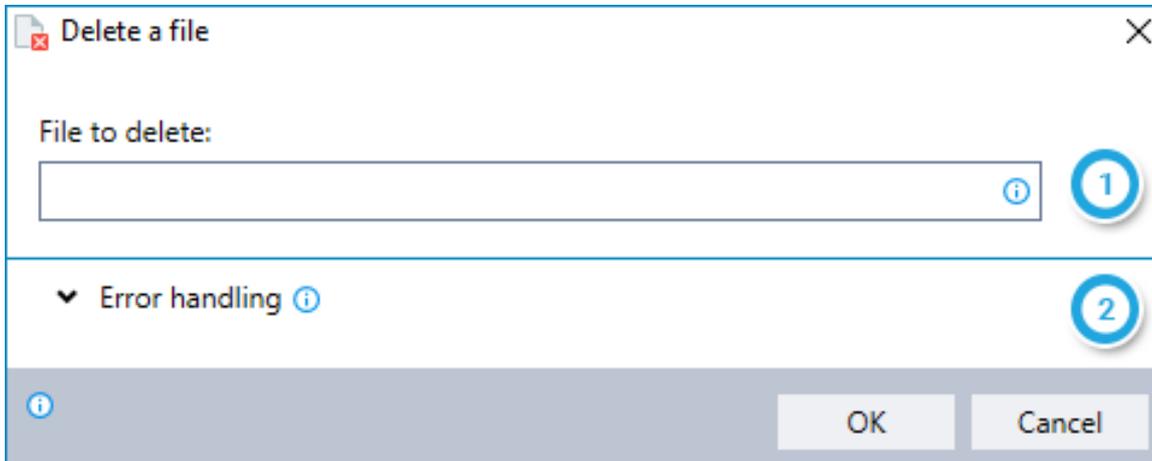
- 1 Enter the **full file path and file name** of the file to rename
- 2 Enter the new name for the file, including the file extension
 - If you enter only the file name (with no path), the file will be renamed and remain in its current location
 - If you enter a new file path along with the file name, 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 the wizard 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 to delete
- 2 Instruct the wizard 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

- 1 Enter the **full file path** of the folder from which 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 deleted
 - The number of files matching the specified pattern that failed to delete
- 4 Instruct the wizard 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.
- The wizard 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 a close button (X) in the top right corner. The dialog is divided into several sections, each with a numbered callout (1-6) in a blue circle:

- 1** Root folder: [Text input field with an information icon (i)]
- Make sure the folder is accessible for all end users.
- Include sub-folders
- 2** File name to monitor: [Text input field with an information icon (i)]
- Enter file name or pattern (use stars: *). Example: *.xlsx
- File created (or renamed)
- File modified
- File deleted
- 3**
- 4** Return file path in variable: [Dropdown menu with "Type a variable name" selected]
- Timeout: [Spin box with "5" and up/down arrows] minutes
- 5**
- Error handling [Information icon (i)]
- 6**

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 **full path** of the top-level folder in which files should be monitored; **and** indicate whether or not to also monitor files in subfolders
- 2 **To monitor a single file:** Enter the name of 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 to monitor
- 4 Enter the name of the variable into which to place the file path of the new, renamed, modified, or deleted file
- 5 Indicate if the wizard should stop monitoring (i.e., timeout) after a certain number of minutes
- 6 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 9: Folder Commands

In this chapter:

Create a Folder	156
Get Folder Location	157
Get Files	158
Does Folder Exist	160
Is Folder Empty	161
Copy a Folder	162
Move a Folder	163
Rename a Folder	164
Delete a Folder	165
Monitor Folder Changes	166

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 to place the path of the new folder

2 Indicate if the folder should 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 the wizard 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:

OK Cancel

- 1 Select the system folder for which to retrieve location
- 2 Enter the name of the variable into which 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

The screenshot shows a dialog box titled "Get files" with a close button (X) in the top right corner. The dialog is divided into several sections, each with a numbered callout (1-7) in a blue circle:

- 1**: A text input field labeled "Root folder:" with a help icon (i) to its right. Below the field is the text "Make sure the folder is accessible for all end users."
- 2**: A text input field labeled "File name to retrieve (optional):" with a help icon (i) to its right. Below the field is the text "Enter file name or pattern (use stars: *). Example: Logs*"
- 3**: A checkbox labeled "Include sub-folders".
- 4**: A checkbox labeled "Include folder name in output".
- 5**: A section labeled "Sort by:" containing a dropdown menu with "File name" selected, and two radio buttons: "Ascending" (selected) and "Descending".
- 6**: A section labeled "Return results (file names) in variable:" containing a dropdown menu with "Type a variable name" selected.
- 7**: A text input field labeled "Result delimiter:".

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

- 1** Enter the **full path** of the root folder from which 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 method for sorting the list
- 6 Enter the name of the variable into which 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.

Return the result (TRUE/FALSE) in variable:

OK Cancel

- 1 Enter the **full path** of the folder for which to check
- 2 Enter the name of the variable into which 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 in the top right corner. The main content area is divided into two sections. The first section contains the text "Check if the folder" followed by a text input field, an information icon, and the text "is empty". A blue circle with the number "1" is positioned to the right of this section. The second section is titled "Return the result (TRUE/FALSE) in variable:" and 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 section. At the bottom of the dialog, there is an information icon on the left and two buttons labeled "OK" and "Cancel" on the right.

- 1 Enter the **full path** of the folder you would like to check
- 2 Enter the name of the variable into which 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

The screenshot shows a dialog box titled "Copy a folder". It has a close button in the top right corner. The dialog is divided into three sections. The first section is labeled "Folder to copy:" and contains a text input field with an information icon to its right. A blue circle with the number "1" is positioned to the right of this input field. The second section is labeled "Target folder:" and contains a text input field with an information icon to its right. A blue circle with the number "2" is positioned to the right of this input field. The third section is labeled "Error handling" with a dropdown arrow and an information icon to its right. A blue circle with the number "3" is positioned to the right of this section. At the bottom of the dialog, there is an information icon on the left, and "OK" and "Cancel" buttons on the right.

- 1 Enter the **full path** of the folder to copy
- 2 Enter the **full path** of the target folder to which 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 the wizard 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

The screenshot shows a dialog box titled "Move a folder" with a close button in the top right corner. The dialog is divided into three sections. The first section, labeled "Folder to move:", contains a text input field with an information icon to its right and a circled "1" callout. The second section, labeled "Target folder:", contains a text input field with an information icon to its right and a circled "2" callout. The third section, labeled "Error handling", has a dropdown arrow, an information icon, and a circled "3" callout. At the bottom of the dialog, there is an information icon on the left and "OK" and "Cancel" buttons on the right.

- 1 Enter the **full path** of the folder to move
- 2 Enter the **full path** of the target folder to which 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Rename a Folder

Rename an existing folder.

Using the RENAME A FOLDER command

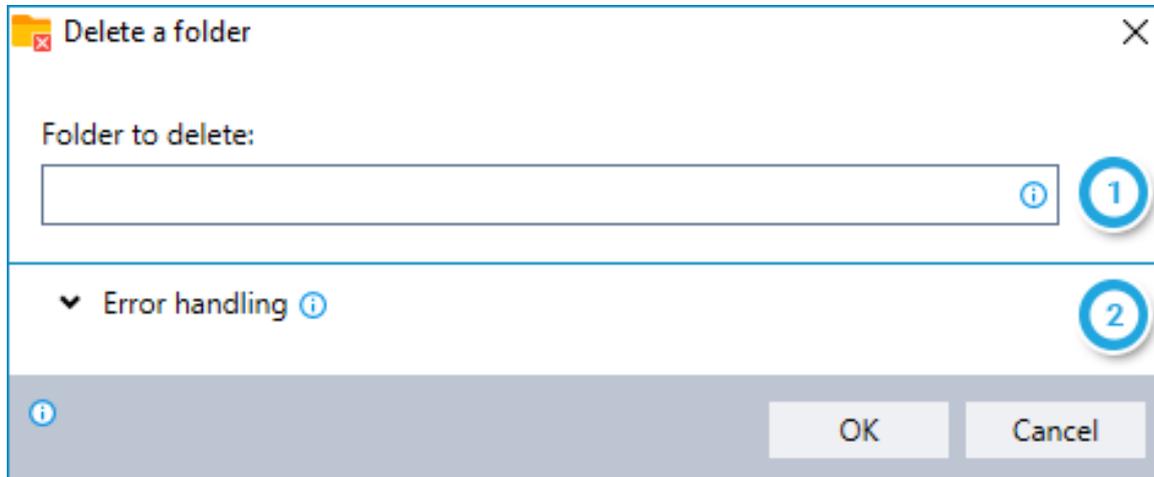
The screenshot shows a dialog box titled "Rename a folder" with a close button in the top right corner. The dialog is divided into three horizontal sections. The first section is labeled "Folder to rename:" and contains a text input field with an information icon (i) to its right, and a circled number "1" to the right of the input field. The second section is labeled "New name:" and contains a text input field with an information icon (i) to its right, and a circled number "2" to the right of the input field. The third section is labeled "Error handling" with a dropdown arrow and an information icon (i) to its right, and a circled number "3" to the right of the section. At the bottom of the dialog, there are "OK" and "Cancel" buttons, and a small information icon (i) on the left side of the bottom bar.

- 1 Enter the **full path** of the folder to rename
- 2 Enter the new name for 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 the wizard 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 to delete
- 2 Instruct the wizard 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.
- The wizard 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

The screenshot shows a dialog box titled "Monitor folder changes" with a close button (X) in the top right corner. The dialog is divided into several sections, each with a numbered callout (1-6) in a blue circle:

- 1:** "Root folder:" followed by a text input field and an information icon (i).
- Below the input field: "Make sure the folder is accessible for all end users."
- "Include sub-folders"
- 2:** "Folder name to monitor (optional):" followed by a text input field and an information icon (i).
- Below the input field: "Enter folder name or pattern (use stars: *). Example: Logs*"
- "Folder created (or renamed)"
- "Folder deleted"
- 3:** (Callout 3 is positioned to the right of the event checkboxes)
- 4:** "Return folder path in variable:" followed by a dropdown menu with the text "Type a variable name" and a downward arrow, and an information icon (i).
- "Timeout: 5 minutes" (The number 5 is in a small input field with up/down arrows)
- 5:** (Callout 5 is positioned to the right of the timeout field)
- 6:** "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 **full path** of the root folder in which 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 to monitor
- 4 Enter the name of the variable into which to place the path of the new, renamed, or deleted folder
- 5 Indicate if the wizard should stop monitoring (i.e., timeout) after a certain number of minutes
- 6 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 10: AI-Powered Document and Text Analysis Commands

In this chapter:

What's Required?	169
Azure Licenses	171
OCR: Documents	172
Get Text (SmartScan)	176
Get Value	177
Get Selection Element State	179
Does Word Exist (SmartScan)	181
Get Date/Time	182
Save as Image	184
Save to Excel	185
OCR: Printed and Handwritten Text	187
Get Text (MS Azure)	190
Does Word Exist (MS Azure)	191
Form Recognizer	192
Get Receipt Data	195
Text Analytics: Analyze Sentiment	197
Text Analytics: Detect Language	198
Text Analytics: Identify Key Phrases	199
Convert to Text (SmartScan+)	200
Convert to Text (Tesseract)	201

What's Required?

AI Booster

Kryon is pleased to offer **AI Booster**, powered by Microsoft Azure Cognitive Services. This suite of AI-based advanced commands gives Kryon robots unprecedented abilities to understand both structured and unstructured data. Using these commands requires a license obtained directly from Microsoft via the [Azure Cognitive Services website](#). A free trial as well as a free-tier license are available in order to help you get started.

AI Booster

The following commands are part of Kryon's AI Booster suite:

- OCR: Printed and Handwritten Text
- Get text
- Does word exist
- Form Recognizer
- Get receipt data
- Text analytics: Analyze sentiment
- Text analytics: Detect language
- Text analytics: Identify key phrases

ABBYY integrations for scanned documents

Kryon RPA offers several advanced scanned document solutions through available integrations with ABBYY OCR products. To explore the options and licensing requirements, get in touch with your Kryon sales contact.

SmartScan Feature

The following commands are part of our **SmartScan** solution, which requires integration with **ABBYY FineReader**:

- OCR: Documents
- Get text
- Get value
- Get checkbox state
- Does word exist
- Get date/time

- Save as image
- Save to Excel

SmartScan+ Feature

SmartScan+ is an integration with the **ABBYY FlexiCapture** platform, which enables the analysis of unstructured data using machine learning. While most of the benefits of this integration are realized outside the world of Advanced Commands, one "bonus" Advanced Command utilizes the ABBYY OCR engine and comes along with **SmartScan+**:

- Convert to text (SmartScan+)

OCR out-of-the-box

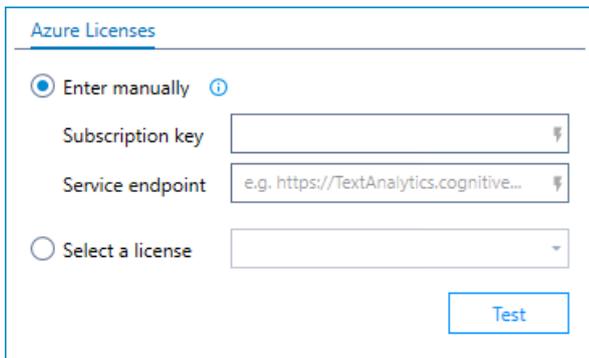
The following Advanced Command offers OCR capabilities as part of the standard Kryon installation and does not require an additional license:

- Convert to text (Tesseract)

Azure Licenses

Using **AI Booster** advanced commands requires a license obtained directly from Microsoft via the [Azure Cognitive Services website](#). At the time each such command is executed (at wizard runtime), the license is validated via the Internet.

When you add a command that requires an Azure license to a wizard, license information must be provided on the command's **AZURE LICENSES** tab. You can choose to enter license information manually or use a license previously defined in Kryon Admin.



The screenshot shows a form titled "Azure Licenses" with two radio button options. The first option, "Enter manually", is selected and includes a help icon. Below it are two text input fields: "Subscription key" and "Service endpoint", both with dropdown arrows. The "Service endpoint" field contains the text "e.g. https://TextAnalytics.cognitive...". The second option is "Select a license", which is accompanied by a dropdown menu. A "Test" button is located at the bottom right of the form.

- To enter license information manually, enter the subscription key and service endpoint, as provided by Microsoft. These fields can accept free text or values stored in variables.
- To select a license previously defined in Kryon Admin, select it from the dropdown list.
- Use the  button to verify your license and connectivity to Azure Cognitive Services.

OCR: Documents

SmartScan Feature

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

Using the OCR: DOCUMENTS command

Step #1 - Analyze & separate the documents

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

OCR: Documents (SmartScan)

Analyze file: ⓘ ...

Pages to Analyze Document Separation

Analyze all pages

Analyze specific pages:

ⓘ

Comma separated. Example: 1, 5, 7-9, 13-*

Ignore pages with any of the following words:

ⓘ

Comma separated.

Return Values

Set the total number of **pages** in variable:

Type a variable name

Set the current **page** number in variable:

Type a variable name

Set the number of **documents on current page** in variable:

Type a variable name

Set the current **document** number in variable:

Type a variable name

▼ Error handling ⓘ

OK Cancel

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 document icons). Below each option is a brief description of the method.

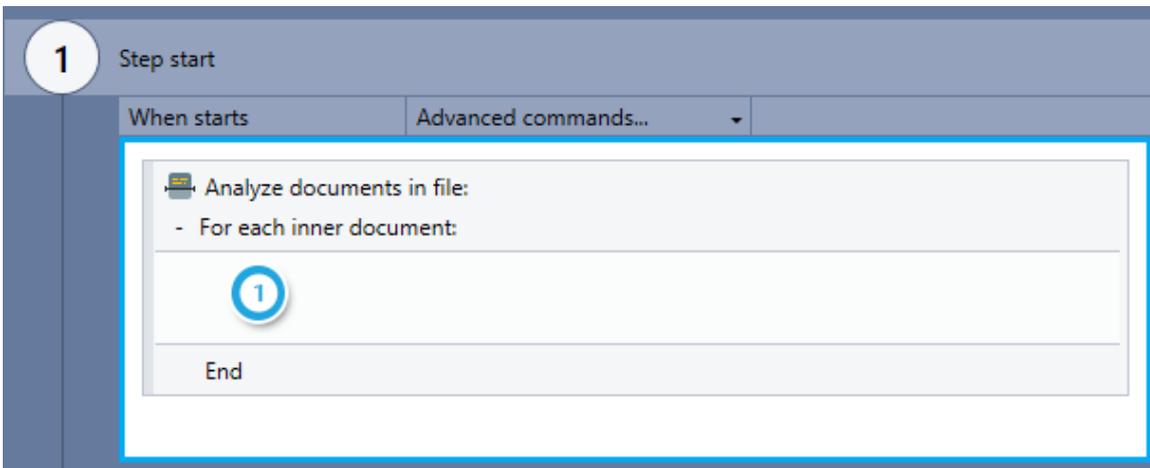
3 Indicate if you wish to place information about total pages/documents and current page/document into variables

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

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

Step #2 - Define the actions

Upon adding the **OCR: DOCUMENTS** 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

The wizard performs 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 **OCR: DOCUMENTS** container (i.e., don't feel limited to using just the OCR 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 **OCR: DOCUMENTS** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

Get Text (SmartScan)

SmartScan Feature

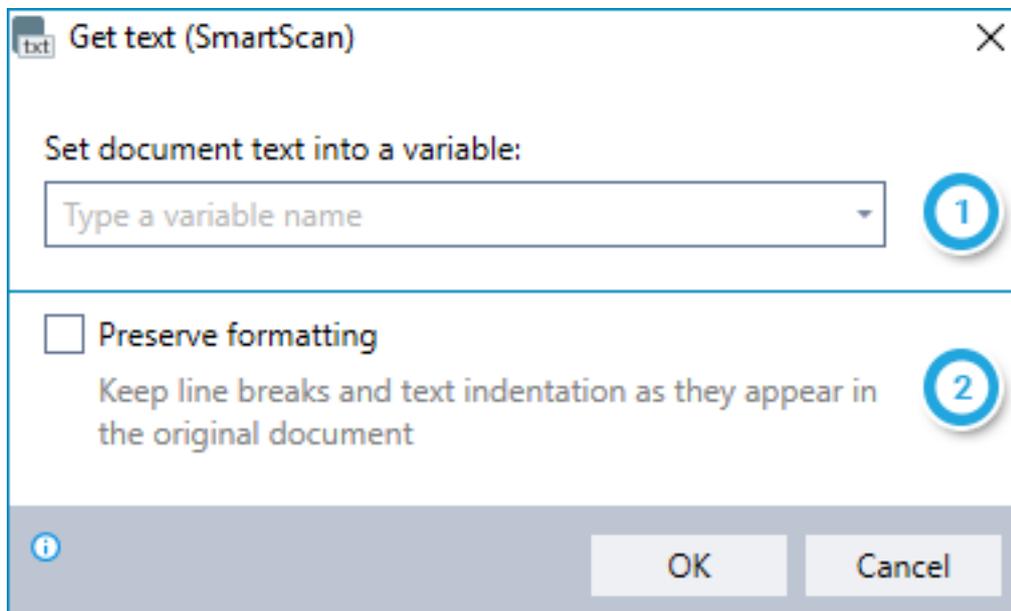
Place text from a scanned document into a variable.



NOTE

This command can be used only within an **OCR: DOCUMENTS** container.

Using the GET TEXT command



Enter the name of the variable into which to place the text



Indicate whether to preserve formatting (line breaks and indentation) from the scanned 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 **OCR: DOCUMENTS** container.

Using the GET VALUE command

The screenshot shows the 'Get value (SmartScan)' dialog box with the following elements and numbered callouts:

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

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

- 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, only the area in close proximity to the label word will be searched
- 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 the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

Get Selection Element State

SmartScan Feature

Determine whether a check box, radio button, or similar selector in a scanned document is selected (checked) by searching for it next to a specified word (the "label word").



NOTE

This command can be used only within an **OCR: DOCUMENTS** container.

Using the GET SELECTION ELEMENT STATE command

The screenshot shows a dialog box titled "Get selection element state (SmartScan)" with a close button (X) in the top right corner. The dialog contains the following elements:

- 1**: A text input field for "Selection element is located to the right of the words:". Below it is a note: "Specify all possible words (comma separated)."
- 2**: A checkbox labeled "Allow close match" with an information icon. Below it is a slider for "Required accuracy" set to 100%.
- 3**: A checkbox labeled "Add another condition".
- 4**: A section titled "Set the value in variable:" with a dropdown menu currently showing "Select a variable".
- 5**: A section titled "Error handling" with a downward arrow.

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

- 1 Choose where the relevant selection element 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 to place the result (will be TRUE if the element is selected or FALSE if unselected)
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Does Word Exist (SmartScan)

SmartScan Feature

Check 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 [OCR: DOCUMENTS](#) container.

Using the DOES WORD EXIST command

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

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 **OCR: DOCUMENTS** container.

Using the GET DATE/TIME command

The screenshot shows a dialog box titled "Get date/time (SmartScan)" with a close button (X) in the top right corner. The dialog is divided into several sections, each with a numbered callout (1-6) in a blue circle:

- 1:** A dropdown menu with the text "Get last date".
- 2:** A section titled "For numeric date patterns:" containing two radio button options: "01/02/2017 = February 1st" (selected) and "01/02/2017 = January 2nd".
- 3:** A section titled "Custom date pattern (optional):" with an information icon (i) and an empty text input field.
- 4:** A section titled "Date range in years:" with two text input fields containing "2016" and "2017", separated by the word "to". Each input field has an information icon (i).
- 5:** A section titled "Return the result in variable:" with a dropdown menu containing the text "Type a variable name".
- 6:** A section titled "Error handling" with a downward-pointing chevron icon.

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

- 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 dates should be identified
 - 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Save as Image

SmartScan Feature

Save a scanned document as an image file.



NOTE

This command can be used only within an [OCR: DOCUMENTS](#) container.

Using the 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Save to Excel

SmartScan Feature

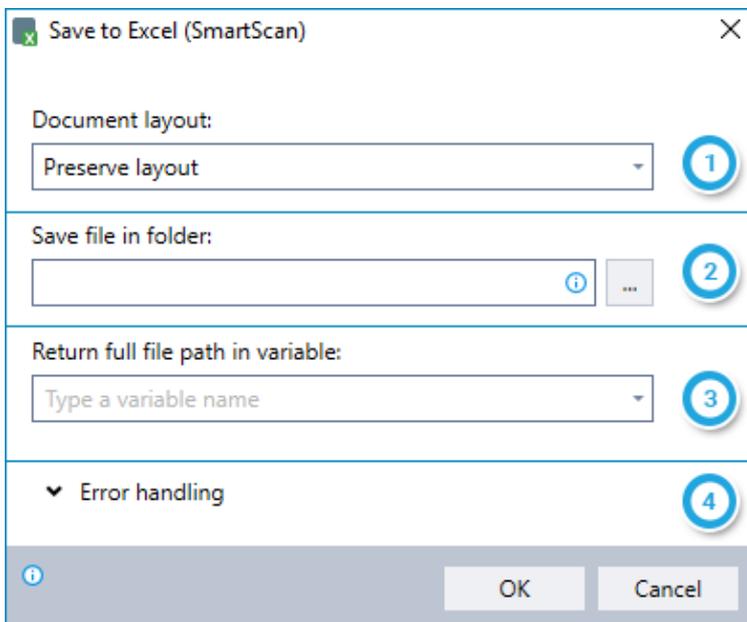
Save text from a scanned document into an Excel spreadsheet.



NOTE

This command can be used only within an **OCR: DOCUMENTS** container.

Using the SAVE TO EXCEL command



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

OCR: Printed and Handwritten Text

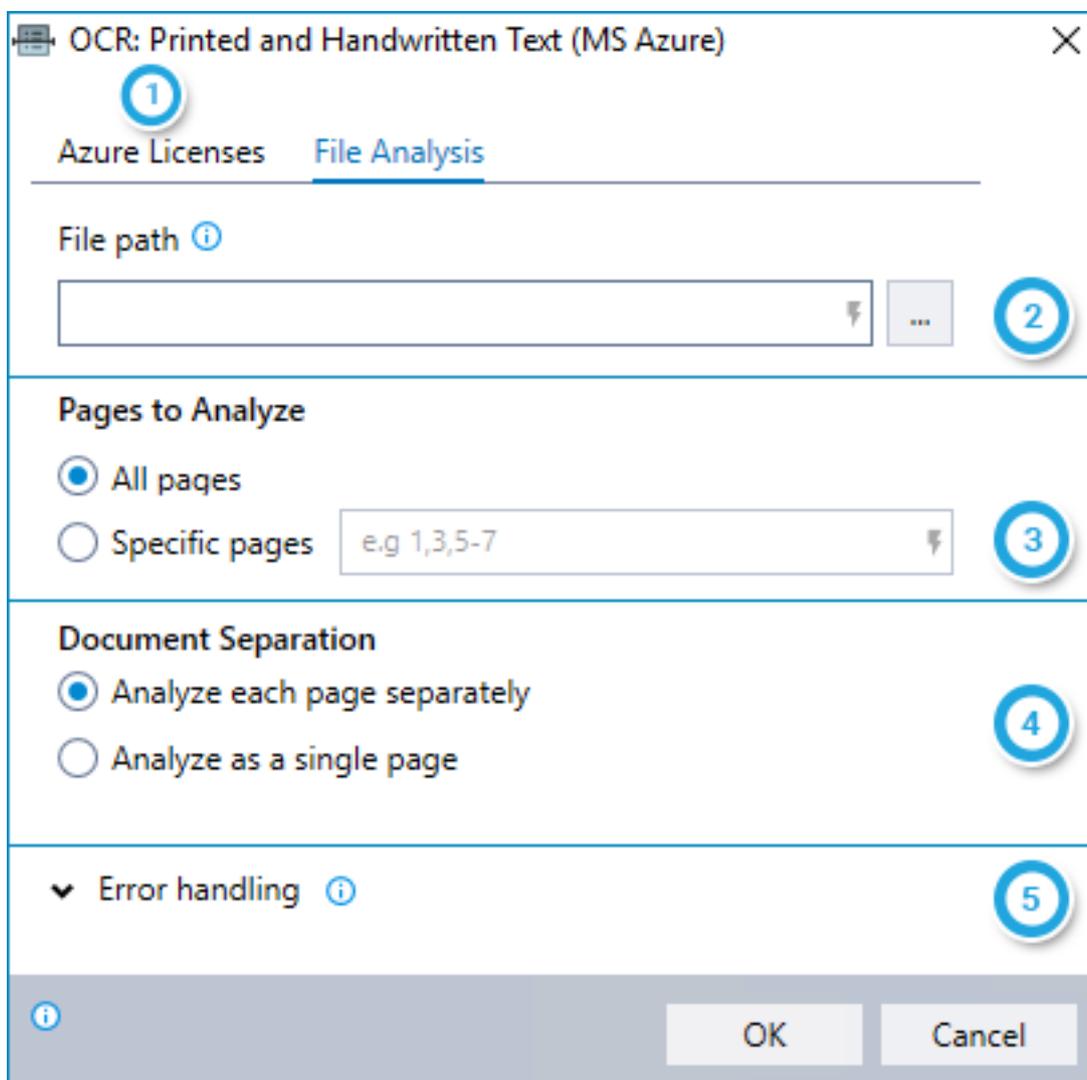
AI Booster

Analyze the printed and handwritten text of a scanned file for the purpose of performing a sequence of actions on each page.

Using the OCR: PRINTED AND HANDWRITTEN TEXT command

Step #1 - Analyze & separate the document

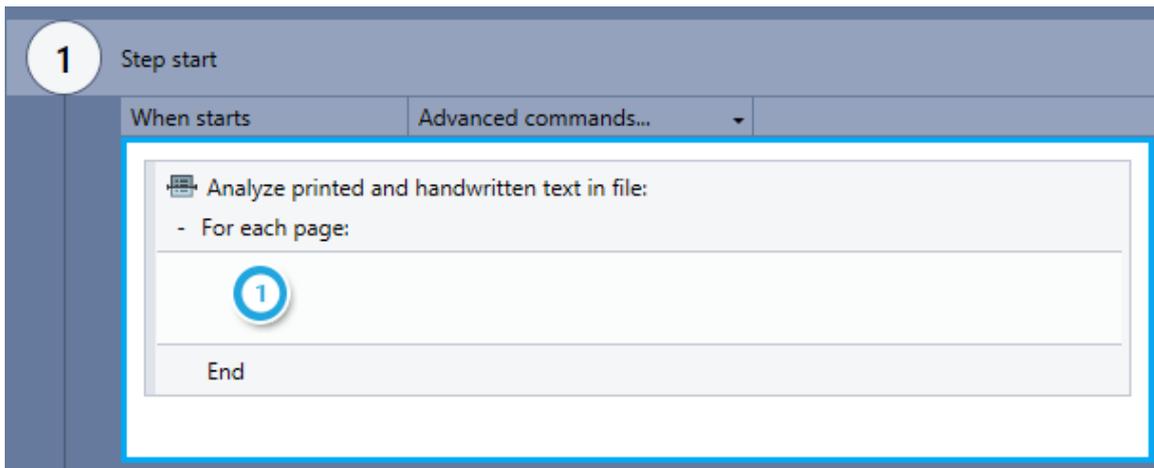
The first step in using the **OCR: PRINTED AND HANDWRITTEN TEXT** command is to analyze and separate the file on which the specified actions will be performed.



- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Select the file to analyze
- 3 Choose whether to analyze all pages of the file or enter specific pages to analyze
- 4 Choose the method for separating the file to be analyzed
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Step #2 - Define the actions

Upon adding the **OCR: PRINTED AND HANDWRITTEN TEXT** 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 the file
 - You can do this by dragging the required Advanced Command(s) directly into the container



NOTES

Loop-the-loop

If you have elected to analyze each page separately, the wizard performs 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 **OCR: PRINTED AND HANDWRITTEN TEXT** container.

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 **OCR: PRINTED AND HANDWRITTEN TEXT** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

Get Text (MS Azure)

AI Booster

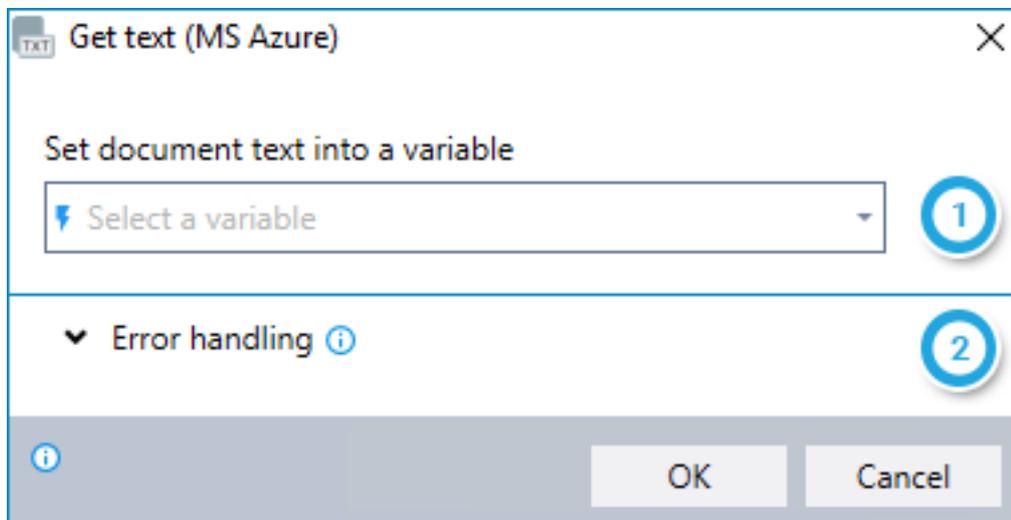
Place text from a printed or handwritten document into a variable.



NOTE

This command can be used only within an **OCR: PRINTED AND HANDWRITTEN TEXT** container.

Using the GET TEXT command



- 1 Enter the name of the variable into which to place the text
- 2 Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

Does Word Exist (MS Azure)

AI Booster

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



NOTE

This command can be used only within an [OCR: PRINTED AND HANDWRITTEN TEXT](#) container.

Using the DOES WORD EXIST command

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

Form Recognizer

AI Booster

Analyze the contents of a scanned file to extract key pieces of structured data for each recognized form.

Using the **FORM RECOGNIZER** command

Step #1 - Analyze & separate the document

The first step in using the **FORM RECOGNIZER** command is to analyze and separate the file on which the specified actions will be performed.

Form Recognizer (MS Azure)

1 Azure Licenses **File Analysis**

File path **2**

Pages to Analyze

All pages

Specific pages e.g 1,3,5-7 **3**

Document Separation

Analyze each page separately **4**

Analyze as a single page

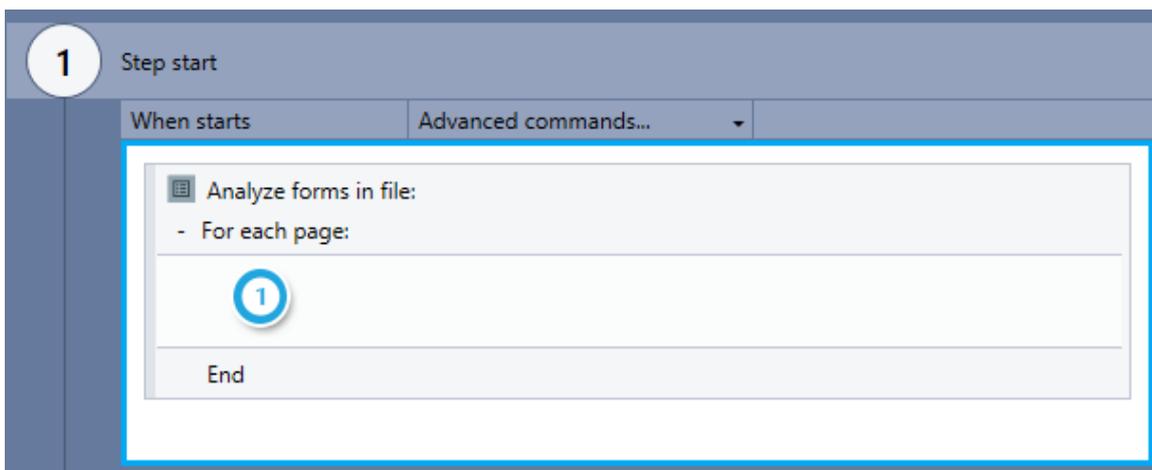
5 Error handling

6 OK Cancel

- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Select the file to analyze
- 3 Choose whether to analyze all pages of the file or enter specific pages to analyze
- 4 Choose the method for separating the file to be analyzed
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Step #2 - Define the actions

Upon adding the **FORM RECOGNIZER** 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 the file
 - You can do this by dragging the required Advanced Command(s) directly into the container



NOTES

Loop-the-loop

If you have elected to analyze each page separately, the wizard performs 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 **FORM RECOGNIZER** container.

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 **FORM RECOGNIZER** container. Make this happen by using the **LOOP: BREAK** and/or the **LOOP: RESTART** command (usually within an **IF ELSE** command).

Get Receipt Data

AI Booster

Extract relevant data from scanned sales receipts (such as transaction time/date, merchant data, taxes/totals, etc.)



NOTE

This command can be used only within a **FORM RECOGNIZER** container.

Using the GET RECEIPT DATA command

Get receipt data (MS Azure)

Select field(s) to extract:
All fields

Result type:
 Text e.g., 17%
 Value e.g., 17
 Both e.g., 17%, 17

Field delimiter: Result type delimiter:

Return results in variable:
Select a variable

▼ Error handling

OK Cancel

- 1 Select from the dropdown list the field from which to extract data, or select to extract data from all available fields
- 2 Choose whether numeric data should be returned as text, values, or both

- 3 Specify the delimiter(s) by which the returned values should be separated:
 - The data from each specified field is returned, separated by the **FIELD DELIMITER**
 - When data is returned as both text and values, the 2 types of data are separated by the **RESULT TYPE DELIMITER**
 - **Note:** This field is enabled (and required) only when you have elected a value of `Both` for the **RESULT TYPE** in 2
- 4 Enter the name of the variable into which to place the results
- 5 Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

Text Analytics: Analyze Sentiment

AI Booster

Analyze the sentiment of a given text and return the result as a score from 0% to 100% (negative to positive).

Using the TEXT ANALYTICS: ANALYZE SENTIMENT command

- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Enter the text to analyze:
 - Can be free text or a variable
 - Maximum characters: 5,120
 - Supported languages: English, Chinese-Simplified, French, German, Spanish
- 3 Enter the name of the variable in which to return the result
 - Result is presented as a score from 0% to 100% (with 0% as the most negative and 100% as the most positive)
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Text Analytics: Detect Language

AI Booster

Analyze a text and detect its language; provide confidence level of the result.

Using the TEXT ANALYTICS: DETECT LANGUAGE command

- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Enter the text to analyze:
 - Can be free text or a variable
 - Maximum characters: 5,120
 - Supported languages: Wide variety of world languages and dialects (certain rare languages may not be properly detected)
- 3 Enter the name of the variable in which to return the result
- 4 Enter the name of the variable in which to return the confidence level in the result
 - Result is presented as a score from 0% to 100% (with 0% as no confidence in the accuracy of the result and 100% as complete confidence)
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Text Analytics: Identify Key Phrases

AI Booster

Analyze a given text and return a list of detected key phrases.

Using the TEXT ANALYTICS: IDENTIFY KEY PHRASES command

The screenshot shows a dialog box titled "Text Analytics: Identify key phrases (MS Azure)". It has a close button (X) in the top right corner. The dialog is divided into several sections:

- Tabbed Interface:** Two tabs are visible: "Azure Licenses" and "Text Analysis". The "Text Analysis" tab is selected.
- Analyze text:** A text input field with a circular callout '2' next to it.
- Key phrase delimiter:** A text input field with a circular callout '3' next to it.
- Return result in variable:** A dropdown menu with "Select a variable" and a circular callout '4' next to it.
- Error handling:** A dropdown menu with a downward arrow and a circular callout '5' next to it.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

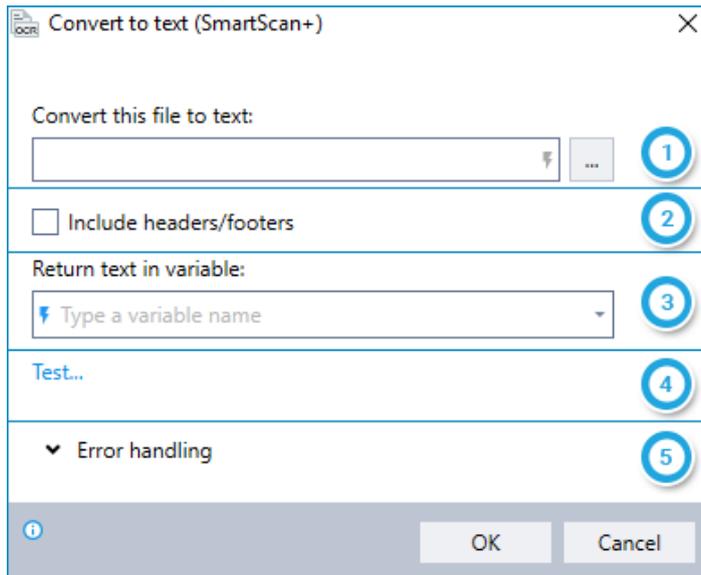
- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Enter the text to analyze:
 - Can be free text or a variable
 - Maximum characters: 5,120
 - Supported languages: English, Chinese-Simplified, French, German, Spanish
- 3 Specify the delimiter(s) by which the detected phrases should be separated in the results
- 4 Enter the name of the variable in which to return the results
- 5 Instruct the wizard 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

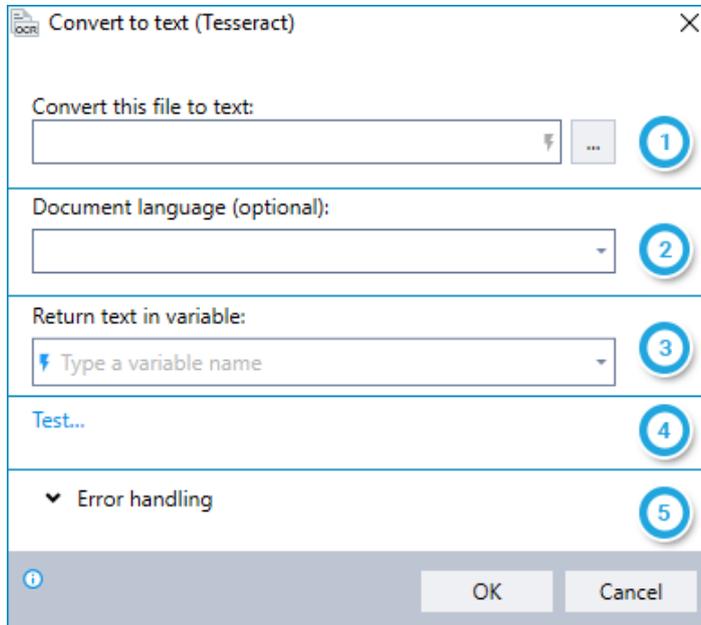


- 1 Select the PDF or image file to convert
- 2 Indicate whether you'd like to include the text of the original document's headers/footers (if any)
- 3 Enter the name of the variable into which you'd like to place the recognized text
- 4 (Optional) Test the text recognition results
- 5 Instruct the wizard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

CHAPTER 11: Digital PDF Analysis

Commands

In this chapter:

Analyze Digital PDF File	203
Page: Get Text	206

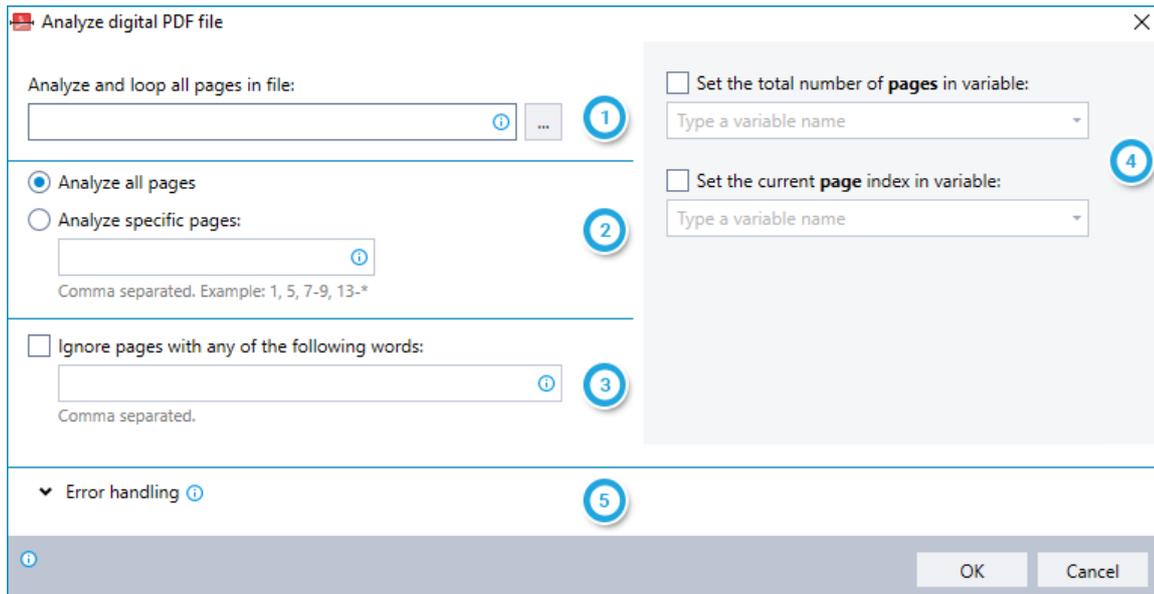
Analyze Digital PDF File

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

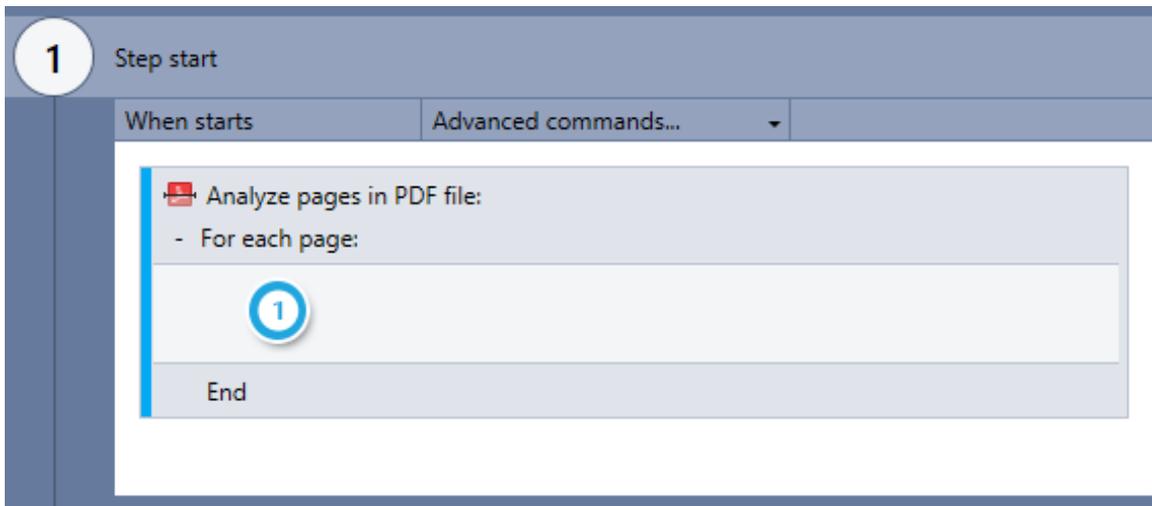
The first step in using the **ANALYZE DIGITAL PDF FILE** command is to analyze and identify the pages on which the specified actions will be performed.



- 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 the wizard 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

The wizard performs 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	208
Place in Clipboard	209
Copy Active Field Value	210
Get OS Version	211
Get Input Language	212
Read From Registry	213
Remove From Registry	215
Write to Registry	216
Get Environment Variable	217
Get Windows Event Log Data	218
Set BI Field	220
Log an Action	221
Query XML	222
Query JSON	224
Call REST API Method	226

Get From Clipboard

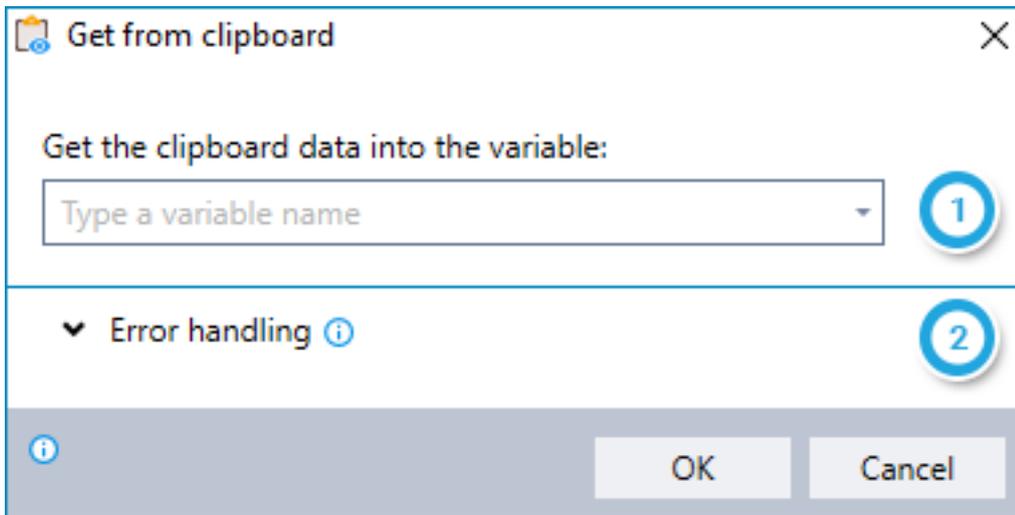
Retrieve the contents of the Windows clipboard and place them into a new or existing variable.



NOTE

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

Using the GET FROM CLIPBOARD command

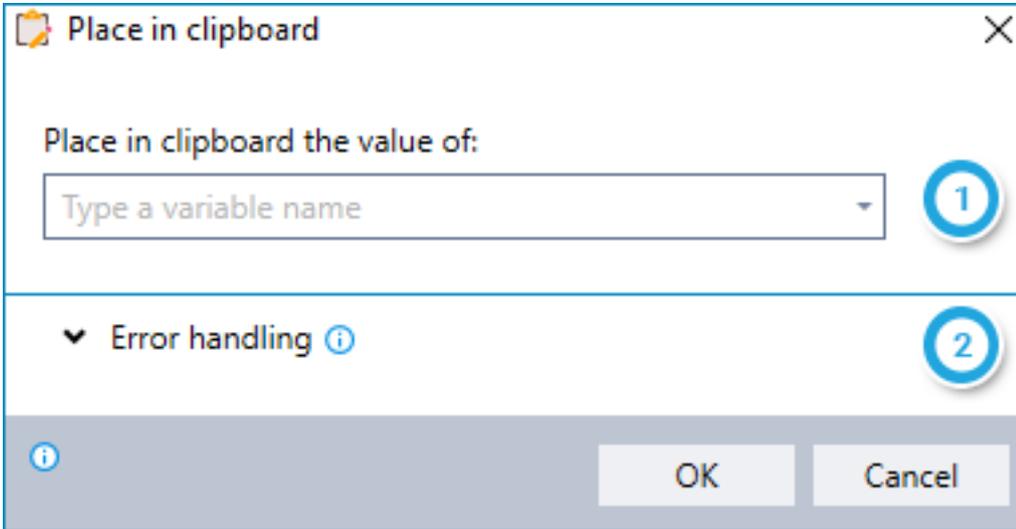


- 1 Enter the name of the variable into which you'd like to place the contents of the clipboard
- 2 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

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

Copy Active Field Value

Copy the value from the active field of 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.

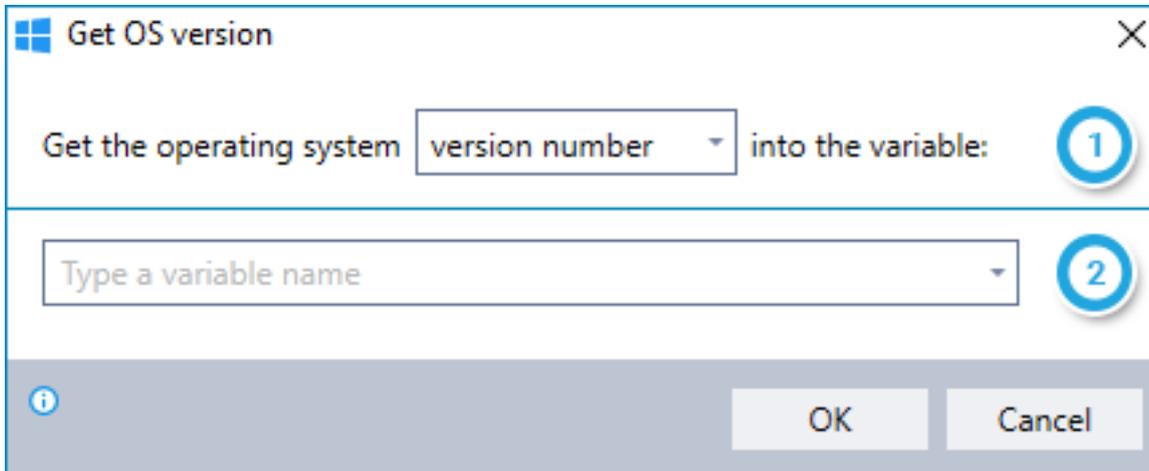
Using the COPY ACTIVE FIELD VALUE command

- 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 the wizard 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

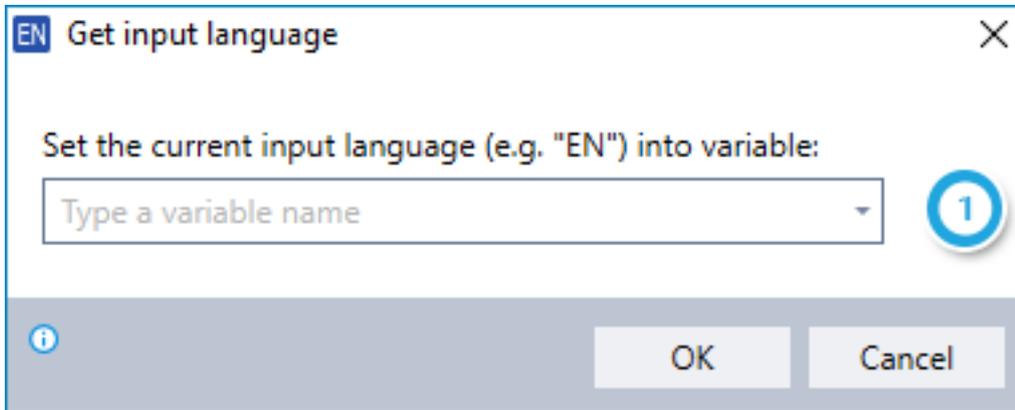


- 1 Choose whether you'd like to retrieve the Windows version number or version name
- 2 Enter the name of the variable into which you'd like to place the result

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

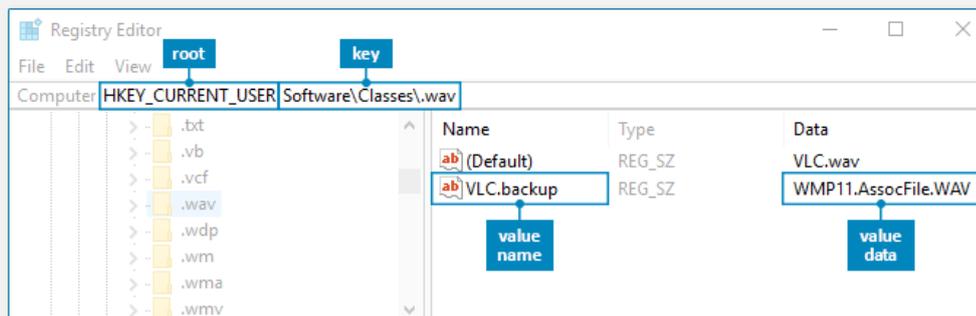
The anatomy of the Windows Registry

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.

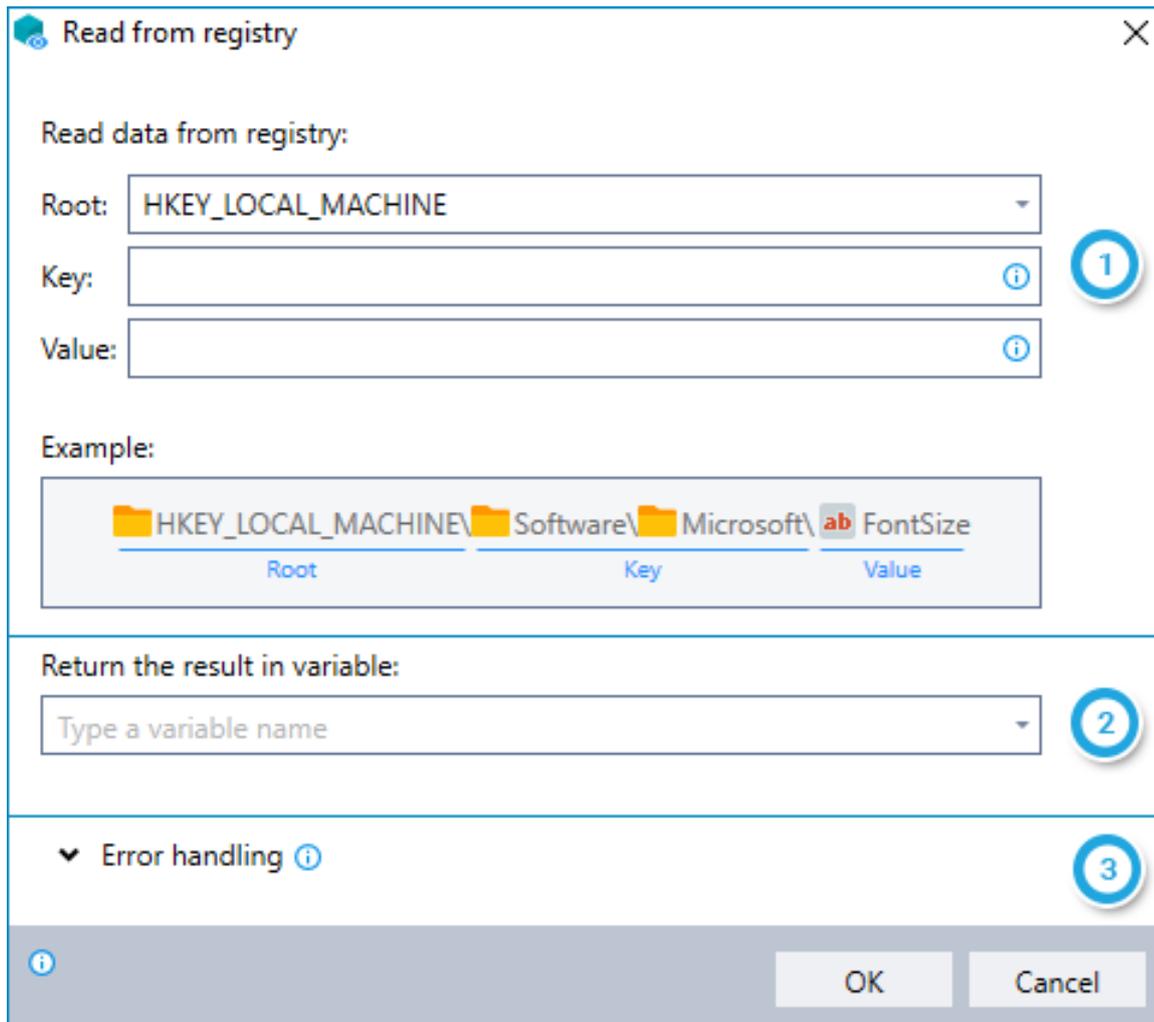
Registry Editor is the Windows component that allow you to view and edit the Registry. To access it, type `Run` from the Windows Start Menu, then `regedit` in the dialog box .

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**, the wizard 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 the wizard 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**, the wizard 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**), the wizard will remove the specified **KEY**

2 Instruct the wizard 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**, the wizard 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 the wizard 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 wizard 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

The screenshot shows a dialog box titled "Get environment variable". It has a close button (X) in the top right corner. The dialog is divided into two sections. The first section is labeled "Get the value of the environment variable:" and contains a text input field with an information icon (i) to its right. A blue circle with the number "1" is overlaid on the information icon. The second section is labeled "Return the result in variable:" and contains a dropdown menu with the placeholder text "Type a variable name" and a downward arrow. A blue circle with the number "2" is overlaid on 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 name of the environment variable for which you would like to retrieve the value
- 2 Enter the name of the wizard 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.



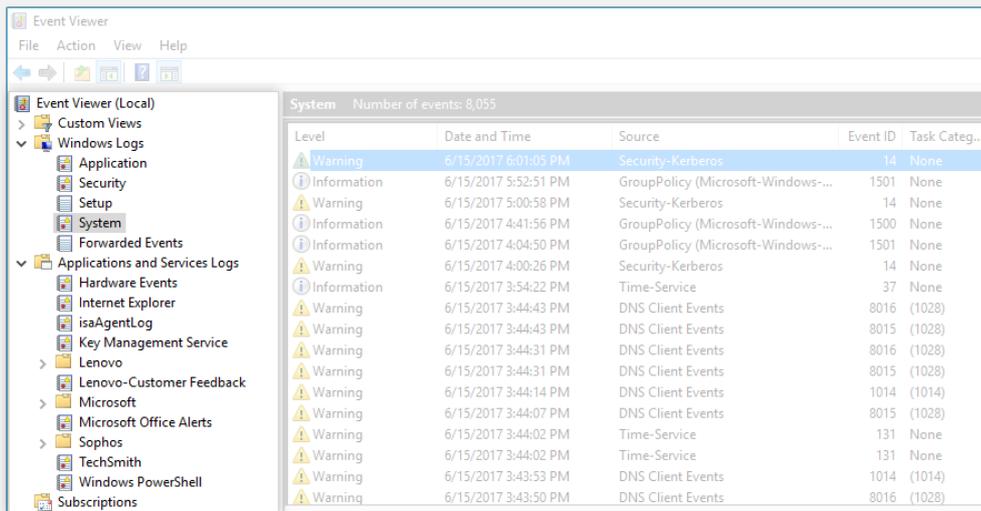
NOTE

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

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

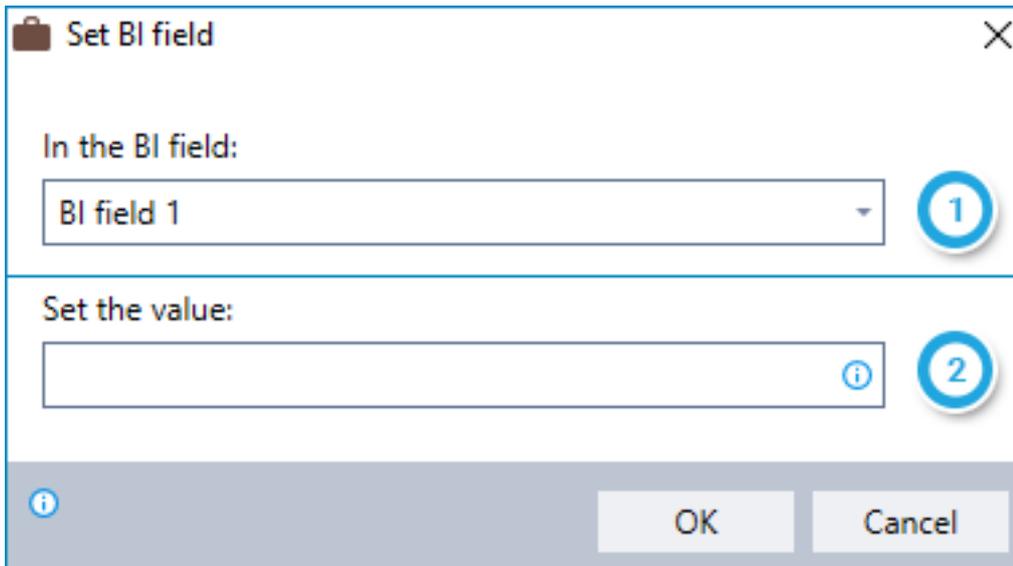
3 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Set BI Field

Report custom Business Intelligence (BI) information into the Kryon 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 Kryon Database* section of the Kryon Studio User Guide.

Using the SET BI FIELD command



The screenshot shows a dialog box titled "Set BI field". It has a close button in the top right corner. The dialog is divided into two sections. The first section, "In the BI field:", contains a dropdown menu with "BI field 1" selected and a circled "1" to its right. The second section, "Set the value:", contains an empty text input field with an information icon to its right and a circled "2" to its right. At the bottom of the dialog, there is an information icon, "OK", and "Cancel" buttons.

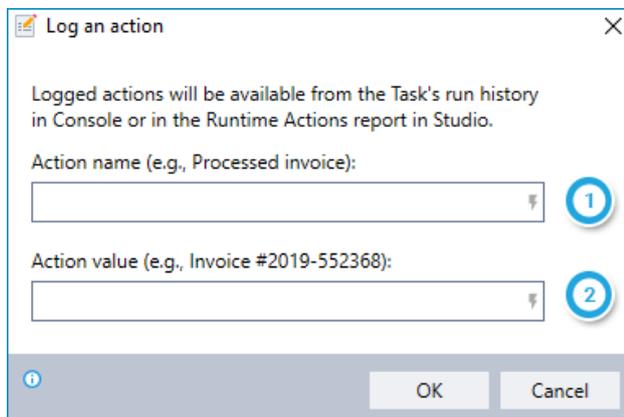
- 1 Select the BI field into which you would like to place the specified value
- 2 Enter the value you would like to place in the selected field (can be free text and/or values copied from different variables)

Log an Action

Log the occurrence of any custom activity and its value during the running of a wizard. The action is logged upon reaching the point in the wizard where you have placed the **LOG AN ACTION** command.

- For an attended automation, you can later access this information by generating the **RUNTIME USER ACTIONS** report from the **Kryon Report Generator**.
- For an unattended automation, you can access the data through Kryon Console in the **Run History** for the relevant task/trigger.

Using the LOG AN ACTION command



- 1 Enter the name to assign to the action to be logged
- 2 Enter the value to assign to the action to be 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 **LOG AN ACTION** command to log that the deletion occurred, along with the names of the files deleted.

Query XML

Use an XPATH 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 blue 'XPATH Tester...' link to its right and an information icon.
- 3** Radio button options: 'First result', 'Last result', and 'All results' (which is selected). Below these is a 'Delimiter' field containing a comma 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 a small 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Query JSON

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



NOTE

What is JSONPATH?

JSONPath (similar to [XPath for XML](#)) is a syntax or language used for finding elements in a JSON document.

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

Using the QUERY JSON command

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

- JSON Source:** A dropdown menu with the text 'Select a variable' and a blue circle with the number 1 next to it.
- JSONPATH query:** A text input field with a blue circle with the number 2 next to it. A link labeled 'JSONPATH Tester...' is visible to the right of the field.
- Result Selection:** Three radio button options: 'First result', 'Last result', and 'All results' (which is selected). A blue circle with the number 3 is next to the 'All results' option. Below these options is a 'Delimiter' field containing a comma character.
- Return values in variable:** A dropdown menu with the text 'Select a variable' and a blue circle with the number 4 next to it.
- Error handling:** A dropdown menu with a downward arrow and the text 'Error handling' and an information icon. A blue circle with the number 5 is next to it.

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 JSON data from which to extract information
- 2 Enter your JSONPath 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 to place the results
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Call REST API Method

Interact with internal and external REST APIs.

Using the CALL REST API METHOD command

Call REST API method

Method: **1** URL: **2**

POST

Headers: **3** + Add header Credentials: **4**

Name	Value

None Manual From Vault

Body (optional): **5**

Timeout: **6** 20 Seconds

Return status code in variable: **7** Return response content in variable: **8**

Type a variable name

Type a variable name

OK Cancel

- 1** Select the request method. The available methods are:
 - GET: used to get a resource from a server
 - POST: used to create a new resource on a server
 - PUT: used to update a resource on a server
 - DELETE: used to delete a resource from a server
- 2** Enter the URL (i.e., *endpoint*) to which the request should be sent (root-endpoint/path/query parameters)

- 3 Enter the headers to be used in the request:
 - To add a header, click the **+ Add header** link
 - Enter the header property name (using free text and/or values copied from different variables)
 - Enter the header property value (using free text and/or values copied from different variables)

To delete an existing header, click its **×** button
- 4 If the API requires authentication, choose to:
 - Enter them manually (in the **Username** and **Password** fields that appear when this option is selected); **or**
 - Retrieve them from the Kryon Credentials Vault (and select the relevant user from the vault)
- 5 (Optional) When using the POST or PUT method, enter the information you want to send to the server (i.e., *body*)
- 6 Enter the maximum time the wizard should wait for the API server to process the request
- 7 Choose whether to return a status code received from the API server in a variable; **and** Enter the name of the variable in which to return it
- 8 Choose whether to return response content received from the API server in a variable; **and** Enter the name of the variable in which to return it

CHAPTER 13: Email Commands

In this chapter:

Send Email Message	229
Get Email Messages	236
Email: Get Data	243
Email: Move to Folder	244
Email: Forward	246
Email: Reply	248
Email: Delete	249
Email: Mark as Read/Unread	250
Email: Save Attachments	251
Email: Save Message	253

Send Email Message

Send an email message.

Using the SEND EMAIL MESSAGE command

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 Kryon credentials vault; *or*
- Enter a credential by variable; *or*
- 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 Kryon credentials vault; *or*
- Enter manually

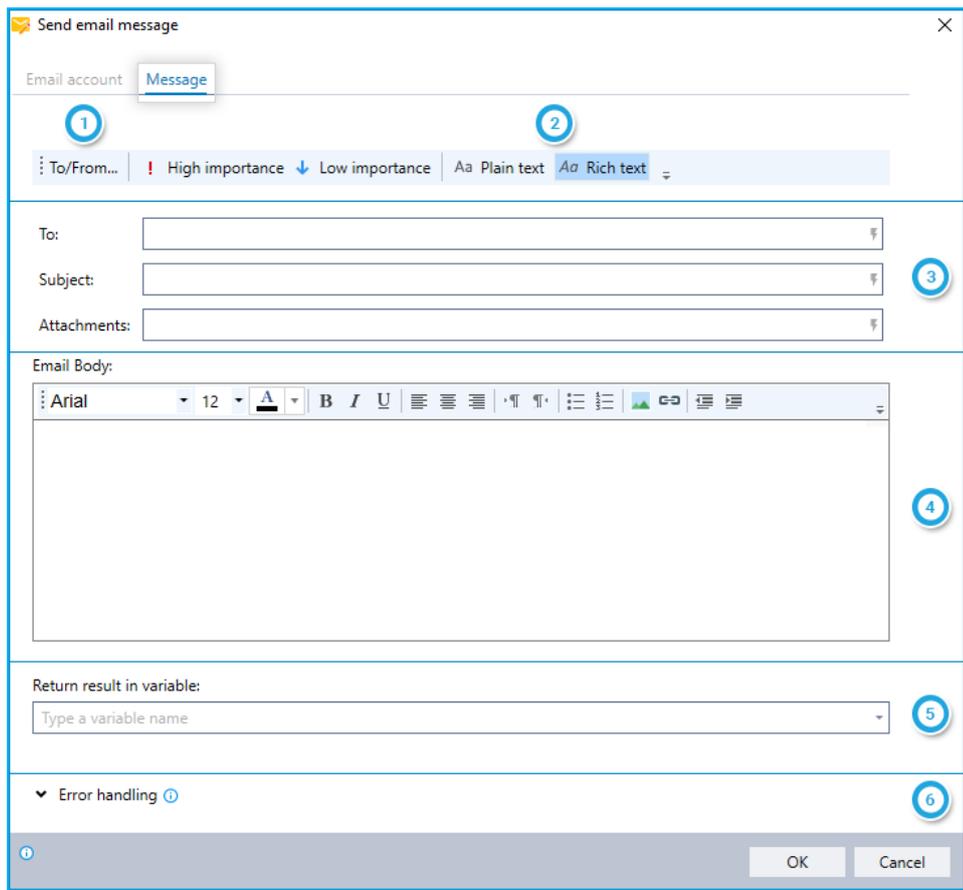
3 Mailbox:

- 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

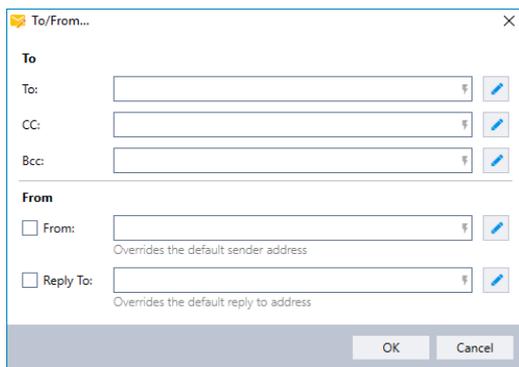
Note: The mailbox data entered will provide the default **FROM:** and **REPLY TO:** addresses for the message. These addresses can be overridden by utilizing [advanced address options](#) in the **Message** tab.

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

Message tab



1 (Optional) Click the **To/From...** 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



Enter the body of the email



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



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.



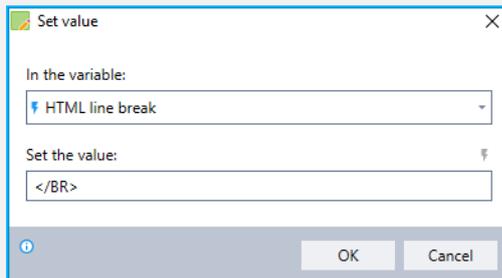
TROUBLESHOOTING TIP

Line breaks in rich text emails

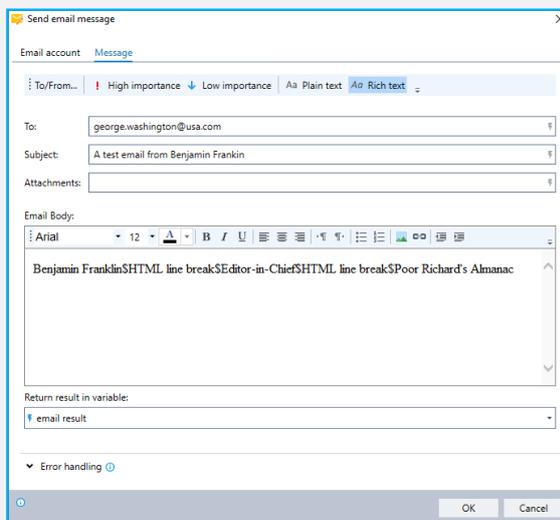
When sending an HTML/rich text email that contains "hard" line breaks, create a special character variable named, for example, **HTML line break**, and set its value to `</BR>`. Then use this variable to replace actual line breaks in your email text.

Example:

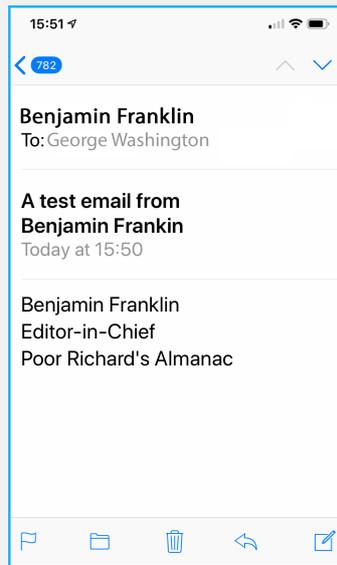
1. Set a variable to define the line break



2. Use this variable to replace the actual line breaks in your rich text-formatted email



Result:





TIP

Using Credentials by Variable

The credential by Variable functionality allows you to create a dynamic connection to the credentials vault. The credential variable is associated with credential display name(s).

Since the credential variables are dynamic, if you export a wizard to another environment, make sure that the credentials associated with the variable are defined in the new environment's credential vault.

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.

Email account tab

The settings available on the **Email account** tab vary for Exchange and IMAP/POP3 servers.

Exchange servers

Get email messages

Email account Get messages

Server 1

Email Server Type:
Exchange (EWS: 2007 and above)

Server:

Credentials 2

From credentials vault:

Enter credential by variable:

Enter manually:

Username:

Password:

Domain:

Mailbox 3

User's default mailbox

Shared mailbox:

Test 4

▼ Error handling ⓘ

OK Cancel

- 1 Enter the settings for your email server
- 2 Email server login credentials:
 - Enter from the Kryon credentials vault; **or**
 - Enter a credential by variable; **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

IMAP/POP3 servers

The screenshot shows a dialog box titled "Get email messages" with a close button (X) in the top right corner. The dialog has two tabs: "Email account" (selected) and "Get messages".

Server 1

Email Server Type: IMAP (dropdown menu)

Server: [text input field]

Use SSL Port: [text input field]

Auto (dropdown menu)

Test 3

Credentials 2

From credentials vault: [dropdown menu] [Add new](#)

Enter credential by variable: [text input field]

Enter manually: Username: [text input field]
Password: [text input field]

▼ Error handling ⓘ

OK Cancel

- 1 Enter the settings for your email server
- 2 Email server login credentials:
 - Enter from the Kryon credentials vault; **or**
 - Enter a credential by variable; **or**
 - Enter manually



(Optional) Test your email account settings



NOTES

IMAP and POP3 servers don't send mail!

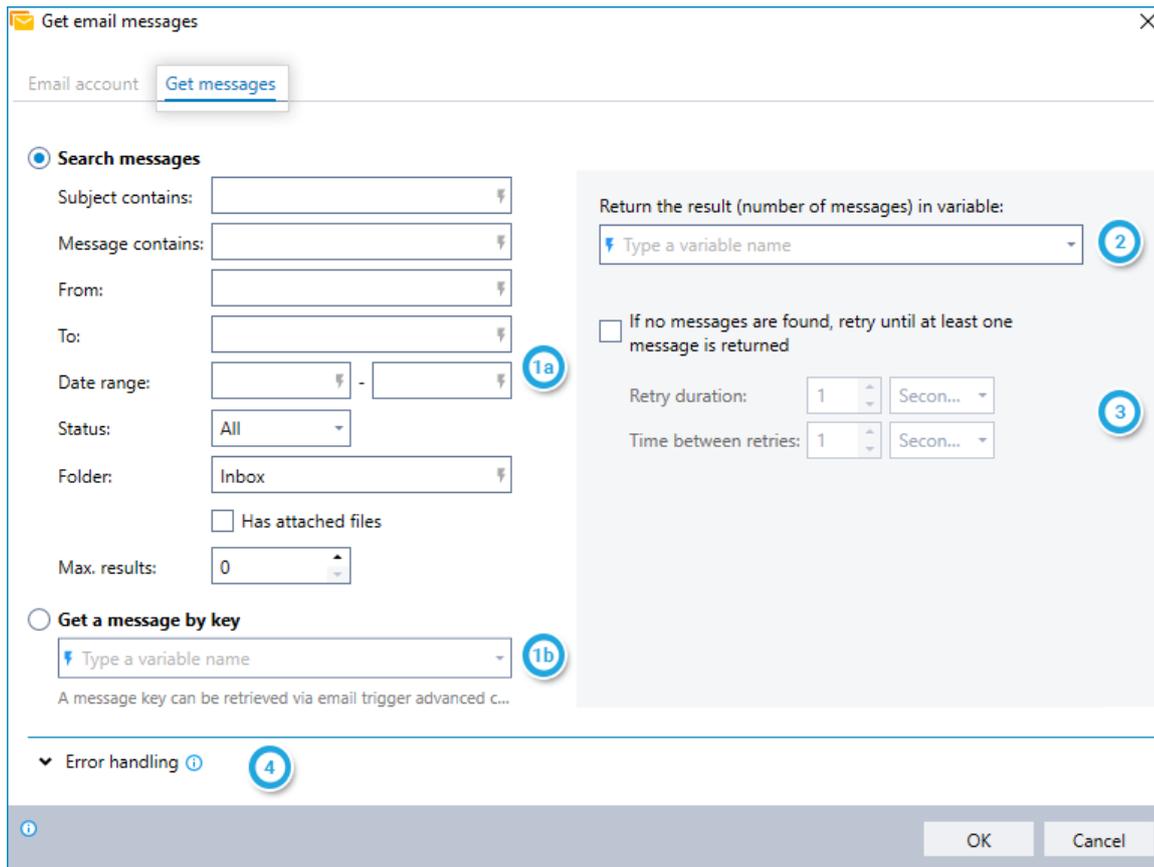
IMAP and POP3 are internet mail protocols designed for retrieving emails, not for sending. As a result, if your **Email Server Type** is set to IMAP or POP3, the following commands are not supported within the **GET EMAIL MESSAGES** command:

- **EMAIL: FORWARD**
- **EMAIL: REPLY**

POP3 servers don't support all of Kryon's capabilities

1. When [defining the filter](#) for the specific mail messages that will be retrieved, POP3 servers do not support searching by the following fields:
 - Message contains
 - Status
 - Folder
2. POP3 servers do not support the following commands within the **GET EMAIL MESSAGES** command:
 - **EMAIL: MOVE TO FOLDER**
 - **EMAIL: MARK AS READ/UNREAD**

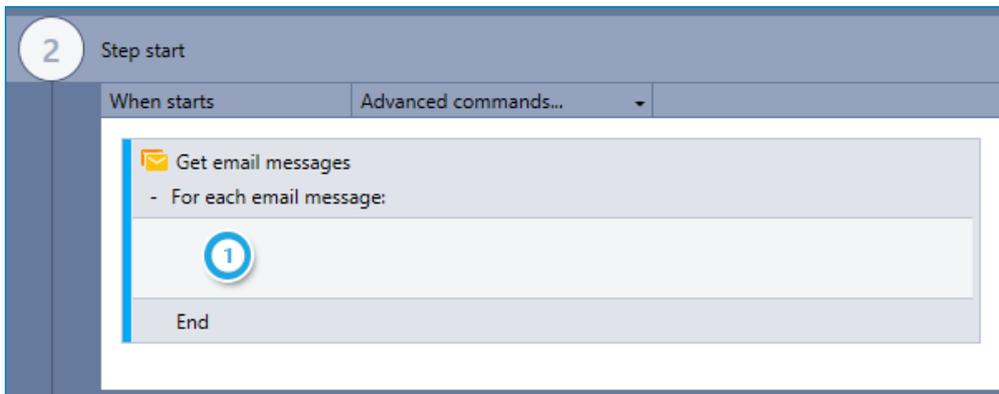
Get messages tab



- 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.](#))
- NOTE:** To define search **From**, enter the email address (for example, `gilgolan@yahoo.com`). To define search **To**, enter the *display name only*. (for example, if the complete email is `Gil Golan <gilgolan@yahoo.com>`, enter `Gil Golan`)
- 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING.](#)

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

The wizard performs 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).



TIP

Using Credentials by Variable

The credential by Variable functionality allows you to create a dynamic connection to the credentials vault. The credential variable is associated with credential display name(s).

Since the credential variables are dynamic, if you export a wizard to another environment, make sure that the credentials associated with the variable are defined in the new environment's credential vault.

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 the wizard 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.

This command is not supported by POP3 servers. For additional details, see [POP3 servers don't support all of Kryon's capabilities](#).



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)

Warning: This command is supported only by **Exchange** and **IMAP** email servers.

▼ Error handling ⓘ

OK Cancel

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

Email: Forward

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



NOTES

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

This command is not supported by IMAP/POP3 servers. For additional details, see [IMAP and POP3 servers don't send mail!](#)

Using the EMAIL: FORWARD command

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

- Forward to:** A text input field with a dropdown arrow on the right, marked with a blue circle containing the number 1.
- From:** A text input field with a dropdown arrow on the right, marked with a blue circle containing the number 2.
- Message text (will be added to original message):** A large text area with a dropdown arrow on the right, marked with a blue circle containing the number 3.
- Warning:** A yellow warning triangle icon followed by the text: "This command is supported only by **Exchange** email servers."
- Error handling:** A dropdown menu with a downward arrow and a help icon (i), marked with a blue circle containing the number 4.

At the bottom of the dialog, there is a grey bar containing an information icon (i) on the left and two buttons: "OK" and "Cancel".

- 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Email: Reply

Reply to a message retrieved via the [GET EMAIL MESSAGES](#) command.



NOTES

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

This command is not supported by IMAP/POP3 servers. For additional details, see [IMAP and POP3 servers don't send mail!](#)

Using the EMAIL: REPLY command

1

2

3

- 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Email: Delete

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



NOTE

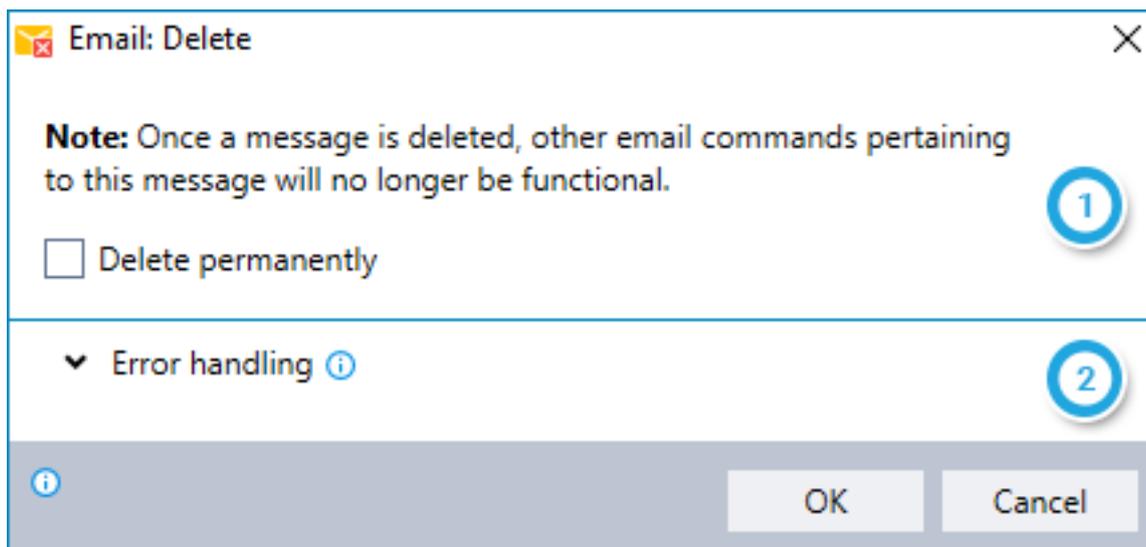
This command can be used only within a [GET EMAIL MESSAGES](#) container.



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 the wizard 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.

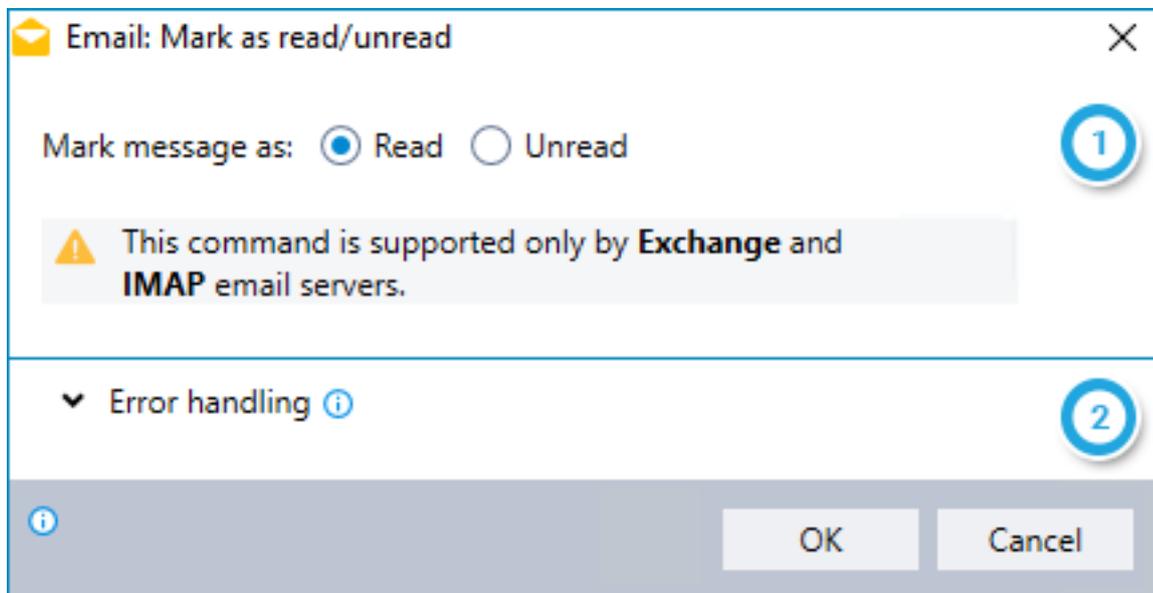


NOTE

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

This command is not supported by POP3 servers. For additional details, see [POP3 servers don't support all of Kryon's capabilities](#).

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 the wizard 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)

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

Email: Save Message

Save a message retrieved via the [GET EMAIL MESSAGES](#) command and place the saved path/filename into a new or existing variable.



NOTE

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

Using the EMAIL: SAVE MESSAGE command

- 1 Enter the **full path** of the folder to which you would like to save the message file; *and* Indicate whether the saved file should replace (overwrite) existing an file with the same filename if it exists
- 2 Select the file format in which to save the message
- 3 Enter the name of the variable in which to place the path/filename of the saved file
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 14: External Program Commands

In this chapter:

Run Program	255
Run .NET Plugin Method	257
Call Web Service Method	259
Run Script	260
Open URL Link (beta)	262
Run curl Command	263
Get Web Page HTML	264
Run JavaScript on Page	265

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. Callout 1 points to the 'Browse...' button.
- Command line parameters:** A text input field. Callout 2 points to the input field.
- Return result in variable:** A dropdown menu with 'Type a variable name' selected. Callout 3 points to the dropdown.
- Options:** Three checkboxes: 'Hide program window', 'Wait for the program to finish', and 'Return program window handle in variable:'. Callout 4 points to the 'Return program window handle in variable:' checkbox.
- Return program window handle in variable:** A dropdown menu with 'Type a variable name' selected, followed by '(Max. wait' and a text input field, and 'ms.)'. Callout 4 also points to this section.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

- 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 Kryon's capabilities.

There are two plugin types:

1. **Embedded:** a wizard is downloaded to the client with the .net plugin dll is embedded within it
2. **Local:** The .NET plugin is deployed to all relevant machines by IT. (The Runtime folder is the folder where IT has placed it)



NOTE

The run time folder is the absolute path on a local drive where IT has placed the dll on the machine. Don't use relative paths to define the runtime folder.

For additional information see the *Kryon Plugin Development* section of the Kryon Studio User Guide.

Name	Type	Direction	Value
userID	String	in	8057201844
folderID	String	in	75749113961
pathToSave	String	in	C:/BoxTarget/

1 Select the .NET plugin to run:

- Select **Embedded** if the .NET plugin dll is embedded within the wizard. Select the plugin name from the list; **or**

- Select **Local** if the .NET plugin was deployed to the machine by IT. Set the absolute path to the plugin in the local drive in the **Runtime folder** field

2 Specify the method to run. For more information, see the *Kryon Plugin Development* section of the Kryon Studio User Guide

3 The Parameters list populates with parameters' name, type and direction. Enter the parameter **Values**

4 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

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:

Credentials No credentials Manual From vault

Parameters:

	Name	Type	Value
+			Enter new parameter name

Return result in variable:

Service timeout: seconds

▼ Error handling ⓘ

ⓘ
OK
Cancel

Run Script

Instruct the wizard 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

The screenshot shows the 'Run script' dialog box with the following elements and numbered callouts:

- 1**: A large text area for entering the script code.
- 2**: A text input field for entering parameters.
- 3**: A dropdown menu for selecting the script language, currently set to 'VbScript'.
- 4**: A spinner control for setting the timeout, currently at '60' seconds.
- 5**: A checkbox labeled 'Use Custom Encoding' with an empty dropdown menu below it.
- 6**: A dropdown menu for selecting a variable to store the result, currently showing 'Select a variable'.

Additional elements include a 'Load from file...' button below the script code area, an 'Information' section at the bottom stating 'Script parameters are space separated', and 'OK' and 'Cancel' buttons at the bottom right.

- 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
 - For Python, you are given the option to specify which version of Python will be used to run the script: the Kryon-installed version or a version installed on the client machine
- 4 Specify a timeout (i.e., if the script does not begin to run within this time frame, the wizard will move on)
- 5 Tick this checkbox to use custom character encoding, then select the encoding set you wish to use
- 6 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

Open URL Link (beta)

Opens a URL link using a selected browser in a selected location (current tab/new tab/new window).

Using the OPEN URL LINK command

- 1 By default, the **Use default browser** check box is selected and the default browser appears in the field. Deselect to change the browser by entering the browser executable file name in the field or clicking the browse button.
- 2 Open a browser to a specific URL. The URL can be dynamic, based on a variable.
- 3 The URL can be opened in a:
 - o New window
 - o New tab
 - o In the active tab
- 4 Indicate additional options:
 - o Whether to hide the window
 - o Whether to wait for the program to finish before the wizard moves on
 - o Whether to return the program's window handle in a variable

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.

The screenshot shows a dialog box titled "Run curl command" with a close button (X) in the top right corner. The dialog contains the following text and fields:

- Instruction: "Use the curl command to transfer data from or to a server, using one of the supported protocols."
- Label: "curl command:" followed by a text input field containing the text "curl".
- Label: "Return result in variable (optional):" followed by a dropdown menu with the placeholder text "Type a variable name".
- Label: "Return exit code in variable (optional):" followed by a dropdown menu with the placeholder text "Type a variable name".
- Text: "curl offers a busload of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL connections, cookies, file transfer resume, Metalink, and more."
- Links: "Online documentation | Manual (offline) | How to use (offline)".
- Buttons: "OK" and "Cancel" at the bottom right.
- An information icon (i) is located at the bottom left of the dialog.

Get Web Page HTML

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



NOTE

Make sure it's the right browser!

This command supports only:

- Internet Explorer; *and*
- Chrome (when the Kryon Connector Chrome extension is installed). To learn more, see [Kryon Connector](#).

So make sure it's one of these that's open when the wizard is run.

Using the GET WEB 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Run JavaScript on Page

Instruct the wizard to execute JavaScript on a web page.



NOTE

Make sure it's the right browser!

This command supports only:

- Internet Explorer (IE5 and above); *and*
- Chrome (when the Kryon Connector Chrome extension is installed). To learn more, see [Kryon Connector](#).

So make sure it's one of these that's open when the wizard is run.

Using the RUN JAVASCRIPT ON PAGE command

Run javascript on page

Javascript Code:

Load from file...

To return a result from the javascript code, explicitly call `LeoSetOutputVariableValue()`
+Add function call

Set the result in variable:

Type a variable name

OK Cancel

- 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  and  are relevant only for a script that is designed to return a result.

CHAPTER 15: Database Commands

In this chapter:

Execute SQL Query	268
Monitor Database Changes	271

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 Kryon Admin or defined within the Advanced Command itself.

Using the EXECUTE SQL QUERY command

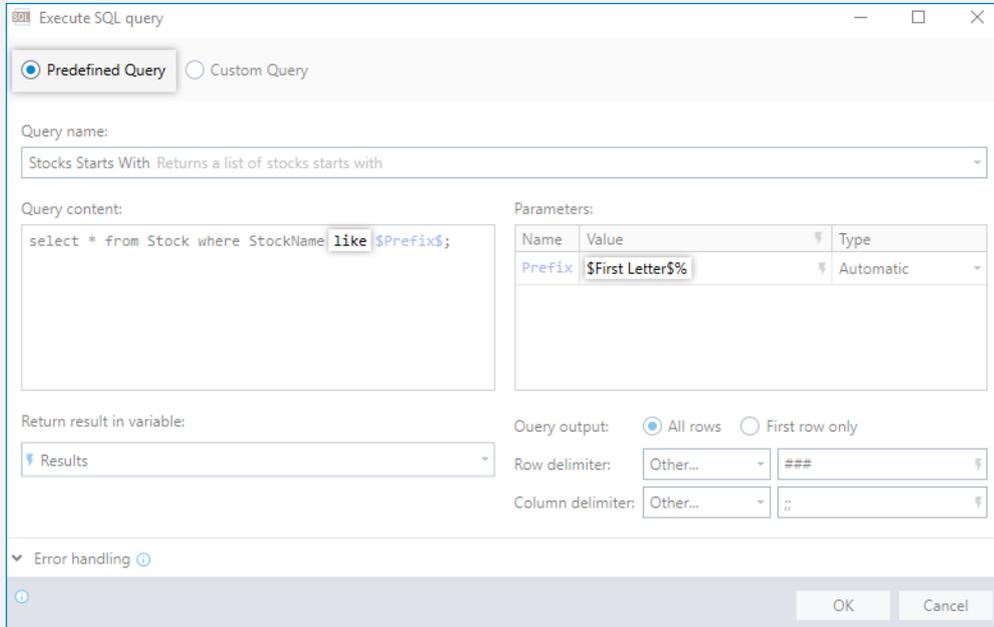
Predefined query

- 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 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

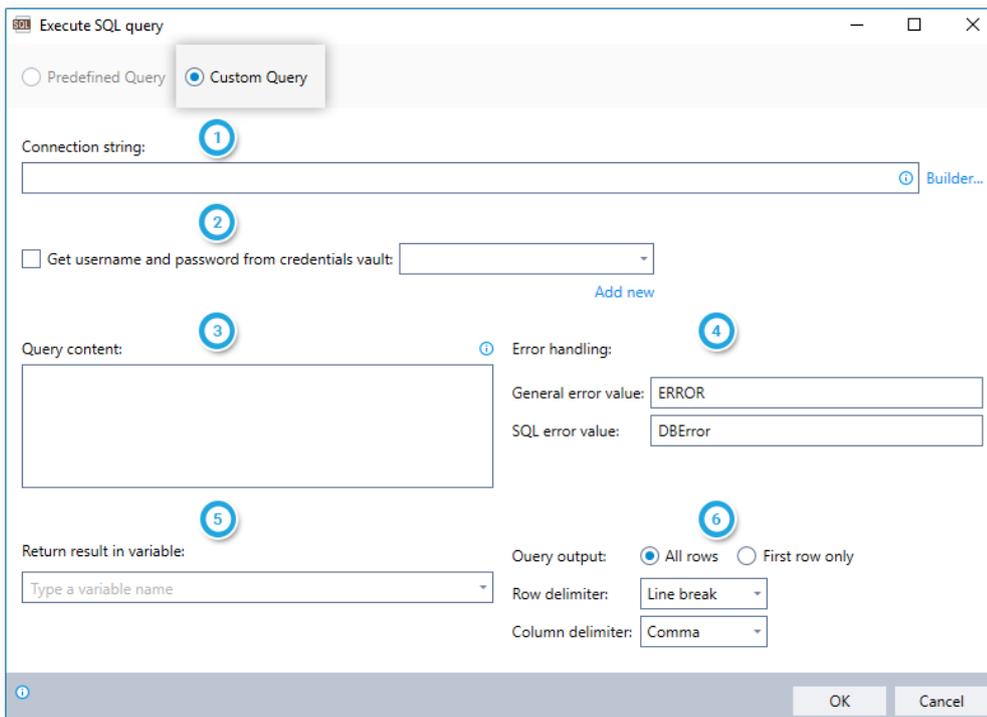


CAUTION
Syntax for "like" operation in predefined queries

When using a "like" operation in a predefined query, include a % at the end of the value of provided parameter.



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 Kryon Credentials Vault
- 3 Enter your query using standard SQL syntax
- 4 Instruct the wizard 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 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 Kryon 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 the wizard 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 16: Credentials Vault Commands

In this chapter:

Insert Username Into Active Field	274
Insert Password Into Active Field	275
Generate New Password	276
Revert Password	277

Insert Username Into Active Field

Place a username from the Kryon 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 the wizard 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Insert Password Into Active Field

Place a password from the Kryon 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 the wizard 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Generate New Password

Create a new password in the Kryon 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Revert Password

Overwrite the current password in the Kryon 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 the wizard 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	279
Get Automation Task Status	282
Get File Trigger Input	284
Get Folder Trigger Input	285
Get Email Trigger Input	286
Get Database Trigger Input	287

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

1 Enter a task name and set queue priority

2 Enter additional information in 3 tabs:

Wizard

Parameters

Robot

Select a wizard:

By wizard ID:

By custom ID:

Wizard

Parameters

Robot

Parameters: + Add

Variable	Value

Wizard

Parameters

Robot

First available robot

First available robot from the group:

A specific robot:

Wizard:

Select the wizard to assign to a robot (the "task wizard"):

- From the Wizard Catalog
- By the wizard ID automatically assigned to the task wizard; **or**
- By the custom ID created for the task wizard

Parameters:

(Optional) Enter parameters (i.e., the initial values of some or all of the task wizard's variables)

Robot:

Choose to which robot the task should be assigned:

- The first available robot
- The first available robot from a specific group (select from list); **or**
- A specific robot (select from list)
 - identified by a specific machine name/username/friendly name combination

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 **Kryon Robot**)
- 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 the wizard 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

- 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

Code	Status
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 the wizard 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 Kryon 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

Get file trigger input

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

File path

Type a variable name

Use advanced commands to read file contents by its path.

File action

Type a variable name

Expected values: NEW / MODIFIED / DELETED

OK Cancel

- 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 Kryon 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 Kryon Console)
- The message key retrieved is used in conjunction with the [GET EMAIL MESSAGES](#) command

Using the GET EMAIL TRIGGER INPUT command

Get email trigger input

If this wizard is initiated by an email trigger, retrieve the email message information into variables:

Email message key

Type a variable name

Use "Get email messages" command to retrieve message information by a given key.

OK Cancel



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 Kryon 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:

Copy from Excel	290
Paste to Excel	294
Delete from Excel	298
Excel Worksheet Actions	301
Excel Row Actions	302
Excel Column Actions	304
Convert Excel to CSV	306
Create New Excel File	307
Query From Excel	308
Run Macro	310



NOTE

Which Excel versions are supported?

As a general rule, Kryon supports desktop (i.e., non-mobile) versions of Excel that are within Microsoft's *Mainstream Support* or *Extended Support* periods. Currently, this includes Excel 2010 and higher. For additional details, see [Microsoft Lifecycle Policy \(Excel\)](#) on the Microsoft support website.

Copy from Excel

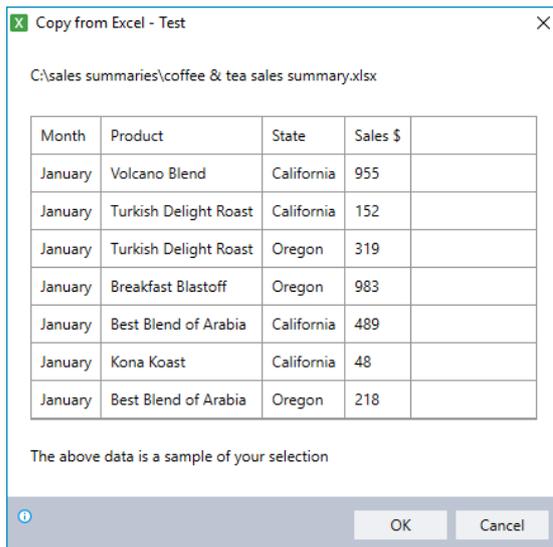
Copy specified cell values from an Excel file and place them into a new or existing variable.

Using the COPY FROM EXCEL command

The screenshot shows the 'Copy from Excel' dialog box with the following elements and callouts:

- 1**: 'Browse...' button in the 'Where to copy from:' section.
- 2**: 'Password' field in the 'Where to copy from:' section.
- 3**: 'Worksheet' dropdown menu and 'Select in Excel' button in the 'What to copy' section.
- 4**: 'Test' button in the 'Copy method' section.
- 5**: 'Actual Values' dropdown menu in the 'Copy method' section.
- 6**: 'Select a variable' dropdown menu in the 'How to save' section.
- 7**: 'Error handling' dropdown menu in the bottom section.

- 1** Click the **Browse...** button and navigate to the Excel file from which you would like to copy cell values
 - **NOTE:** All other options and fields in the dialog box will be enabled once you have selected a file
- 2** If the selected Excel file is password protected, enter the password
- 3** Specify the worksheet and cells from which to copy values, either:
 - **by selecting them directly in Excel;** *or*
 - **by entering them manually** (this method can be particularly useful when you want to use variable values to select cells)
- 4** (Optional) Click the **Test** button to display a sample of the data that will be copied based on your selections



5 Select whether to copy the values as **ACTUAL VALUES** or **FORMATTED TEXT**

6 Specify options for saving the data:

- Enter the name of the variable into which you'd like to place the copied values; **and**
- Enter the delimiters to use to separate each column and row in the returned data

7 Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.



TIP

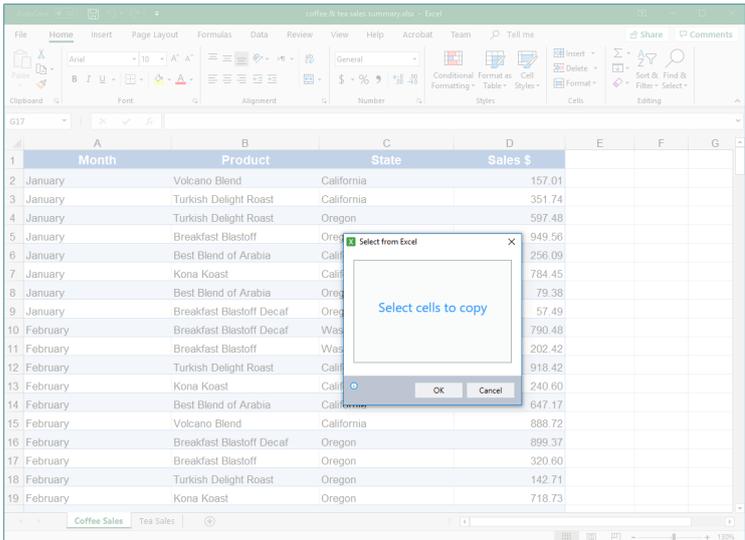
Should I use ACTUAL VALUES or FORMATTED TEXT as the copy method?

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.)

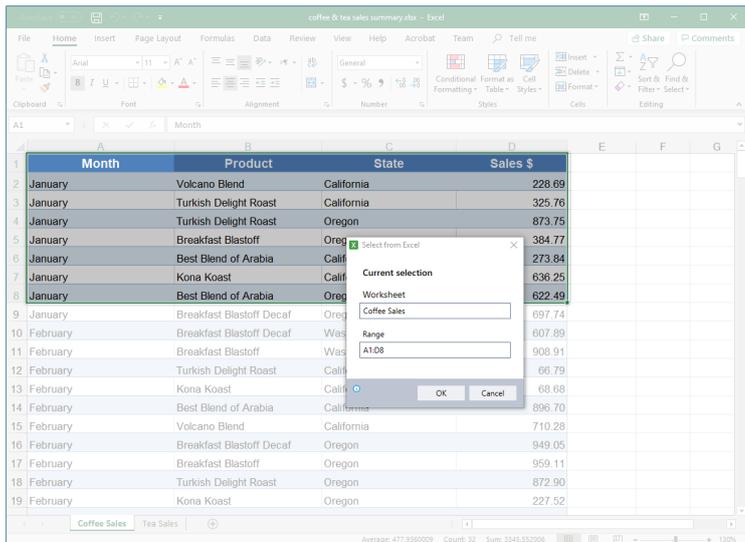
For reading all other cells, use the **ACTUAL VALUES** option.

Selecting worksheet & cells directly in Excel

1. Click the  button to invoke the **Excel selector**
2. The **Excel selector** will appear with the selected Excel file open behind it



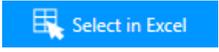
3. Click directly in the Excel file and select the worksheet (tab) and cells you want to copy from
 - The worksheet name and cell range you have selected will appear in the selector window's **Current selection** fields



TIP

If the selector window interferes with viewing or selecting the cells you need, simply drag the window to a more convenient location on the screen.

4. When you are satisfied with your selection, click **OK**
5. The main **COPY FROM EXCEL** window will appear, with the **Worksheet** and **Range** fields populated by your selections

6. If you want to make modifications to your selections, you can do so either [manually](#) or by repeating the  process

Entering worksheet & cells manually

1. Specify the worksheet (tab) in which the cells you want to copy are located
2. Choose one of three methods for specifying the cells from which you want to copy:
 - i. **Range** – Designate the range using "standard" Excel format (e.g., the range from **Column A Row 1** to **Column D Row 8** is designated as A1 : D8)
 - ii. **From/To** –
 - Designate the starting column and row, using numbers and/or letters (e.g., cell **C5** could be designated as Column 3, Row 5 or Column C, Row 5)
 - (Optional) Designate the ending column and row, using numbers and/or letters
 - If you don't designate an ending column, the robot will copy data from all columns until it detects it has reached the last column in the range (by reading 20 consecutive empty columns)
 - If you don't designate an ending row, the robot will copy data from all rows until it detects it has reached the last row in the range (by reading 20 consecutive empty columns)
 - iii. **Entire worksheet** – The robot will copy data from the whole worksheet until it detects it has reached the last row and column in the sheet (by reading 20 consecutive empty rows/columns)

Paste to Excel

Paste value(s) or formula(s) into an existing Excel file. This command allows you to paste either:

- a table (i.e., multiple values) into multiple cells; *or*
- an identical single value into multiple cells

Using the PASTE TO EXCEL command

The screenshot shows the 'Paste to Excel' dialog box with the following elements and callouts:

- 1**: Radio buttons for 'Paste a table' (selected) and 'Paste a single value'.
- 2**: 'Paste the contents of this variable:' dropdown menu.
- 3**: 'How to paste:' dropdown menu set to 'Actual Values'.
- 4**: 'Where to paste:' text field containing a file path and a 'Browse...' button.
- 5**: 'Password:' text field with 'optional' below it.
- 6**: 'Start pasting here:' section with 'Cell address' (Example A2) and 'Column/Row' options.
- 7**: 'Error handling' dropdown menu.

1 Choose whether to paste a table or a single value

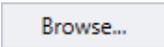
2 When pasting a **table**, enter information about the table you want to paste:

- Enter the variable that contains the table; *and*
- Enter the delimiters that separate each column and row of the table

When pasting a **single value**:

- Enter the variable that contains the value to paste; *or*
- Manually enter the value itself

3 Select whether to paste the data as **ACTUAL VALUES** or as **FORMULAS**

4 Enter a variable, the path and name of the file to paste into or click the  button and navigate to the folder and choose the name of the Excel file within it that you want to paste the data into.

You can select **Create a new file if not exists** to create a new file from the file name you

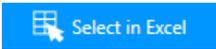
entered.

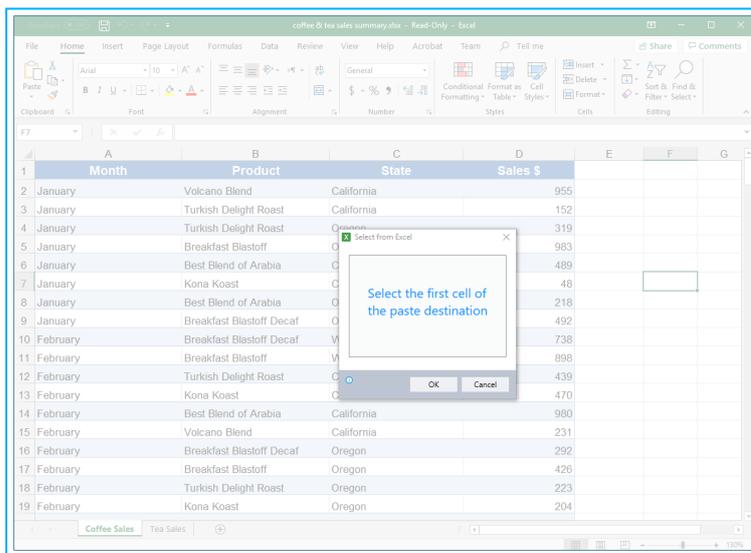
- 5 If the selected Excel file is password protected, enter the password
- 6 Specify the worksheet and cells into which to paste the data, either:
 - by selecting them directly in Excel; **or**
 - by entering them manually (this method can be particularly useful when you want to use variable values to select cells)

NOTE: When you select **Create a new file if not exists**, the worksheet you define here is automatically created also

- 7 Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

Selecting worksheet & cells directly in Excel

1. Click the  button to invoke the **Excel selector**
2. The **Excel selector** appears with the selected Excel file open behind it

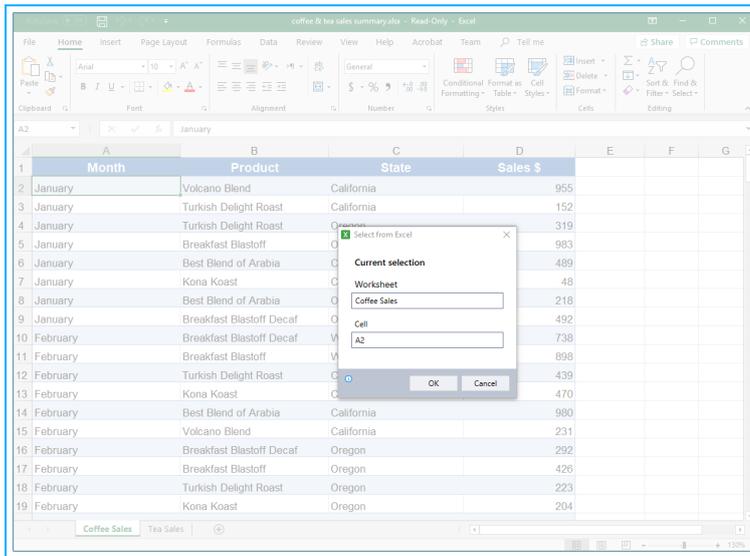


TIP

If the selector window interferes with viewing or selecting the cells you need, simply drag the window to a more convenient location on the screen.

3. If you are pasting a single value, skip to [step 4](#)
If you are pasting a table, click directly in the Excel file and select the worksheet (tab) and first cell of the destination range

- The worksheet name and cell you have selected will appear in the selector window's **Current selection** fields

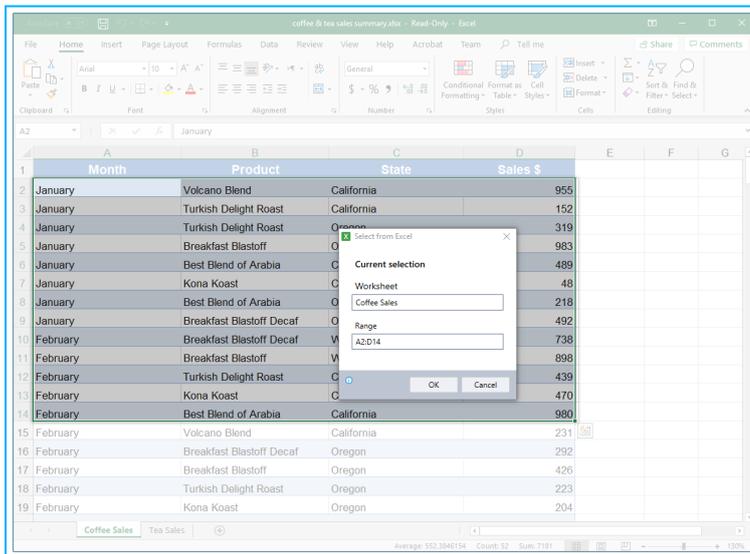


- At runtime, the robot will begin pasting at the cell you specified and automatically write to the number of columns and rows required to paste the entire source table

4. If you are pasting a table, skip to [step 5](#)

If you are pasting a single value, click directly in the Excel file and select the worksheet (tab) and cells you want to copy the value to

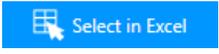
- The worksheet name and cell range you have selected will appear in the selector window's **Current selection** fields



5. When you are satisfied with your selection, click **OK**

6. The main **PASTE TO EXCEL** window will appear, with the relevant fields populated by your

selections

7. If you want to make modifications to your selections, you can do so either **manually** or by repeating the  process

Entering worksheet & cells manually

1. Specify the worksheet (tab) in which the cells you want to copy are located
2. If you are pasting a single value, skip to [step 3](#)

If you are pasting a table, choose one of three methods for specifying where to **start pasting**:

- i. **Cell address** – Designate the first destination cell using "standard" Excel format, e.g., A2 (this is the method used by the **Excel selector**)
 - ii. **Column/Row** – Designate the starting column and row, using numbers and/or letters (e.g., cell **C5** could be designated as Column 3, Row 5 or Column C, Row 5)
 - iii. **Column** – Designate the starting column only, and at runtime, the robot will start pasting at that column in the first available row
3. If you are pasting a table, you're done!

If you are pasting a single value, choose one of two methods for specifying the cells you want to copy to

- i. **Range** – Designate the range using "standard" Excel format (e.g., the range from **Column A Row 1 to Column D Row 8** is designated as A1 : D8)
- ii. **Column/Row** – Designate the **starting column and row** and the **ending column and row**, using numbers and/or letters (e.g., cell C5 could be designated as Column 3, Row 5 or Column C, Row 5)

Delete from Excel

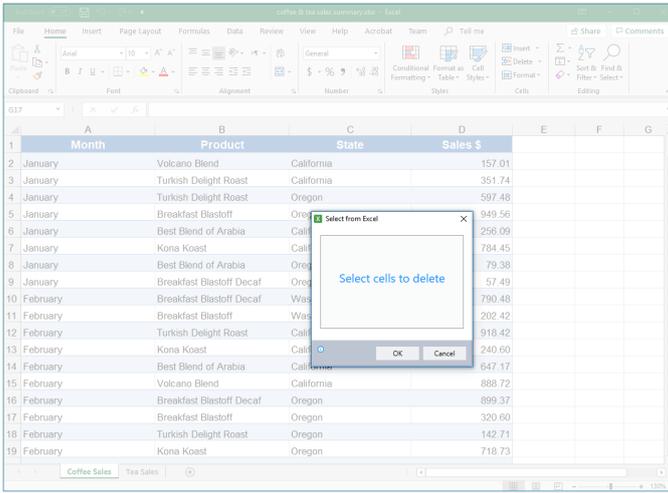
Delete specified cell values from an Excel file.

Using the DELETE FROM EXCEL command

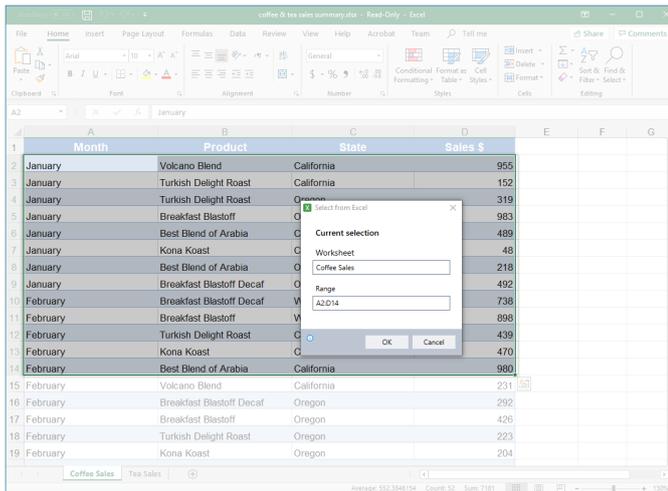
- 1 Click the **Browse...** button and navigate to the Excel file from which you would like to delete cell values
- 2 If the selected Excel file is password protected, enter the password
- 3 Specify the worksheet and cells from which to delete values, either:
 - by selecting them directly in Excel; **or**
 - by entering them manually (this method can be particularly useful when you want to use variable values to select cells)
- 4 Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

Selecting worksheet & cells directly in Excel

1. Click the **Select in Excel** button to invoke the **Excel selector**
2. The **Excel selector** will appear with the selected Excel file open behind it



3. Click directly in the Excel file and select the worksheet (tab) and cells you want to delete from
 - The worksheet name and cell range you have selected will appear in the selector window's **Current selection** fields



TIP

If the selector window interferes with viewing or selecting the cells you need, simply drag the window to a more convenient location on the screen.

4. When you are satisfied with your selection, click **OK**
5. The main **DELETE FROM EXCEL** window will appear, with the **Worksheet** and **Range** fields populated by your selections
6. If you want to make modifications to your selections, you can do so either **manually** or by repeating the  **Select in Excel** process

Entering worksheet & cells manually

1. Specify the worksheet (tab) in which the cells you want to delete are located
2. Choose one of three methods for specifying the cells you want to delete:
 - i. **Range** – Designate the range using "standard" Excel format (e.g., the range from **Column A Row 1** to **Column D Row 8** is designated as A1 : D8)
 - ii. **From/To** – Designate the **starting column and row** and the **ending column and row**, using numbers and/or letters (e.g., cell C5 could be designated as Column 3, Row 5 or Column C, Row 5)
 - iii. **Entire worksheet** – The robot will delete data from the whole worksheet

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 the wizard 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

- 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 the wizard 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

-  Provide additional information as required (fields will vary by the action selected)
-  Instruct the wizard 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 with the following fields and controls:

- 1**: 'Select Excel file:' text label above a text input field with an information icon and a 'Browse...' button.
- 2**: Radio buttons for 'Worksheet name' (selected) and 'Worksheet position'. Below 'Worksheet name' is a text input field with an information icon. Below 'Worksheet position' is a text input field containing '1' with an information icon.
- 3**: 'Action:' text label above a dropdown menu showing 'Get column count'.
- 4**: 'Row index:' text label above a text input field containing '1' with an information icon.
- 'Number of empty cells allowed:' text label above a text input field containing '1' with an information icon.
- Instructional text: 'Set the maximum number of empty cells before Leo stops counting columns. The last non-empty cell will define the column count.'
- 'Return result in variable:' text label above a dropdown menu with the placeholder text 'Type a variable name'.
- 5**: 'Error handling' section with a dropdown arrow and an information icon.

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

- 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 the wizard 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 the wizard 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.

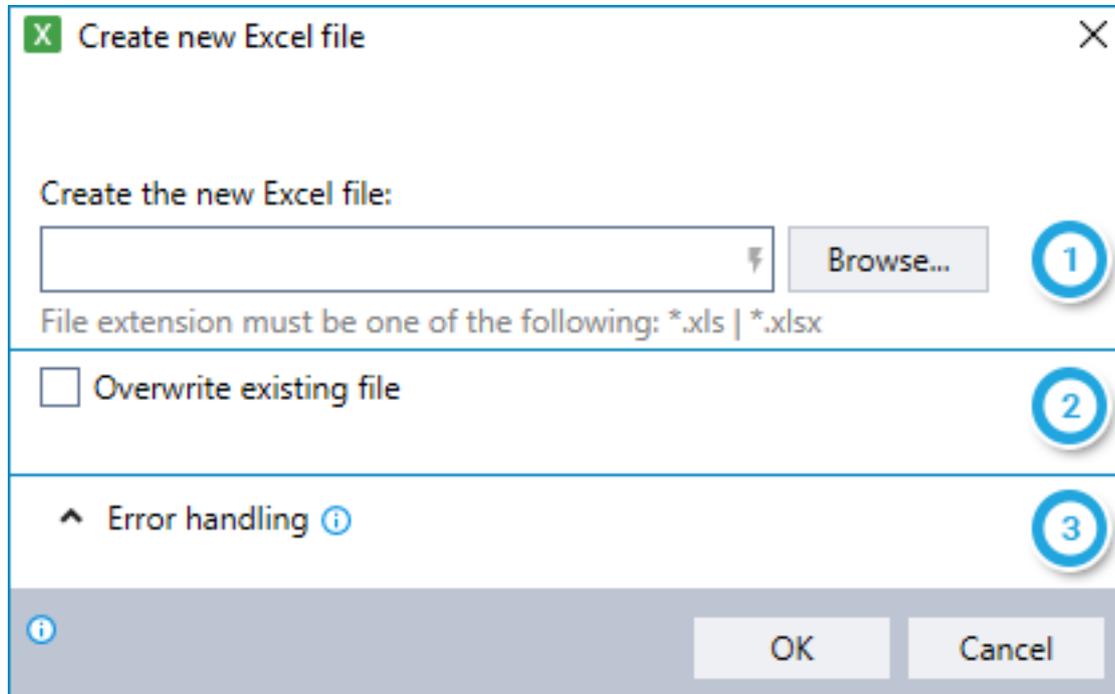
Using the CONVERT EXCEL TO CSV command

- 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 the converted file should be saved, including the full file path
 - If the file does not exist, the wizard will create one with the name you entered
 - If the file does exist, it will be overwritten by the new file
- 4 Instruct the wizard 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 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 the wizard 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

The screenshot shows the 'Query from excel' dialog box with the following sections and callouts:

- 1**: 'Get the values from Excel file:' section with a 'Browse...' button.
- 2**: 'Table data:' section with radio buttons for 'All values in worksheet' and 'A specific range:' (with an example range 'A2:D10'), and a checked checkbox for 'Table contains column headers'.
- 3**: 'Define parameters for your query:' section with a table for parameters and input fields for 'SELECT clause:' and 'WHERE clause:'.
- 4**: 'Return values in variable:' dropdown, 'Column delimiter:', and 'Row delimiter:' fields.
- 5**: 'Error handling' dropdown.

Additional text in the dialog includes:

Referring to table columns in SELECT and WHERE clauses:
Use column header in square brackets. Example: [Quantity]
Or, use the letter F and column index in square brackets. Example: [F2]

Using parameters in WHERE clauses:
Use '@' before parameter name. Example: [Quantity] > @qty

- 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 the wizard 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 screenshot shows the 'Run macro' dialog box with the following elements and numbered callouts:

- 1**: 'Run Macro on file:' text label.
- 1**: Text input field for the file path, with an information icon (i) to its right.
- 1**: 'Browse...' button.
- 2**: Radio button for 'Macro is embedded in file' (selected).
- 2**: Radio button for 'Custom macro' with an 'Edit' link.
- 2**: 'Module name:' text label.
- 2**: Text input field for the module name, with an information icon (i) to its right.
- 2**: 'Procedure name:' text label.
- 2**: Text input field for the procedure name, with an information icon (i) to its right.
- 3**: 'Set the macro function returned result in the variable:' text label.
- 3**: Text input field for the variable name, with a dropdown arrow and an information icon (i) to its right.
- 4**: Checked checkbox for 'Save file when macro ends'.
- 5**: 'Error handling' section with a dropdown arrow and an information icon (i).
- 5**: 'OK' and 'Cancel' buttons at the bottom.

- 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 the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

CHAPTER 19: SAP Commands

In this chapter:

Get SAP Object Text	313
Get SAP Object Value	314
Get SAP Object Location	315
Set SAP Object Value	317



NOTES

Enable SAP GUI scripting first

To use **SAP COMMANDS**, you must first enable GUI scripting on your SAP server and on SAP clients for: (i) your robots; and (ii) the machine(s) on which Kryon Studio is installed. For additional instructions, see *Appendix C: Enabling SAP GUI Scripting* of the Kryon Installation & Upgrade Guide.

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.

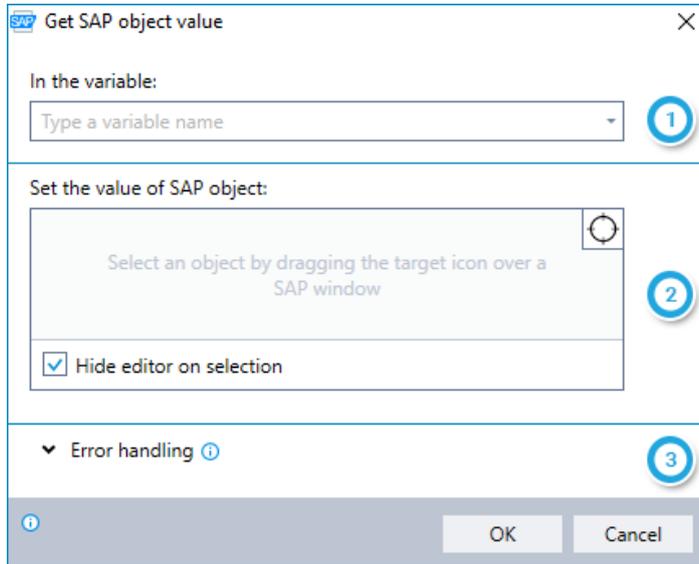
Using the GET SAP OBJECT TEXT command

- 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 Kryon 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 the wizard 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 Kryon 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 the wizard 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 components:

- 1**: A dropdown menu set to 'rectangle' with the label 'Set the ... of selected SAP object'.
- 2**: Four input fields for 'Left', 'Top', 'Width', and 'Height', each with a 'Type a variable name' placeholder.
- 3**: A large text area containing the instruction 'Select an object by dragging the target icon over a SAP window' and a 'Hide editor on selection' checkbox which is checked.
- 4**: An 'Error handling' section with a dropdown arrow and an information icon.

At the bottom of the dialog 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 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 Kryon 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 the wizard 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 Kryon 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 Enter the value you would like to place (can be free text or copied from values stored in variables)
- 3 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 20: UI Automation Commands

In this chapter:

Get UI Object Text	319
Get UI Object Value	320
Get UI Object Location	321
Set UI Object Value	323



NOTE

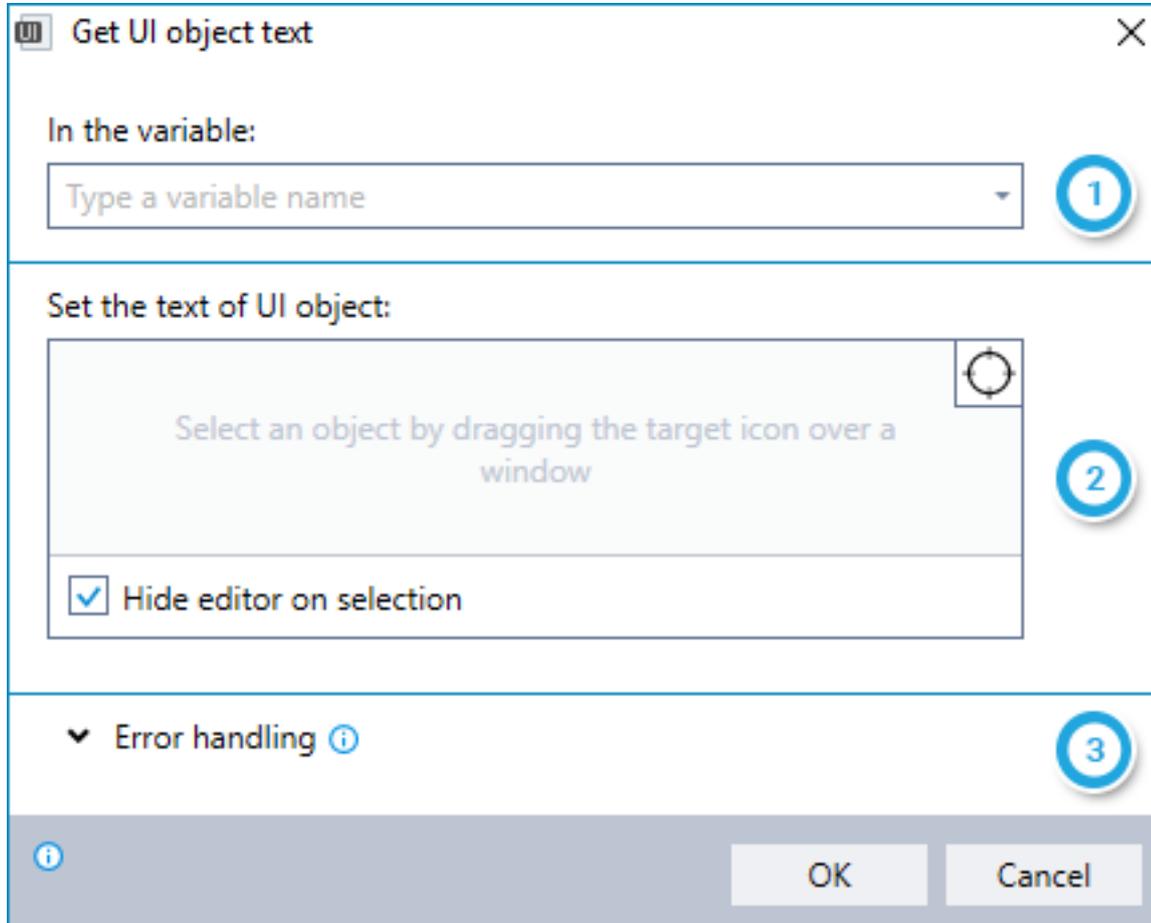
Use in recorded steps only

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

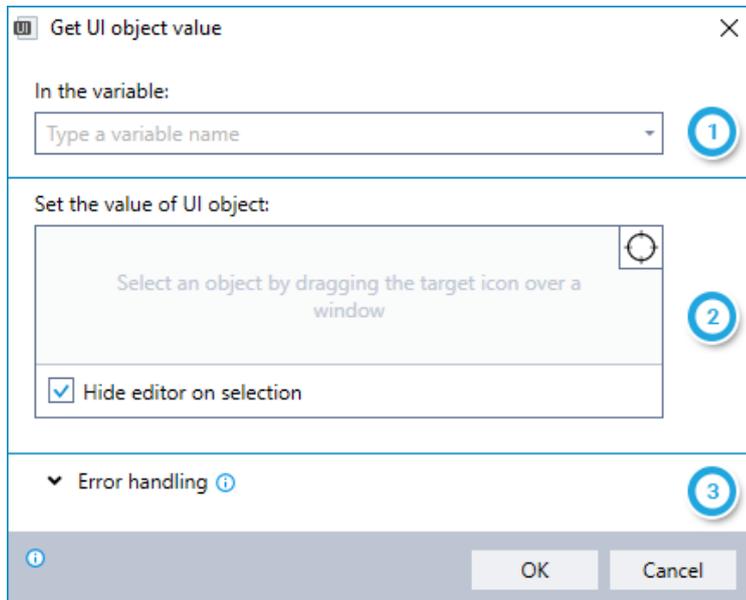


- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in an application window
 - Indicate whether you would like to hide Kryon Studio while you are selecting the object
 - After selecting the object, the following additional options will become available:
 -  Display method used to identify the text of selected object
 -  Display additional information about the selected object
 -  Test retrieving the text of the selected object
- 3 Instruct the wizard 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 Kryon 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 the wizard 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

- 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 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 Kryon 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

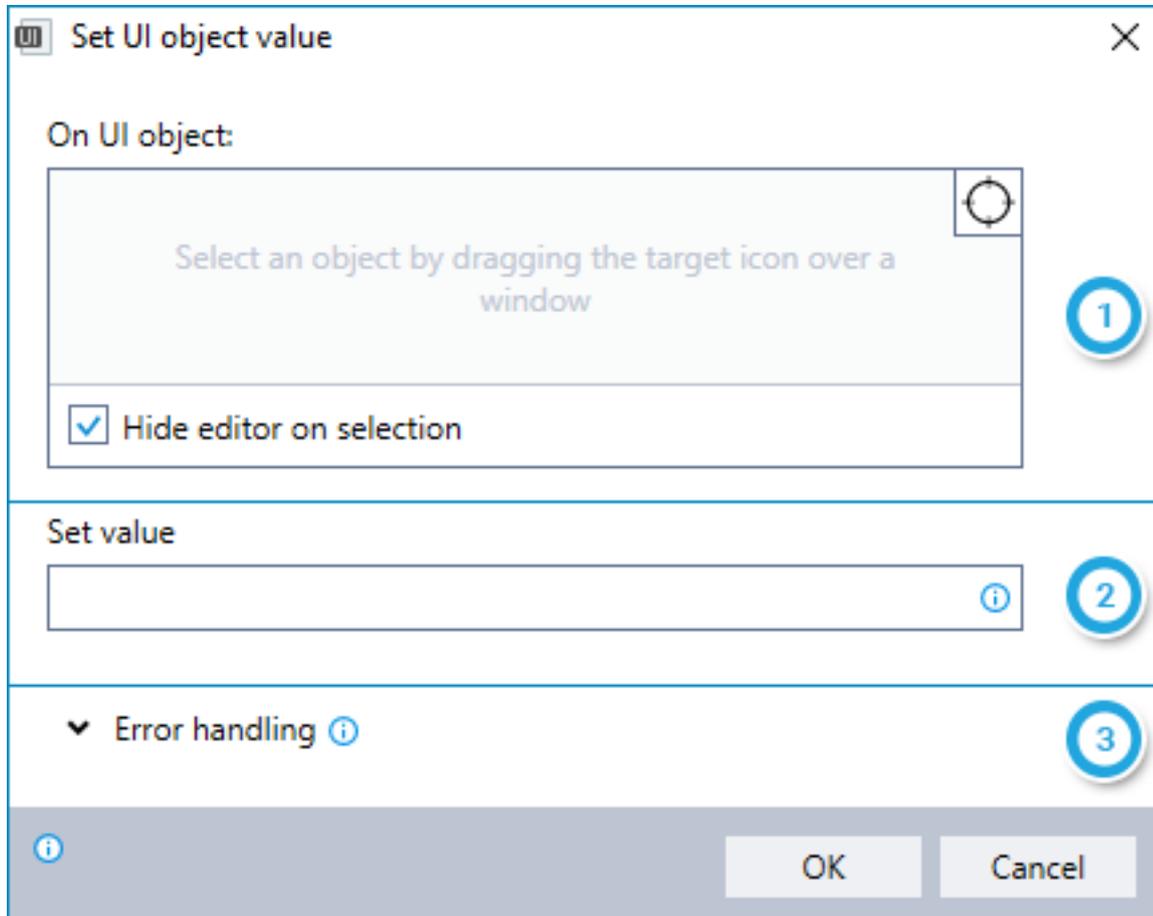


Instruct the wizard how to handle any errors encountered. Read more about **ERROR HANDLING**.

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 Kryon 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** Enter the value you would like to place (can be free text or copied from values stored in variables)
- 3** Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 21: HTML Commands

In this chapter:

HTML Object Selector	326
Get HTML Table	331
Get HTML Object Text	333
Get HTML Object Value	334
Get HTML Object	335
Set HTML Object Value	336
Does HTML Object Exist	337
Click on HTML Object	338
Scroll to HTML Object	339
Extract from HTML Table/List	340



NOTES

Make sure it's the right browser!

HTML COMMANDS support:

- Internet Explorer (IE9 and above); **and**
- Chrome (when the Kryon Connector Chrome extension is installed). To learn more, see [Kryon Connector Chrome Extension](#) .

So make sure:

- You are using either Internet Explorer or Chrome when you select objects as part of the wizard creation/editing process; **and**
- That one of these 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.



NOTE

The HTML object selector makes it easy

All of the HTML commands (with the exception of Extract from HTML table/list) employ Kryon's easy-to-use **HTML OBJECT SELECTOR** to help you identify the HTML object you want to work with. See the [full instructions](#) for all you need to know about using it.

HTML Object Selector

The following HTML commands utilize Kryon's **HTML OBJECT SELECTOR** to make it easy to identify the HTML object you want to work with:

- **GET HTML TABLE**
- **GET HTML OBJECT TEXT**
- **GET HTML OBJECT VALUE**
- **GET HTML OBJECT**
- **SET HTML OBJECT VALUE**
- **DOES HTML OBJECT EXIST**
- **CLICK ON HTML OBJECT**
- **SCROLL TO HTML OBJECT**

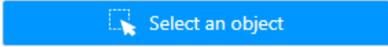
Using the HTML OBJECT SELECTOR

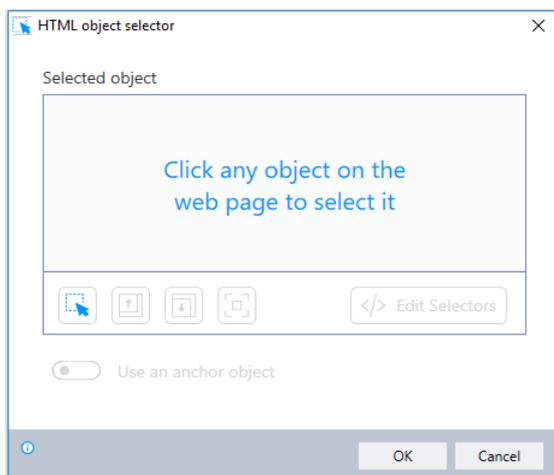


TIP

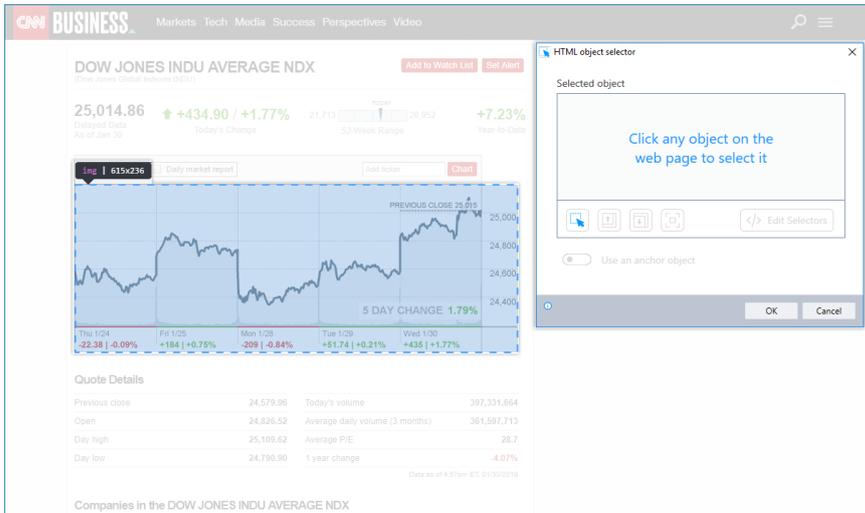
It's easiest to open your web browser and navigate to the desired page prior to using any the HTML commands. Once your browser is open to the page you want to work with, return to Kryon Studio and select the relevant command.

The dialog box for any of the above-listed commands will include a button prompting you to select an object:

1. Click the  button to invoke the **HTML OBJECT SELECTOR**
2. The **HTML OBJECT SELECTOR** will appear with your web browser open behind it



- As you roll over the open page in the web browser, the various HTML objects on the page will be highlighted



TIP

If the **HTML OBJECT SELECTOR** interferes with viewing or selecting the object you need, simply drag the dialog window to a more convenient location on the screen.

- Click on the object you want to select
- The object you selected will appear in the **Selected object** field, and additional options will be enabled, allowing you to work with your selection:

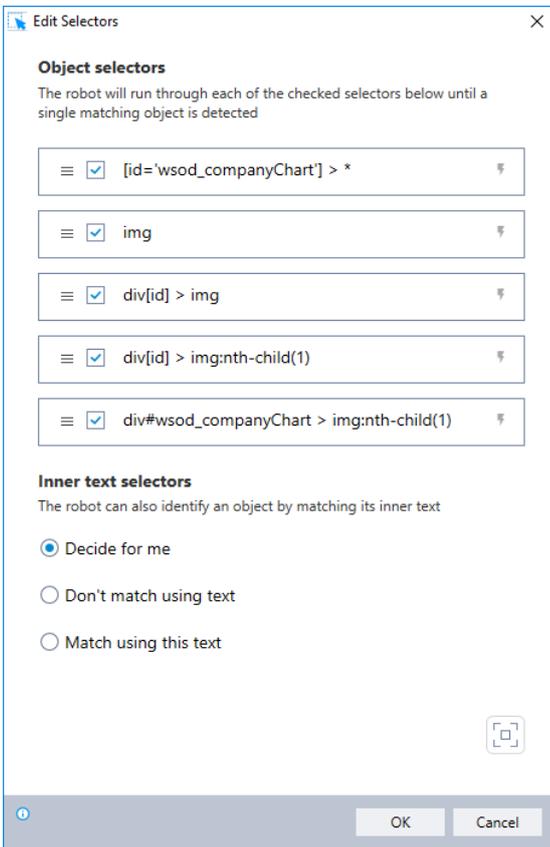
	Expands your selection to the next higher object (the "parent object") in the HTML hierarchy, if one exists
	Narrows your selection to the next lower object (the "child object") in the HTML hierarchy, if it one exists
	Briefly highlights the selected object in the browser
Edit Selectors	Opens a dialog box allowing you to directly edit selectors used to identify the relevant object
<input checked="" type="checkbox"/> Use an anchor object	When toggled On , allows you to define a nearby anchor object that will help to identify the selected object

- When you are satisfied with your selection, click **OK** to return to the main dialog window for the HTML command you are using

Editing selectors

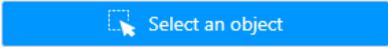
The  button invokes a dialog for directly editing two types of selectors used by the robot to identify the relevant object on the webpage:

- Object selectors
- Inner text selectors



When you are finished editing the selectors, ensure that the correct object is identified by clicking the  button. The selected object will be briefly highlighted in your browser.

Object selectors

The **Object selectors** section allows advanced users to directly edit the HTML object selectors identified during the  process. By default, the robot will run through these selectors in the order they appear until a single matching object is detected.

You can use this dialog to:

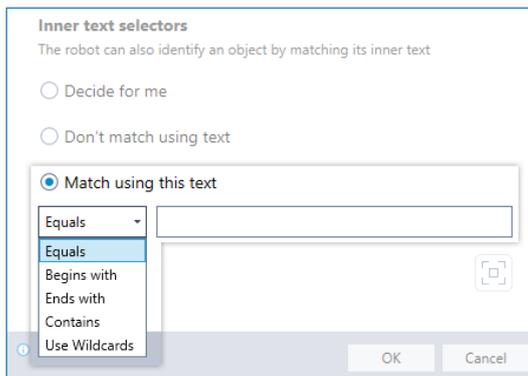
- **Select/deselect selectors:** tick/untick the checkbox of any selector to choose whether it should or should not be used in identifying the relevant object
- **Change the order in which the selectors will be processed:** reorder the selectors by dragging them into the desired order using the ≡ icon at the far left of each selector
- **Modify the selectors altogether:** click in a selector's field to directly edit/overwrite its text (can be free text and/or values copied from variables)

Inner text selectors

The robot can also identify an HTML object by matching the text inside it. By default, Kryon's visual algorithm will determine whether matching inner text will improve accuracy. The **Inner text** section allows you to override the default setting of **Decide for me** by choosing:

- **Don't match using text;** or
- **Match using this text**

When this option is selected, you will be prompted to specify the text and the operator to be used for matching



Using an anchor object



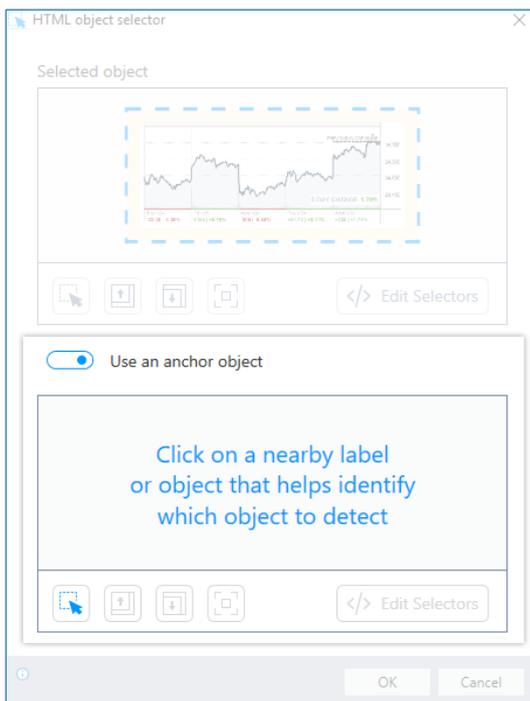
NOTE

What is an anchor object? When should I use one?

An anchor object is either a text label or another unique object that appears on the page somewhere near your "main" selected object (i.e., the object you need the robot to work with).

You should use an anchor object when this main object is identical or similar to another object on the page. The anchor object will be used to help the robot correctly identify the main object when the wizard is run.

When the **Use an anchor object** slider is toggled **On**, a second HTML selection field opens, allowing you to select the anchor object. Use this selection field exactly as you use the main HTML selector.



Get HTML Table

Retrieve the text of an HTML table in the currently active web page and place it into a new or existing variable. 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

1. Use the **HTML OBJECT SELECTOR** to select the table whose text you would like to copy



NOTE

When using this command, the **HTML OBJECT SELECTOR** will only allow the selection of an HTML table. No other type of HTML object can be selected.

2. After selecting the table, the **GET HTML TABLE** dialog will appear as follows:

Get HTML table

Selected object

Previous close	24,579.96
Open	24,826.52
Day high	25,109.62
Day low	24,790.90

</> Edit Selectors

Choose a different object

How to save

Select a variable

Column Delimiter

Row Delimiter

Error handling

OK Cancel

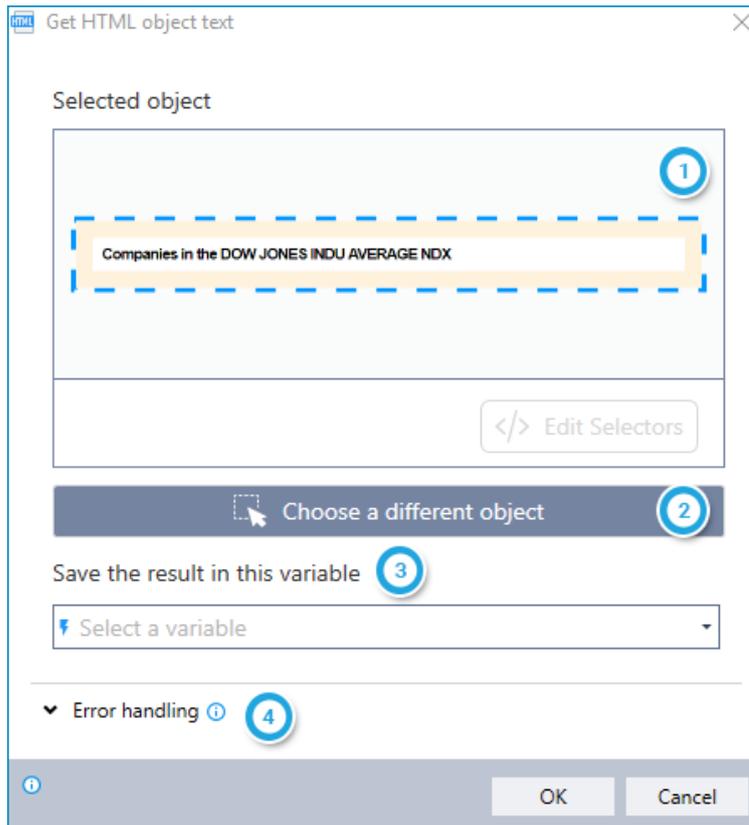
- 1 Preview a thumbnail image of the selected table
- 2 Click to restart the selection process if you need to select a different table
- 3 Enter the name of the variable into which you would like to place the text of the selected table; *and*
the delimiters to use to separate each column and row in the returned data
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Get HTML Object Text

Retrieve the text of an HTML object in the currently active web page and place it into a new or existing variable.

Using the GET HTML OBJECT TEXT command

1. Use the **HTML OBJECT SELECTOR** to select the object whose text you would like to copy
2. After selecting the object, the **GET HTML OBJECT TEXT** dialog will appear as follows:



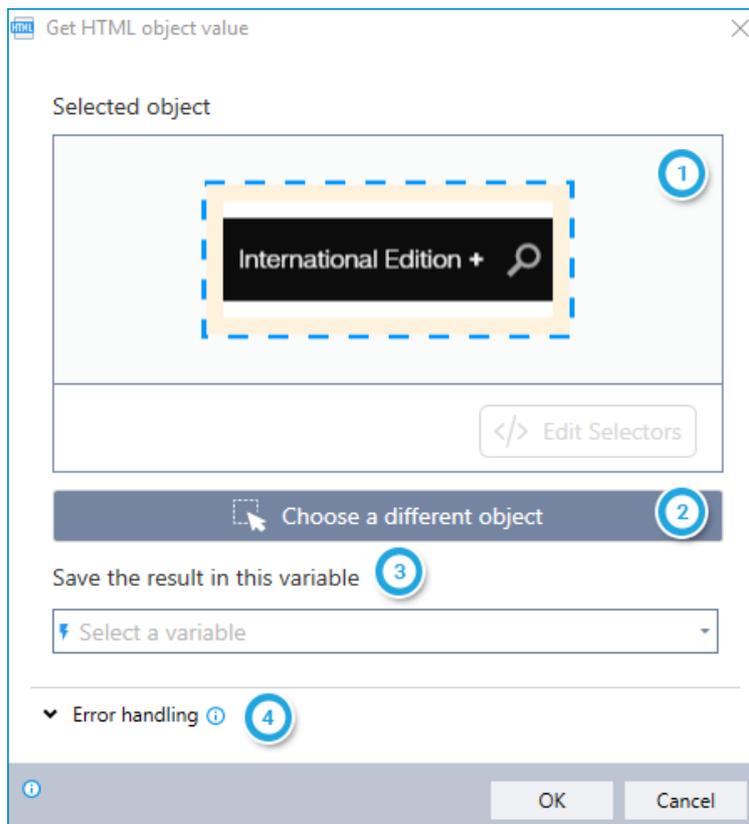
1. Preview a thumbnail image of the selected object
2. Click to restart the selection process if you need to select a different object
3. Enter the name of the variable into which you would like to place the text of the selected object
4. Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Get HTML Object Value

Retrieve the value of an HTML object in the currently active web page 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 HTML OBJECT VALUE command

1. Use the **HTML OBJECT SELECTOR** to select the object whose value you would like to retrieve
2. After selecting the object, the **GET HTML OBJECT VALUE** dialog will appear as follows:



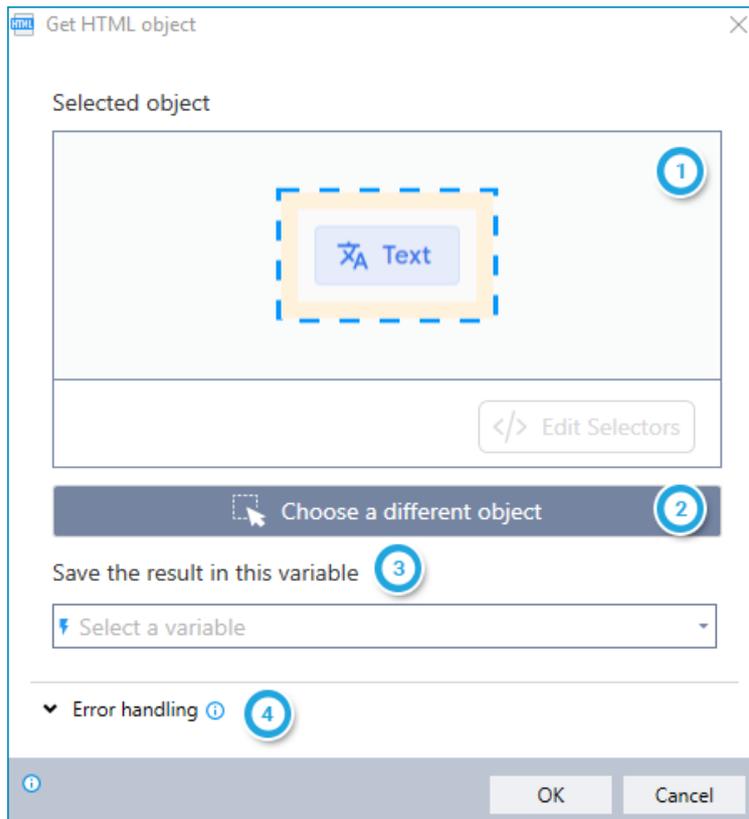
- 1 Preview a thumbnail image of the selected object
- 2 Click to restart the selection process if you need to select a different object
- 3 Enter the name of the variable into which you would like to place the value of the selected object
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Get HTML Object

Retrieve the HTML code for a selected object in the currently active web page and place it into a new or existing variable.

Using the GET HTML OBJECT command

1. Use the **HTML OBJECT SELECTOR** to select the object whose code you would like to retrieve.
2. After selecting the object, the **GET HTML OBJECT** dialog will appear as follows:



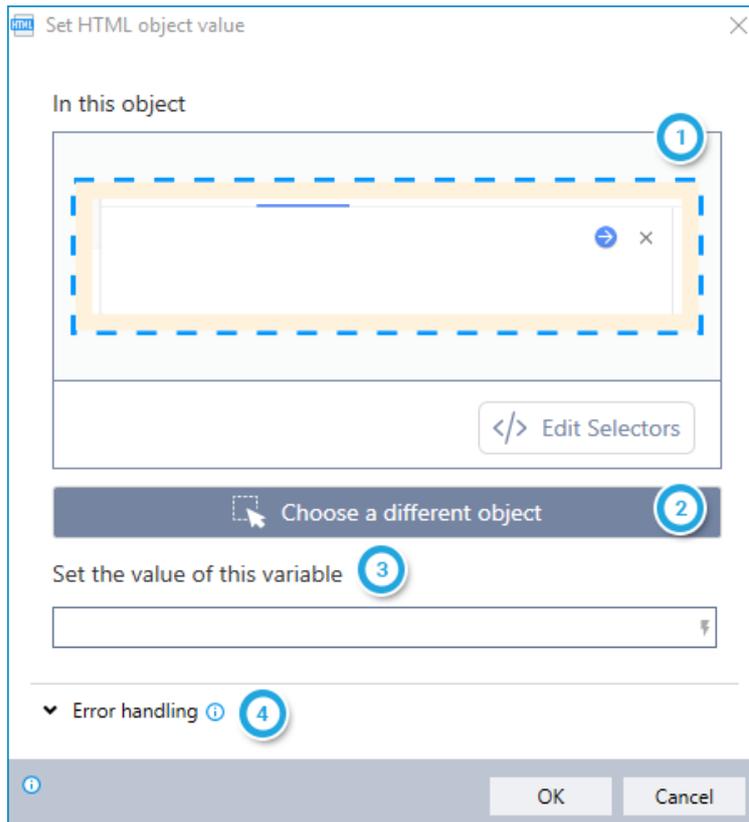
- 1 Preview a thumbnail image of the selected object
- 2 Click to restart the selection process if you need to select a different object
- 3 Enter the name of the variable into which you would like to place the HTML code of the selected object
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Set HTML Object Value

Place a value into an object in the currently active web page.

Using the SET HTML OBJECT VALUE command

1. Use the **HTML OBJECT SELECTOR** to select the object into which you would like to place a value
2. After selecting the object, the **SET HTML OBJECT VALUE** dialog will appear as follows:



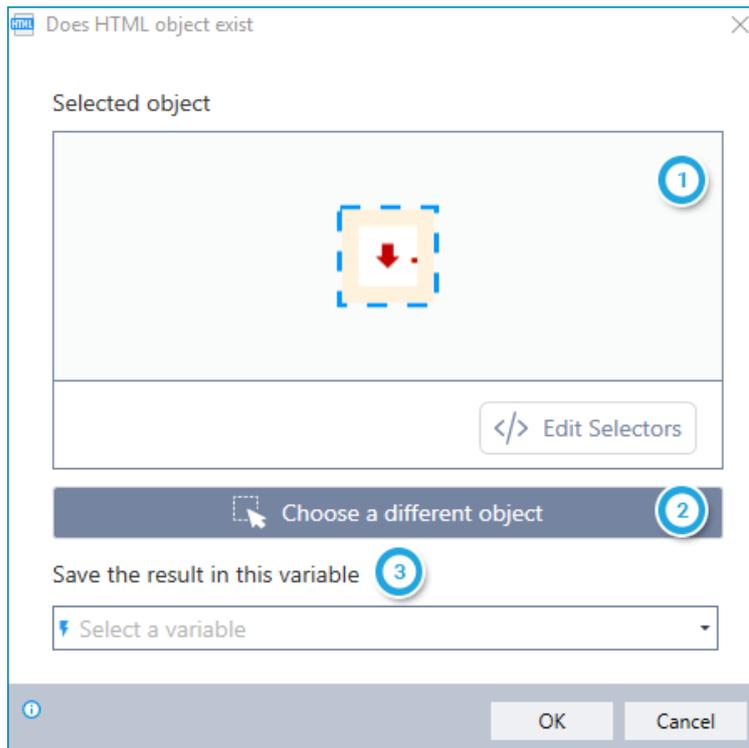
- 1 Preview a thumbnail image of the selected object
- 2 Click to restart the selection process if you need to select a different object
- 3 Enter the value you would like to place (can be free text or copied from values stored in variables)
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Does HTML Object Exist

Check to see if an object in the currently active web page exists and place the result of the check (TRUE/FALSE) into a variable.

Using the DOES HTML OBJECT EXIST command

1. Use the **HTML OBJECT SELECTOR** to select the object whose existence you would like to check
2. After selecting the object, the **DOES HTML OBJECT EXIST** dialog will appear as follows:



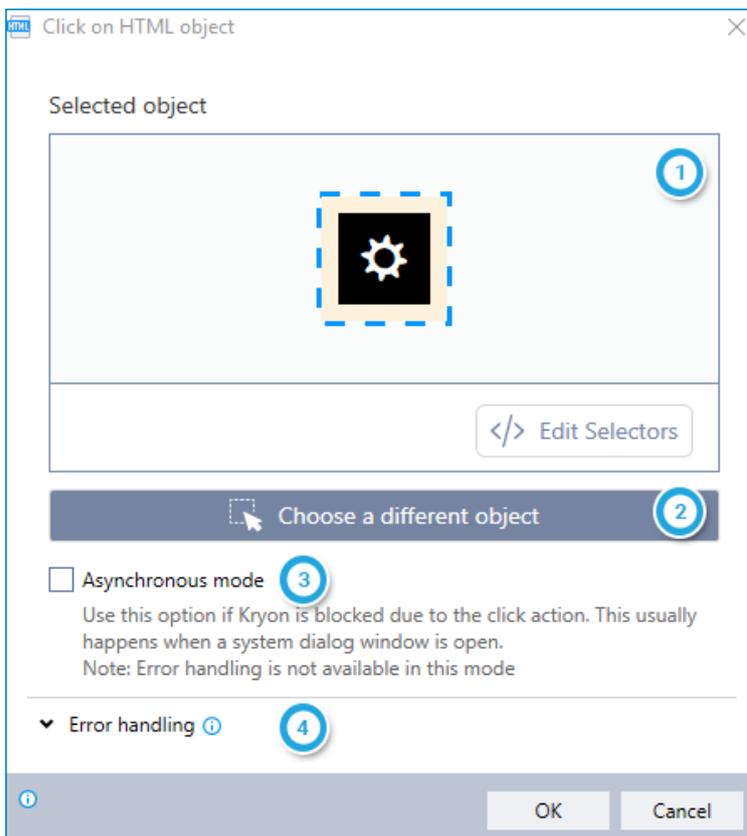
1. Preview a thumbnail image of the selected object
2. Click to restart the selection process if you need to select a different object
3. Enter the name of the variable into which you'd like to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)

Click on HTML Object

Click the mouse on an object in the currently active web page.

Using the **CLICK ON HTML OBJECT** command

1. Use the **HTML OBJECT SELECTOR** to select the object on which you would like to click
2. After selecting the object, the **CLICK ON HTML OBJECT** dialog will appear as follows:



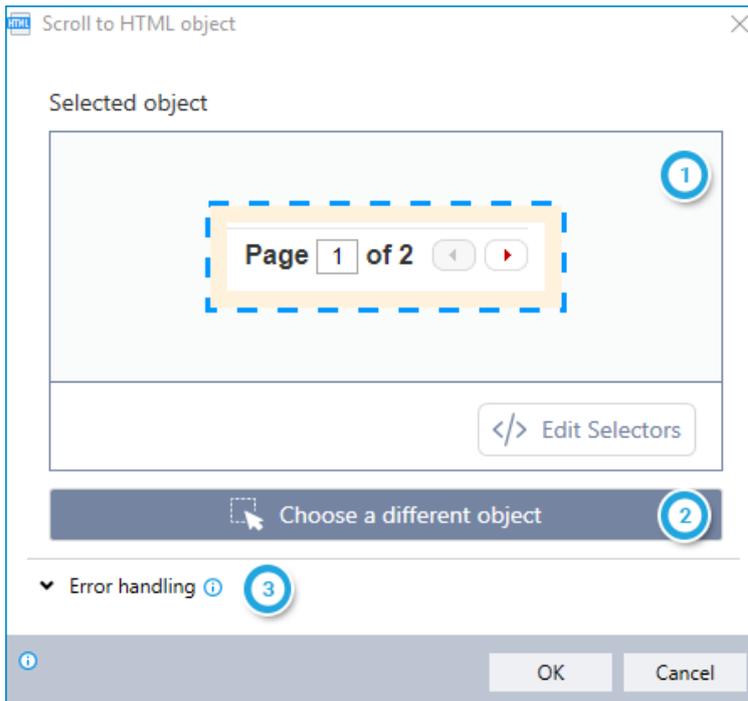
- 1 Preview a thumbnail image of the selected object
- 2 Click to restart the selection process if you need to select a different object
- 3 Indicate whether you would like to use Asynchronous mode
- 4 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Scroll to HTML Object

Scroll to an object in the currently active web page so that it is visible within the window.

Using the SCROLL TO HTML OBJECT command

1. Use the **HTML OBJECT SELECTOR** to select the object to which you would like to scroll
2. After selecting the object, the **SCROLL TO HTML OBJECT** dialog will appear as follows:



1. Preview a thumbnail image of the selected object
2. Click to restart the selection process if you need to select a different object
3. Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Extract from HTML Table/List

Extract all or selected data from a web-based table or list and place it into a new or existing variable.



NOTE

A terminology shortcut

Throughout this topic, the term "table" is used as a shortcut to refer to both tables and lists. However, as the name of this command suggests, it designed to extract data from both, and the instructions in this topic refer equally to both.

Using the **EXTRACT FROM HTML TABLE/LIST** command



TIP

It's easiest to open your web browser and navigate to the desired page prior to using this command. Once your browser is open to the page you want to work with, return to Kryon Studio and select the **EXTRACT FROM HTML TABLE/LIST** command to get started.

Selecting the data to extract

This command employs a "specialized" version of the [HTML object selector](#) that is designed to recognize patterns.

Most web pages are comprised of a series of HTML building blocks (or objects). Because a table presents data in a structured format, the HTML objects within a table generally appear in a regular, repeating pattern. By using the **EXTRACT FROM HTML TABLE/LIST** selector, you actually "teach" the robot to recognize this pattern so that it can extract from the table exactly the data you are looking for.



NOTE

A little bit more terminology

Throughout this command, the following terms are used –

- **Item:** refers to each "row" in a table or list
- **Element:** refers to each piece of data presented about each item (in table terminology, this would be called a "column")

EXAMPLE

On the way to the store...

Assume you are a **VERY** organized grocery shopper, and you have a list of all the products you need to buy. For each product, you have details about: the brand, the package size, and the price. (Yes, you are **VERY, VERY** organized!)

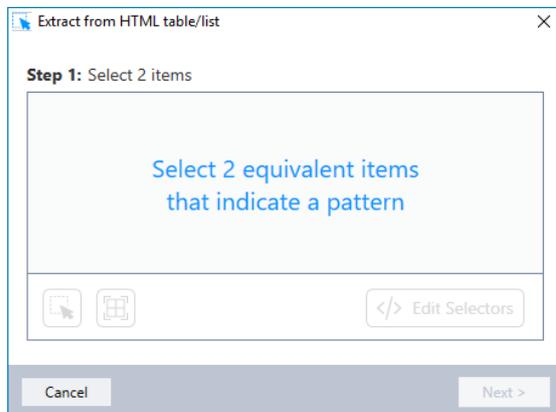
In this case:

- Each product on the list (such as milk, eggs, cola, pasta, and tomatoes) would be an **item**; and
- For each item, each piece of detailed information (brand, size, and price) is an **element**

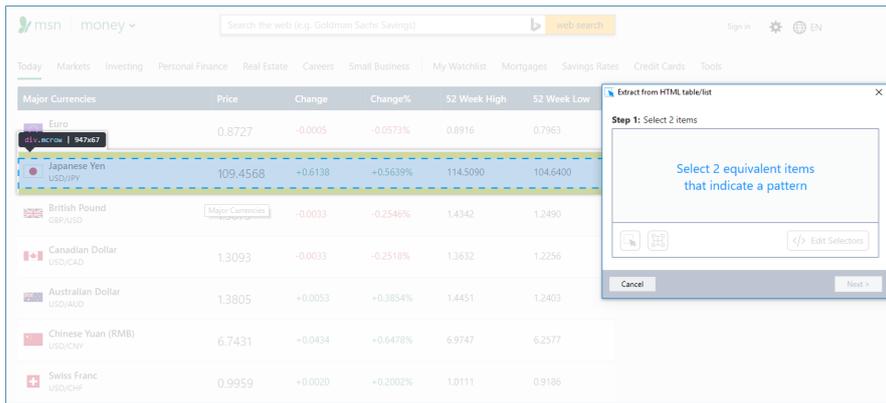
Selecting items

When you first open the command, you will be prompted to select items. By selecting just 2 equivalent items (at the same level of the HTML hierarchy), you will teach the robot the pattern for identifying items.

1. Click the  button to invoke the selector
2. The selector will appear with your web browser open behind it



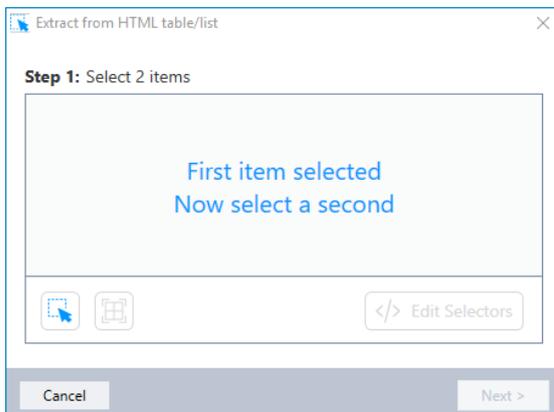
3. As you roll over the open page in the web browser, the various HTML objects on the page will be highlighted



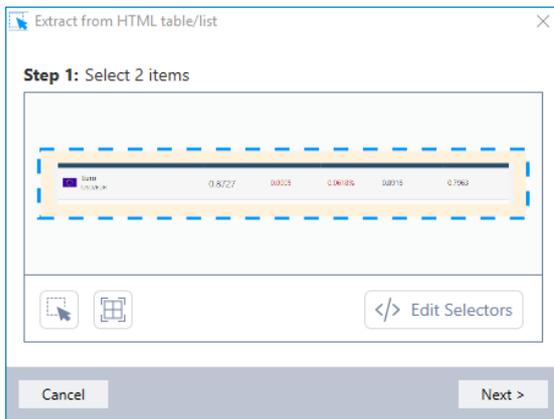
TIP

If the selector window interferes with viewing or selecting the object you need, simply drag the dialog window to a more convenient location on the screen.

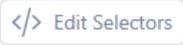
4. Click on the object that represents an item in the table you want to work with
5. You will be prompted to select a second (equivalent) item



6. After you select the second item, an item representing the selected pattern will appear in the **Select 2 items** field



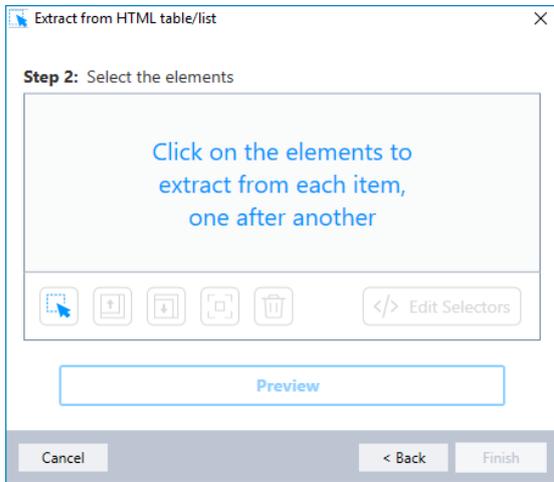
The following additional options will be enabled, allowing you to work the your item selection:

	Briefly highlights the pattern of selected items in the browser
	Opens a dialog box allowing you to directly edit selectors used to identify the relevant object

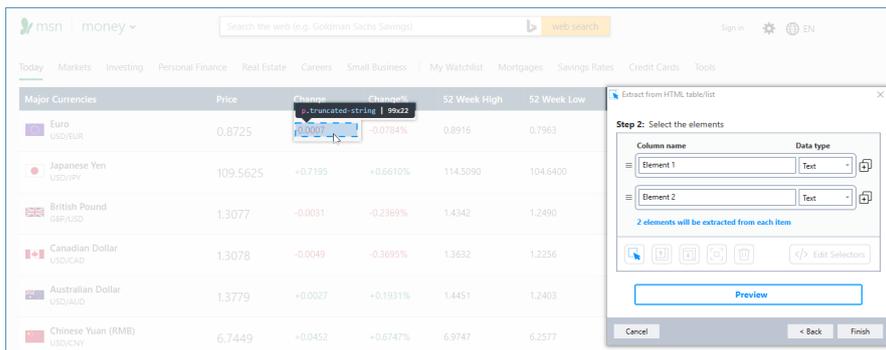
7. When you are satisfied with your selection, click **Next** to move on to selecting elements

Selecting elements

- Now that you've shown the robot what items to select, the selector will prompt you to select the elements you want to extract from each item.



- Click, one by one, on each element that you want to extract from within the item
As you click on each element, it will be added to the selector window:



- As you add elements, you can work with them within the selector window:

	Expands selected element to the next higher object (the "parent object") in the HTML hierarchy, if one exists
	Narrows selected element to the next lower object (the "child object") in the HTML hierarchy, if it one exists
	Briefly highlights the selected element in the browser
	Deletes the selected element from the list

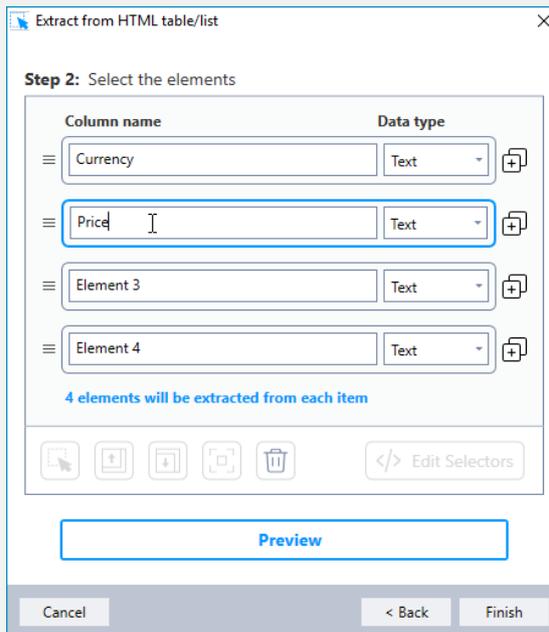
 Edit Selectors	Opens a dialog box allowing you to directly edit selectors used to identify the element
--	--



TIPS

Recognize your elements

Highly recommended: Click within the **Column name** field of each element and change the default name (i.e., Element #) to a name that will be easy to recognize within the extracted data.



Keep your elements in order

By default, elements will appear in the list in the order you selected them. You can change the order in which they appear either by:

- using the  icon at the far left of each element to drag them into the desired order; *or*
- clicking the  button at the far right of each element to move it up one position

- Click on the Preview button at any time to see a sample of the data that will be extracted based on your selections:

The data below is a sample of your selection

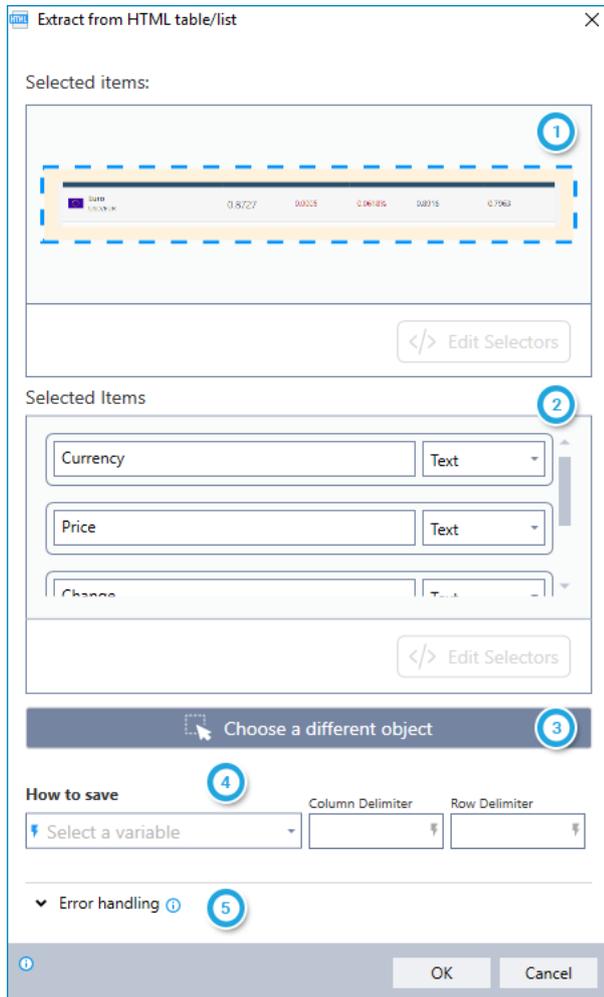
	Currency	Price	Change	Change%
1	Euro	0.8726	-0.0006	-0.0699%
2	Japanese Yen	109.3845		
3	British Pound	1.3097	-0.0011	-0.0820%
4	Canadian Dollar	1.3094	-0.0033	-0.2495%
5	Australian Dollar	1.3778		
6	Chinese Yuan (RMB)	6.7449		
7	Swiss Franc	0.9952		
8	Hong Kong Dollar	7.8471		
9	Swedish Krona	9.0413	-0.0014	-0.0155%
10	South Korean Won	1,118.7000		
11	Russian Rouble	65.5260		
12	South African Rand	13.3277		
13	Indian Rupee	71.4450		
14	Brazilian Real	3.6400		

OK

- When you are satisfied with your selections, click **Finish** to return to the main **EXTRACT FROM HTML TABLE/LIST** dialog

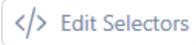
Finalize command options

After item/element selections have been made, the **EXTRACT FROM HTML TABLE/LIST** dialog will appear as follows:

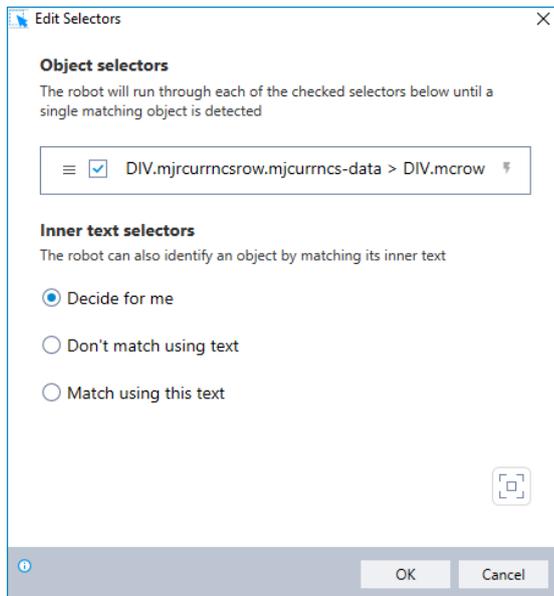


- 1 Preview a thumbnail image of a typical selected item
- 2 Review the list of elements that will be extracted from each item
- 3 Click to restart the selection process (if necessary)
- 4 Enter the name of the variable into which you would like to place the extracted data; **and** the delimiters to use to separate each column and row in the returned data
- 5 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

Editing selectors (optional)

After you have selected items and/or elements, the  gives you the option to directly edit two types of selectors used by the robot to identify the relevant objects:

- **Object selectors**
- **Inner text selectors**(for item selections only, not applicable to element selections) (for item selections only)



When you are finished editing the selectors, ensure that the correct object is identified by clicking the  button. The selected object will be briefly highlighted in your browser.

Object selectors

The **Object selectors** section allows advanced users to directly edit the HTML object selectors identified during the **select items/elements** process. By default, the robot will run through these selectors in the order they appear until a single matching object is detected.

You can use this dialog to:

- **Select/deselect selectors:** tick/untick the checkbox of any selector to choose whether it should or should not be used in identifying the relevant object
- **Change the order in which the selectors will be processed:** reorder the selectors by dragging them into the desired order using the  icon at the far left of each selector
- **Modify the selectors altogether:** click in a selector's field to directly edit/overwrite its text (can be free text and/or values copied from variables)

Inner text selectors

(for item selections only, not applicable to element selections)

The robot can also identify an HTML object by matching the text inside it. By default, Kryon's visual algorithm will determine whether matching inner text will improve accuracy. The **Inner text** section allows you to override the default setting of **Decide for me** by choosing:

- **Don't match using text;** or
- **Match using this text**

When this option is selected, you will be prompted to specify the text and the operator to be used for matching

Inner text selectors
The robot can also identify an object by matching its inner text

Decide for me

Don't match using text

Match using this text

Equals
Equals
Begins with
Ends with
Contains
Use Wildcards

OK Cancel

CHAPTER 22: .NET Automation Commands

In this chapter:

Get .NET Object Text	351
Get .NET Object Value	352
Get .NET Object Location	353
Set .NET Object Value	355



NOTE

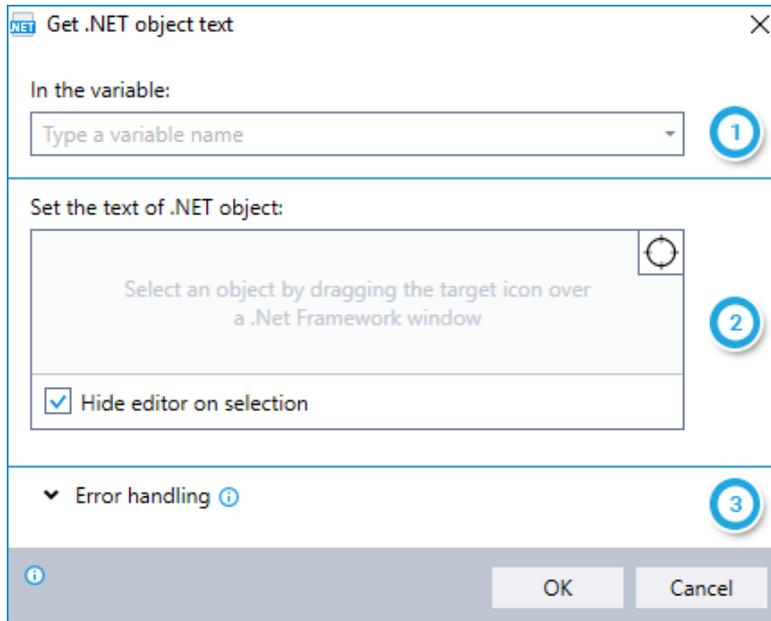
Use in recorded steps only

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 Kryon 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 the wizard 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 Kryon 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
- 4 Instruct the wizard 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

- 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 to place the location information



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 Kryon 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



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

CHAPTER 23: Java Automation Commands

In this chapter:

Get Java Object Text	358
Get Java Object Value	359
Get Java Object Location	360
Set Java Object Value	362



NOTES

Install the Kryon Java bridge first

Use of **JAVA AUTOMATION COMMANDS** requires installation of the Kryon Java bridge on: (i) your robots; and (ii) the machine(s) on which Kryon Studio is installed. For additional information, contact your Kryon 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)

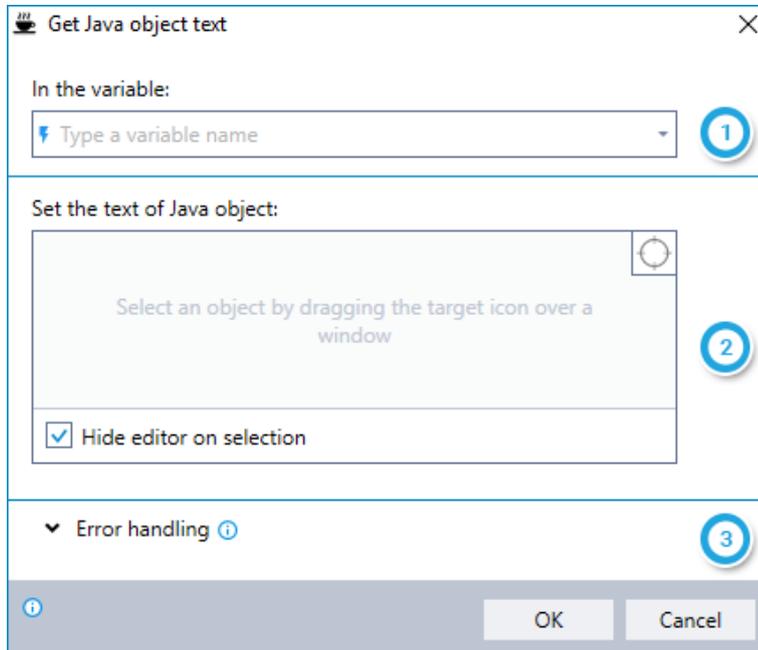
Supported controls:

- 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

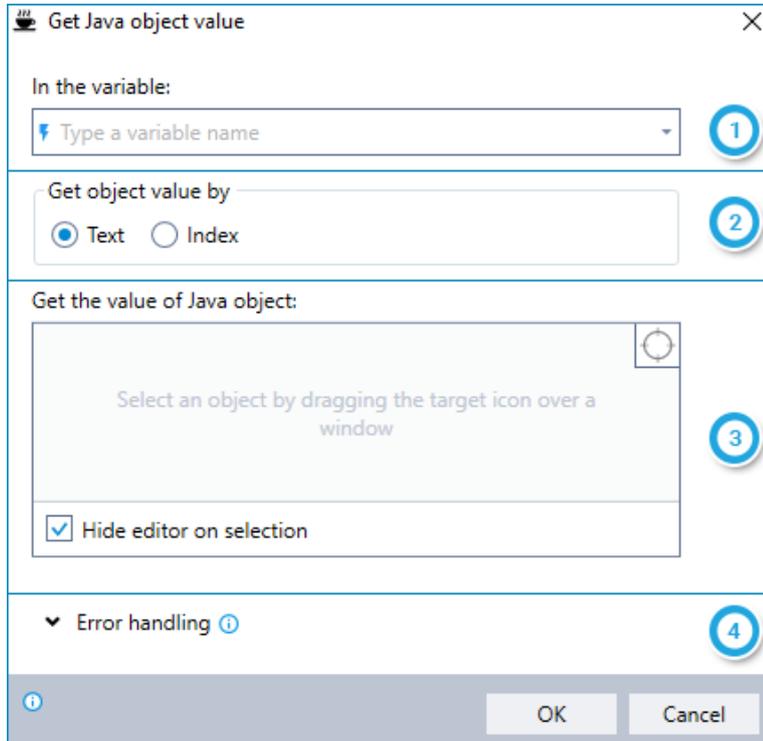


- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in a Java window
 - Indicate whether you would like to hide Kryon Studio while you are selecting the object
 - 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 the wizard 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 Kryon 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 the wizard 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 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

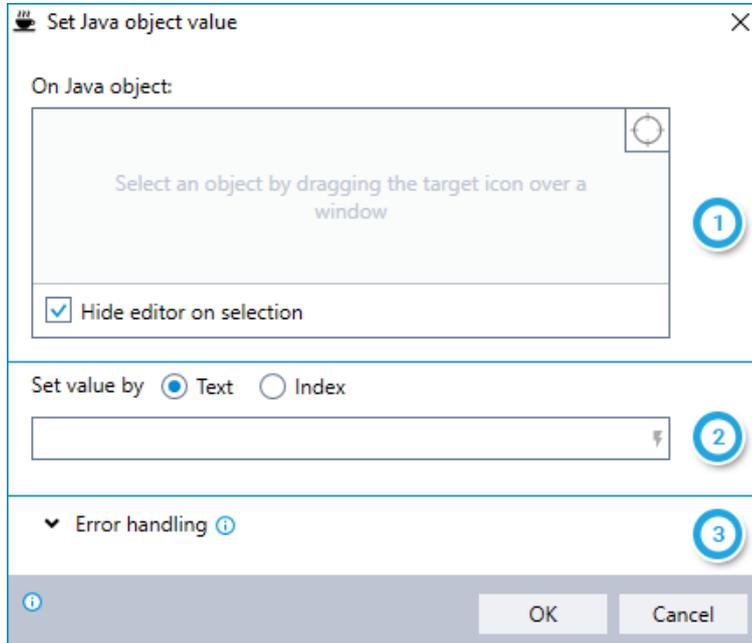
- Indicate whether you would like to hide Kryon Studio while you are selecting the object
- After selecting the object, the following additional options will become available:
 -  Configure available additional settings
 - Info** Display additional information about the selected object
 - Test** Test retrieving the location of the selected object

4 Instruct the wizard 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 Kryon 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 the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

CHAPTER 24: Global Variable Commands

In this chapter:

Set Global Variable	364
Get Global Variable	365
Delete Global Variable	366

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  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". It has a close button 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 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

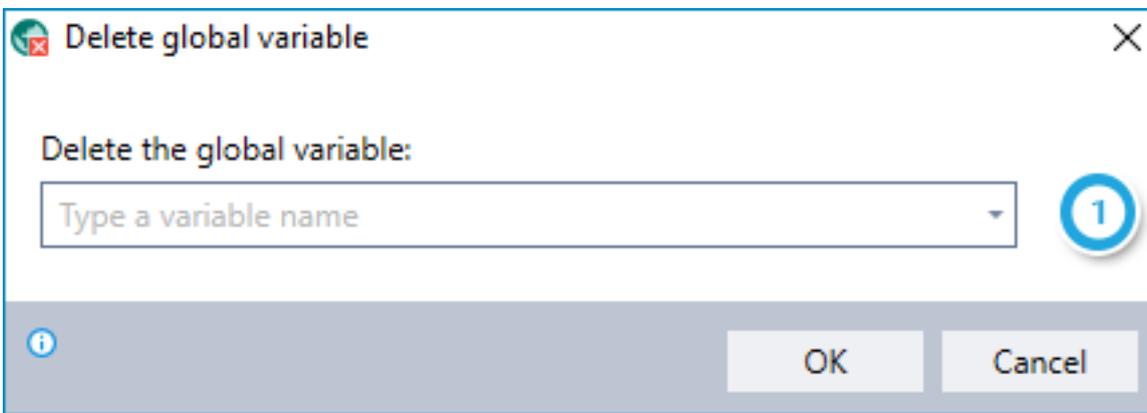
Delete a global variable (previously created in this or a different wizard using the **SET GLOBAL VARIABLE** command) so that it is no longer available for use.



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	368
Notes for Mobile/Web Access	369
View Variable List	370
Show Debug Message	372

Note

Enter an internal note to appear in the Editor Pane of the Advanced Commands view.



TIP

Just do it.

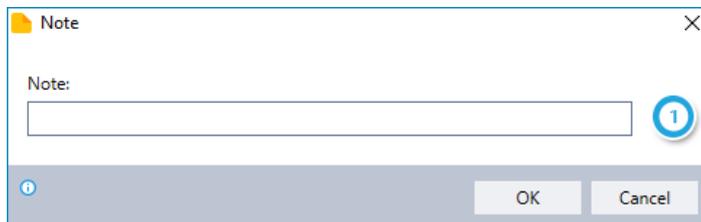
Anyone who has ever developed code knows how important it is to document it internally. But all too often, in the rush of getting things done, this crucial practice is overlooked. Don't let it happen to you!

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



Enter the text of the note as you want it to appear

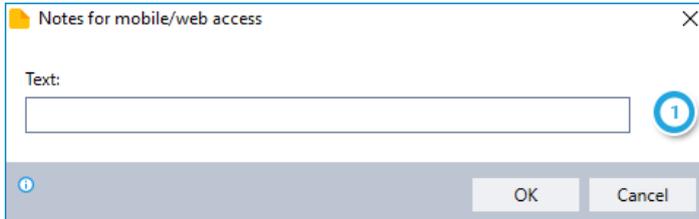
- The note will appear only in the Advanced Commands Editor, so end users will never see it when the wizard/sensor is run

Notes for Mobile/Web Access

Enter an internal note to appear on the mobile/webpage summary of the wizard.

- Applicable only if Kryon Mobile/Web Access has been deployed and configured for your organization

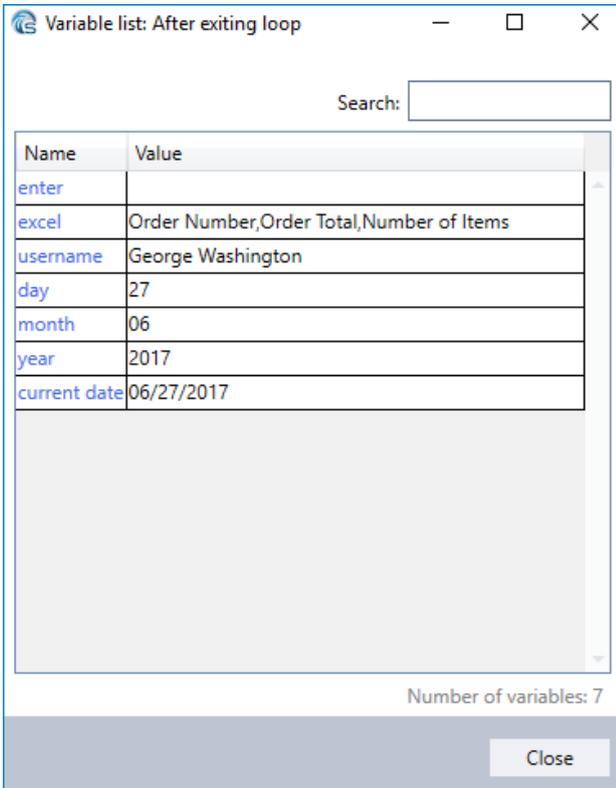
Using the NOTES FOR MOBILE/WEB ACCESS command



- 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:

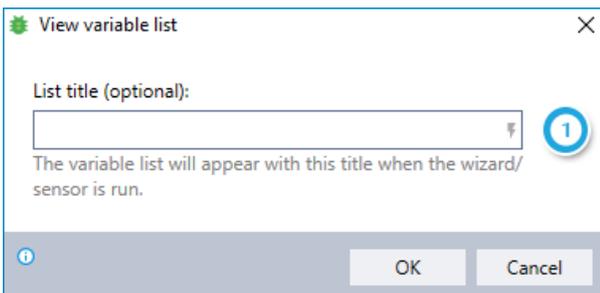


CAUTION

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





(Optional) Enter a list title that will appear when the wizard is run

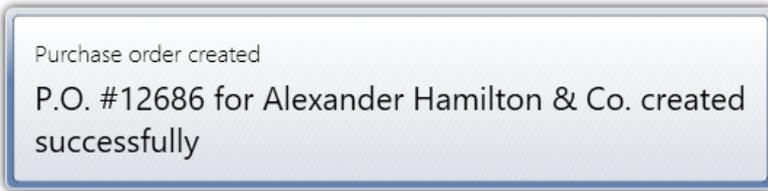


TIP

Give your list a title that will help you identify the point at which it was created.

Show Debug Message

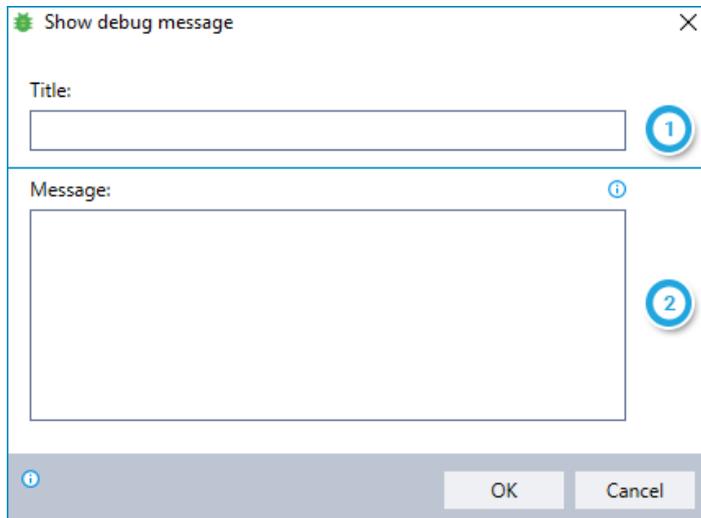
Display a text message to the Kryon Studio user when the wizard is run in debug mode. Here's a sample:



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 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 the wizard should report errors in any Advanced Command in which you see ▼ Error handling ⓘ options:

^ Error handling ⓘ

Error variable: 1

General error value:

Value not found: 2

Key not found:

- 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**

APPENDIX B: Kryon Connector Chrome Extension

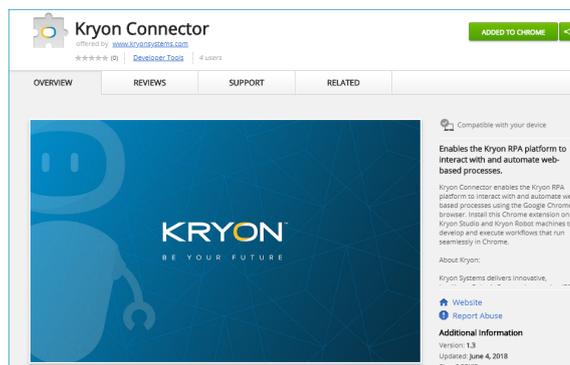
Install the [Kryon Connector](#) Chrome extension to use Chrome when developing wizards with these Advanced Commands:

GET WEB PAGE HTML

RUN JAVASCRIPT ON PAGE

HTML commands:

- **GET HTML TABLE**
- **GET HTML OBJECT TEXT**
- **GET HTML OBJECT VALUE**
- **GET HTML OBJECT**
- **SET HTML OBJECT VALUE**
- **DOES HTML OBJECT EXIST**
- **CLICK ON HTML OBJECT**
- **SCROLL TO HTML OBJECT**
- **EXTRACT FROM HTML TABLE/LIST**



NOTES

Don't forget the robots!

Kryon Connector should also be installed on unattended and attended robots to enable them to run wizards containing these Advanced Commands in Chrome.

Window detection with HTML objects

In addition to Advanced Commands, Kryon Connector also enables window detection with HTML objects when using Chrome. For more information about advanced window detection techniques, see the *Window Detection* section of the Kryon Studio User Guide.



TIP

The Kryon Connector extension will be automatically installed when you install or upgrade your Kryon Robot and Studio clients to version 5.21 or later. Be sure to enable it when prompted in Chrome:

"Kryon Connector" added

Another program on your computer added an extension that may change the way Chrome works.

It can:

- Read and change all your data on the websites you visit
- Communicate with cooperating native applications

Enable extensionRemove from Chrome

Kryon Connector is also available as a free download from the [Chrome Store](#).