



Kryon Studio

VERSION 20.9

Advanced Commands

Reference Guide

KRYON™

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: 21-Sep.-2020

Contents

CHAPTER 1: Variable Commands

Set Value	18
Using the SET VALUE command	18
Find	20
Using the FIND command	20
Replace	28
Using the REPLACE command	28
Mathematics	30
Using the MATHEMATICS command	30
Evaluate Expression	31
Using the EVALUATE EXPRESSION command	31
Split	36
Using the SPLIT command	36
Get Array Data	40
Using the GET ARRAY DATA command	41
Get Table Data	44
Using the GET TABLE DATA command	44
Table Lookup	49
Using the TABLE LOOKUP command	49
Get ASCII Character	54
Using the GET ASCII CHARACTER command	54
Remove Blank Spaces	56
Using the REMOVE BLANK SPACES command	56
Change Case	57
Using the CHANGE CASE command	57
Check Type	58
Using the CHECK TYPE command	58
Get Random Number	59
Using the GET RANDOM NUMBER command	59
Reverse	60
Using the REVERSE command	60

Get Length	61
Using the GET LENGTH command	61
Extract Numeric Values	62
Using the EXTRACT NUMERIC VALUES command	62

CHAPTER 2: Flow Commands

If Else	66
Using the IF ELSE command	66
Complex If Else	70
Using the COMPLEX IF ELSE command	70
Multi-Value If Else	72
Using the MULTI-VALUE IF ELSE command	72
Loop	74
Using the LOOP command	74
Loop: Break	76
Loop: Restart	77
Loop Table	78
Using the LOOP TABLE command	78
Loop Items	80
Using the LOOP ITEMS command	80
Pause	82
Using the PAUSE command	82
Group	83
Using the GROUP command	83
SET GROUP AS GLOBAL option	84
Global Functions	86
Creating Global Function	86
Adding the global function to wizards	88

CHAPTER 3: Wizard Commands

Continue Wizard	94
How does it work?	94
End Wizard	95
Using the END WIZARD command	95

Go To Step	96
Using the GO TO STEP command	96
Get Step Data	98
Using the GET STEP DATA command	98
Get Wizard Data	99
Using the GET WIZARD DATA command	99
Check Application	100
Using the CHECK APPLICATION command	100
Get User Data	101
Using the GET USER DATA command	101
Check Run Mode	102
Using the CHECK RUN MODE command	102
Check Video Recording Mode	103
Using the CHECK VIDEO RECORDING MODE command	103
Set User Interrupt Mode	104
Using the SET USER INTERRUPT MODE command	104
Show Message	105
Using the SHOW MESSAGE command	105
Raise Wizard Error	107
Using the RAISE WIZARD ERROR command	107
Get Last Failure Type	108
Using the GET LAST FAILURE TYPE command	108
Resume Error	109
Report Wizard Output	110
Using the REPORT WIZARD OUTPUT command	110

CHAPTER 4: Mouse and Keyboard Commands

Use Keyboard Shortcut	112
Using the USE KEYBOARD SHORTCUT command	112
Input Text	113
Using the INPUT TEXT command	113
Get Mouse Position	114
Using the GET MOUSE POSITION command	114

Drag & Drop	115
Using the DRAG & DROP command	115
Mouse Click	117
Using the MOUSE CLICK command	117
Wait for Busy Cursor	118
Set Caps Lock State	119
Using the SET CAPS LOCK STATE command	119
Get Caps Lock State	120
Using the GET CAPS LOCK STATE command	120

CHAPTER 5: Block Commands

Allow Last Stored Action	121
Deny Last Stored Action	121
Remove All Blocks	122

CHAPTER 6: Date and Time Commands

Get Current Date	125
Using the GET CURRENT DATE command	125
Validate Date	126
Using the VALIDATE DATE command	126
Get Day of Month	127
Using the GET DAY OF MONTH command	127
Compare Dates	129
Using the COMPARE DATES command	129
Add/Subtract Date	131
Using the ADD/SUBTRACT DATE command	131
Calculate Date Range	133
Using the CALCULATE DATE RANGE command	133
Check Day of Week	135
Using the CHECK DAY OF WEEK command	135
Format Date	137
Using the FORMAT DATE command	137
Specifying custom output formats	139
Get Current Time	140

Using the GET CURRENT TIME command	140
Compare Time	141
Using the COMPARE TIME command	141
Add/Subtract Time	143
Using the ADD/SUBTRACT TIME command	143
Calculate Time Range	144
Using the CALCULATE TIME RANGE command	144
Convert Between Time Zones	146

CHAPTER 7: Window Commands

Get Step Window Handle	148
Using the GET STEP WINDOW HANDLE command	148
Find Matching Window Handles	149
Using the FIND MATCHING WINDOW HANDLES command	149
Get Active Application	151
Using the GET ACTIVE APPLICATION command	151
Control Window State	152
Using the CONTROL WINDOW STATE command	152
Check Window State	154
Using the CHECK WINDOW STATE command	154
Get Active Window/Web Page	155
Using the Get ACTIVE WINDOW/WEB PAGE command	155
Capture Step Window Image	156
Using the CAPTURE STEP WINDOW IMAGE command	156

CHAPTER 8: File Commands

Create a Text File	158
Using the CREATE A TEXT FILE command	158
Read From Text File	160
Using the READ FROM TEXT FILE command	160
Write to Text File	161
Using the WRITE TO TEXT FILE command	161
Does File Exist	162
Using the DOES FILE EXIST command	162

Copy a File	163
Using the COPY A FILE command	163
Move a File	164
Using the MOVE A FILE command	164
Rename a File	165
Using the RENAME A FILE command	165
Delete a File	166
Using the DELETE A FILE command	166
Delete File(s)	168
Using the DELETE FILE(S) command	168
Monitor File Changes	170
Using the MONITOR FILE CHANGES command	170

CHAPTER 9: Folder Commands

Create a Folder	173
Using the CREATE A FOLDER command	173
Get Folder Location	174
Using the GET FOLDER LOCATION command	174
Get Files	175
Using the GET FILES command	175
Does Folder Exist	177
Using the DOES FOLDER EXIST command	177
Is Folder Empty	178
Using the IS FOLDER EMPTY command	178
Copy a Folder	179
Using the COPY A FOLDER command	179
Move a Folder	180
Using the MOVE A FOLDER command	180
Rename a Folder	181
Using the RENAME A FOLDER command	181
Delete a Folder	182
Using the DELETE A FOLDER command	182
Monitor Folder Changes	185

Using the MONITOR FOLDER CHANGES command	185
CHAPTER 10: AI-Powered Document and Text Analysis Commands	
What's Required?	188
AI Booster	188
ABBYY integrations for scanned documents	188
OCR out-of-the-box	189
Azure Licenses	190
OCR: Documents	191
Using the OCR: DOCUMENTS command	191
Get Text (SmartScan)	195
Using the GET TEXT command	195
Get Value	196
Using the GET VALUE command	196
Get Selection Element State	198
Using the GET SELECTION ELEMENT STATE command	198
Does Word Exist (SmartScan)	200
Using the DOES WORD EXIST command	200
Get Date/Time	201
Using the GET DATE/TIME command	201
Save as Image	203
Using the SAVE AS IMAGE command	203
Save to Excel	204
Using the SAVE TO EXCEL command	204
OCR: Printed and Handwritten Text	206
Using the OCR: PRINTED AND HANDWRITTEN TEXT command	206
Get Text (MS Azure)	209
Using the GET TEXT command	209
Does Word Exist (MS Azure)	210
Using the DOES WORD EXIST command	210
Form Recognizer	211
Using the FORM RECOGNIZER command	211
Get Receipt Data	214

Using the GET RECEIPT DATA command	214
Text Analytics: Analyze Sentiment	216
Using the TEXT ANALYTICS: ANALYZE SENTIMENT command	216
Text Analytics: Detect Language	217
Using the TEXT ANALYTICS: DETECT LANGUAGE command	217
Text Analytics: Identify Key Phrases	218
Using the TEXT ANALYTICS: IDENTIFY KEY PHRASES command	218
Convert to Text (SmartScan+)	219
Using the CONVERT TO TEXT (SMARTSCAN+) command	219
Convert to Text (Tesseract)	220
Using the CONVERT TO TEXT (TESSERACT) command	220

CHAPTER 11: Digital PDF Analysis Commands

Analyze Digital PDF File	222
Using the ANALYZE DIGITAL PDF FILE command	222
Page: Get Text	225
Using the PAGE: GET TEXT command	225

CHAPTER 12: External Data Commands

Get From Clipboard	227
Using the GET FROM CLIPBOARD command	227
Clear Clipboard	227
Place in Clipboard	229
Using the PLACE IN CLIPBOARD command	229
Copy Active Field Value	230
Using the COPY ACTIVE FIELD VALUE command	230
Get OS Version	231
Using the GET OS VERSION command	231
Get Input Language	232
Using the GET INPUT LANGUAGE command	232
Read From Registry	233
Using the READ FROM REGISTRY command	234
Remove From Registry	235
Using the REMOVE FROM REGISTRY command	235

Write to Registry	236
Using the WRITE TO REGISTRY command	236
Get Environment Variable	237
Using the GET ENVIRONMENT VARIABLE command	237
Get Windows Event Log Data	238
Using the GET WINDOWS EVENT LOG DATA command	239
Set BI Field	240
Using the SET BI FIELD command	240
Log an Action	241
Using the LOG AN ACTION command	241
Query XML	242
Using the QUERY XML command	242
Query JSON	244
Using the QUERY JSON command	244
Call REST API Method	246
Using the CALL REST API METHOD command	246

CHAPTER 13: Email Commands

Send Email Message	249
Using the SEND EMAIL MESSAGE command	249
Get Email Messages	256
Using the GET EMAIL MESSAGES command	256
Email: Get Data	263
Using the EMAIL: GET DATA command	263
Email: Move to Folder	264
Using the EMAIL: MOVE TO FOLDER command	264
Email: Forward	266
Using the EMAIL: FORWARD command	266
Email: Reply	268
Using the EMAIL: REPLY command	268
Email: Delete	269
Using the EMAIL: DELETE command	269
Email: Mark as Read/Unread	270

Using the EMAIL: MARK AS READ/UNREAD command	270
Email: Save Attachments	271
Using the EMAIL: SAVE ATTACHMENTS command	271
Email: Save Message	273
Using the EMAIL: SAVE MESSAGE command	273

CHAPTER 14: External Program Commands

Run Program	275
Using the RUN PROGRAM command	275
Run .NET Plugin Method	278
Call Web Service Method	280
Run Script	281
Using the RUN SCRIPT command	282
Open URL Link (beta)	284
Using the OPEN URL LINK command	284
Run curl Command	285
Get Web Page HTML	286
Using the GET WEB PAGE HTML command	286
Run JavaScript on Page	287
Using the RUN JAVASCRIPT ON PAGE command	287

CHAPTER 15: Database Commands

Execute SQL Query	290
Using the EXECUTE SQL QUERY command	290
Monitor Database Changes	293
Using the MONITOR DATABASE CHANGES command	293

CHAPTER 16: Credentials Vault Commands

Insert Username Into Active Field	296
Using the INSERT USERNAME INTO ACTIVE FIELD command	296
Insert Password Into Active Field	297
Using the INSERT PASSWORD INTO ACTIVE FIELD command	297
Generate New Password	298
Using the GENERATE NEW PASSWORD command	298
Revert Password	299

Using the REVERT PASSWORD command	299
CHAPTER 17: Robotic Process Automation Commands	
Add Automation Task to Queue	301
Using the ADD AUTOMATION TASK TO QUEUE command	301
Get Automation Task Status	304
Using the GET AUTOMATION TASK STATUS command	304
Get File Trigger Input	306
Using the GET FILE TRIGGER INPUT command	306
Get Folder Trigger Input	307
Using the GET FOLDER TRIGGER INPUT command	307
Get Email Trigger Input	308
Using the GET EMAIL TRIGGER INPUT command	308
Get Database Trigger Input	309
Using the GET DATABASE TRIGGER INPUT command	309
CHAPTER 18: Excel Commands	
Copy from Excel	312
Using the COPY FROM EXCEL command	312
Paste to Excel	318
Using the PASTE TO EXCEL command	319
Delete from Excel	324
Using the DELETE FROM EXCEL command	324
Excel Worksheet Actions	327
Using the EXCEL WORKSHEET ACTIONS command	327
Excel Row Actions	329
Using the EXCEL ROW ACTIONS command	330
Excel Column Actions	332
Using the EXCEL COLUMN ACTIONS command	333
Convert Excel to CSV	335
Using the CONVERT EXCEL TO CSV command	335
Create New Excel File	337
Using the CREATE NEW EXCEL FILE command	337
Query From Excel	338

Using the QUERY FROM EXCEL command	338
Run Macro	340
Using the RUN MACRO command	341
CHAPTER 19: SAP Commands	
Get SAP Object Text	345
Using the GET SAP OBJECT TEXT command	345
Get SAP Object Value	347
Using the GET SAP OBJECT VALUE command	347
Get SAP Object Location	348
Using the GET SAP OBJECT LOCATION command	348
Set SAP Object Value	350
Using the SET SAP OBJECT VALUE command	350
CHAPTER 20: UI Automation Commands	
Get UI Object Text	353
Using the GET UI OBJECT TEXT command	353
Get UI Object Value	355
Using the GET UI OBJECT VALUE command	355
Get UI Object Location	357
Using the GET UI OBJECT LOCATION command	358
Set UI Object Value	360
Using the SET UI OBJECT VALUE command	360
Extract UI table text	362
Click on UI object	364
Does UI Object Exist	366
Click on UI object	368
CHAPTER 21: HTML Commands	
HTML Object Selector	372
Using the HTML object selector	372
Get HTML Table	377
Using the GET HTML TABLE command	377
Get HTML Object Text	379
Using the GET HTML OBJECT TEXT command	379

Get HTML Object Value	382
Using the GET HTML OBJECT VALUE command	382
Get HTML Object	384
Using the GET HTML OBJECT command	384
Set HTML Object Value	386
Using the SET HTML OBJECT VALUE command	386
Does HTML Object Exist	388
Using the DOES HTML OBJECT EXIST command	388
Click on HTML Object	390
Using the CLICK ON HTML OBJECT command	390
Scroll to HTML Object	392
Using the SCROLL TO HTML OBJECT command	392
Extract from HTML Table/List	394
Using the EXTRACT FROM HTML TABLE/LIST command	394
Selecting the data to extract	394
Finalize command options	402

CHAPTER 22: .NET Automation Commands

Get .NET Object Text	406
Using the GET .NET OBJECT TEXT command	406
Get .NET Object Value	407
Using the GET .NET OBJECT VALUE command	407
Get .NET Object Location	408
Using the GET .NET OBJECT LOCATION command	408
Set .NET Object Value	410
Using the SET .NET OBJECT VALUE command	410

CHAPTER 23: Java Automation Commands

Support	411
Get Java Object Text	413
Using the GET JAVA OBJECT TEXT command	413
Get Java Object Value	415
Using the GET JAVA OBJECT VALUE command	415
Get Java Object Location	417

Using the GET JAVA OBJECT LOCATION command	418
Set Java Object Value	420
Using the SET JAVA OBJECT VALUE command	420
Extract Java Table Text	422
Does Java Object Exist	424
Click on Java Object	426
CHAPTER 24: Global Variable Commands	
Set Global Variable	430
Using the SET GLOBAL VARIABLE command	430
Get Global Variable	431
Using the GET GLOBAL VARIABLE command	431
Delete Global Variable	432
Using the DELETE GLOBAL VARIABLE command	432
CHAPTER 25: Scripting Commands	
Note	434
Using the NOTE command	434
Notes for Mobile/Web Access	435
Using the NOTES FOR MOBILE/WEB ACCESS command	435
View Variable List	436
Using the VIEW VARIABLE LIST command	436
Show Debug Message	438
Using the SHOW DEBUG MESSAGE command	438
APPENDICES	
APPENDIX A: Timeout Monitoring Support for Advanced Commands	
Monitoring Types	440
Configuration	440
To enable Timeout Monitoring for an Advanced Command:	440
Timeout Monitoring Availability	441
APPENDIX B: Timeout Monitoring for Click-Blocked	
APPENDIX C: Error Handling	
Using ERROR HANDLING options	444
Error handling fields of Global Function	445

[APPENDIX D: Kryon Connector Extension](#)

[APPENDIX E: Installing Kryon Java Bridge](#)

[APPENDIX F: Java Supported Controls and Accessibility](#)

[APPENDIX G: Editing the UI Automation Selector Syntax](#)

CHAPTER 1: Variable Commands

In this chapter:

Set Value	18
Find	20
Replace	28
Mathematics	30
Evaluate Expression	31
Split	36
Get Array Data	40
Get Table Data	44
Table Lookup	49
Get ASCII Character	54
Remove Blank Spaces	56
Change Case	57
Check Type	58
Get Random Number	59
Reverse	60
Get Length	61
Extract Numeric Values	62

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 the following fields:

- In the variable:** A dropdown menu with 'separator' selected.
- Set the value:** A text input field containing the character '/'. There is a help icon (i) to the right of the field.

At the bottom of the dialog are 'OK' and 'Cancel' buttons, and a help 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 the following fields:

- In the variable:** A dropdown menu with 'date' selected.
- Set the value:** A text input field containing the string '08\$separator\$10\$separator\$1967'. There is a help icon (i) to the right of the field.

At the bottom of the dialog are 'OK' and 'Cancel' buttons, and a help 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

The screenshot shows a 'Find' dialog box with the following elements and numbered callouts:

- 1**: A dropdown menu labeled 'In the variable:' with the placeholder text 'Type a variable name'.
- 2**: A dropdown menu labeled 'Find text'.
- 3**: A text input field for 'Search start position' containing the value '1', and a dropdown menu for 'Search direction' set to 'Start to end'.
- 4**: A checkbox for 'Allow close match' (unchecked), a slider for 'Required accuracy' set to 100%, and a 'Test...' button.
- 5**: A checkbox for 'Ignore letter case' (checked).
- 6**: A dropdown menu labeled 'Return the result in:' with the placeholder text 'Type a variable name'.

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

- 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.

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

- In the variable:** dropdown menu set to `quote`.
- Find text:** dropdown menu set to `th`.
- Search start position:** input field set to `1`.
- Search direction:** dropdown menu set to `Start to end`.
- Allow close match:** checkbox is unchecked.
- Required accuracy:** slider is set to 100%.
- Test..** button is visible.
- Example:** A text box containing: "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:** checkbox is unchecked.
- Return the result in:** dropdown menu set to `find result`.
- Buttons:** `OK` and `Cancel` buttons are at the bottom right.

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**: 'Find text matching a regular expression' dropdown menu.
- 3**: 'Search type:' dropdown menu with 'Full match'.
- 4**: 'Pattern:' text input field.
- 5**: 'Match delimiter:' text input field.
- 6**: 'Test...' section containing an 'Example:' box with the text: 'Finding the regular expression `(?:\d*\.)?\d+` will return all decimal numbers in the given variable.'
- 7**: 'Ignore letter case' checkbox, which is checked.
- 8**: 'Return the result in:' dropdown menu with 'Select a variable'.

At the bottom of the dialog, there is an information icon (i), '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://regexr.com/foo.html?q=bar
                https://mediatemple.net
```

Find
✕

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..](#)

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

?
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>
```

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.

Replace

In the variable:
quote

Replace the text
think

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

Ignore letter case

With:
breathe

Replace all
 Replace first
 Replace last

[i](#) OK Cancel

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

Mathematics

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

Using the MATHEMATICS command

- 1 Enter the 2 values on which you want to perform the calculation and select the operation you wish to perform. The available operations are:
 - Addition
 - Subtraction
 - Multiplication
 - Division
 - Modulus (returns the remainder obtained when dividing the first value by the second)
- 2 Enter the name of the variable into which you'd like 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 it in **cv text**.

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

Split

Split:
cv text

By characters:
First occurrence of the delimiter Name: Match case

By character position:
 characters from the start

Into the variables:
trash and cv text

Remove blank spaces around results

OK Cancel

Result:

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

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

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

Result:

`ename` = Harry S. Truman

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

Get Array Data

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

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



NOTE

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

Using the GET ARRAY DATA command

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

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

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

- 1 Enter the name of the variable from which you want to obtain data
- 2 Enter the delimiter or the characters that appear before and after each item in the array
- 3 Choose the type of data you want to obtain. For a mathematical calculation, select the operation.
- 4 Enter the name of the variable into which you'd like 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
```

Get array data

From the array stored in:
daily orders

Where items are delimited by: ;

Where items are preceeded by:

and followed by:

Get the value of item #

Get the total number of items

Get the maximum value

Return the result in variable:
number of orders

OK Cancel

Result: `⚡ number of orders` = 7

Get array data

From the array stored in:
daily orders

Where items are delimited by: ;

Where items are preceeded by: and followed by:

Get the value of item #

Get the total number of items

Get the average

Return the result in variable:
average order

OK Cancel

Result: ⚡ average order = 237.51

Get array data

From the array stored in:
daily orders

Where items are delimited by: ;

Where items are preceeded by: and followed by:

Get the value of item #

Get the total number of items

Get the maximum value

Return the result in variable:
largest order

OK Cancel

Result: ⚡ largest order = 645.76

Get Table Data

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

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

3

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

<div style="background-color: #f0f0f0; padding: 5px; margin-bottom: 10px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Single value ▾</div> <p>Column number: <input style="width: 50px;" type="text"/> Row number: <input style="width: 50px;" type="text"/></p> </div> <div style="background-color: #f0f0f0; padding: 5px; margin-bottom: 10px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Number of items ▾</div> </div> <div style="background-color: #f0f0f0; padding: 5px; margin-bottom: 10px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Number of rows ▾</div> </div> <div style="background-color: #f0f0f0; padding: 5px; margin-bottom: 10px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Number of columns ▾</div> </div> <div style="background-color: #f0f0f0; padding: 5px; margin-bottom: 10px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Entire row ▾</div> <p>Row number: <input style="width: 50px;" type="text"/></p> </div> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Data to get:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Entire column ▾</div> <p>Column number: <input style="width: 50px;" type="text"/></p> </div>	<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>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>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>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>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 2 <p>Entire column:</p> <p>The wizard will retrieve all the values in a specific column</p>
--	--

- 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:

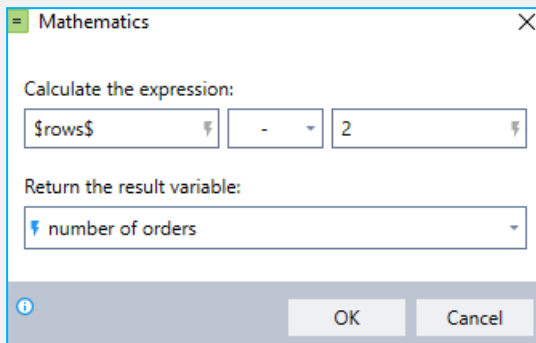
```
⚡ daily order table =Order Number, Order Total, Number of
Items
34567, 350.00, 10
34568, 785.00, 15
34569, 649.00, 14
34570, 134.00, 1
34571, 100.00, 1
Total, 2018.00, 41
```

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 options:

- 1**: 'From the table stored in:' dropdown menu.
- 2**: 'Column delimiter:' and 'Row delimiter:' dropdown menus.
- 3**: 'Find:' dropdown menu.
- 4**: 'equals (ignore case)' dropdown menu.
- 5**: Empty text input field.
- 6**: Radio buttons for 'Search the entire table', 'Search column:', and 'Search row:'. 'Search column:' and 'Search row:' have associated dropdown menus.
- 7**: 'Return:' dropdown menu.
- 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)

Return:

Column #:

Row #:

Return:

Output variable:

Return:

Output variable:

Return:

Column:

Address (column & row):

The wizard will return the location of the lookup value

- Enter the name(s) of the variable(s) in which to return the value(s) retrieved
 - 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

Entire row:

The wizard will return all the values in the lookup value's row

- Enter the name of the variable in which to return the values retrieved

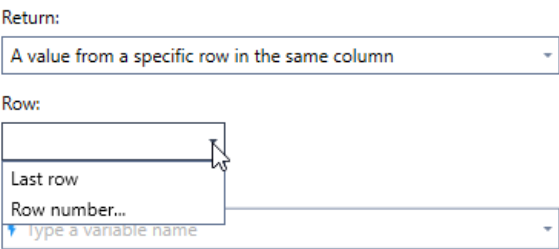
Entire column:

The wizard will return all the values in the lookup value's column


- Enter the name of the variable in which to return the values retrieved

A value from a specific column in the same row:

- Select whether you want the wizard to return:
 - a.** The value from the last column in the lookup value's row; **or**


 <p>The screenshot shows two dropdown menus. The first is labeled 'Return:' and has 'A value from a specific row in the same column' selected. The second is labeled 'Row:' and has 'Last row' selected. A mouse cursor is pointing at the 'Last row' option.</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) <p>NOTE: This should be the actual column number, NOT the column's position relative to the lookup value</p> <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; or b. The value from a specific row in the lookup value's column (and enter the row number) <p>NOTE: This should be the actual row number, NOT the row's position relative to the lookup value</p> <ul style="list-style-type: none"> • Enter the name of the variable in which to return the value(s) retrieved
--	--

<p>Return:</p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">The matching value itself</div> <p>Output variable:</p> <div style="border: 1px solid #ccc; padding: 2px;">Select a variable</div>	<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

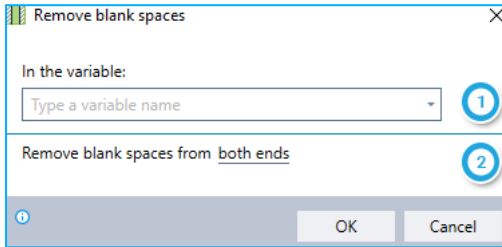
At the end of each day, you download data from a web application for further analysis. The web application uses <Page Break> to separate individual records, so you also need to separate the data at each <Page Break>. How can you make this happen?

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



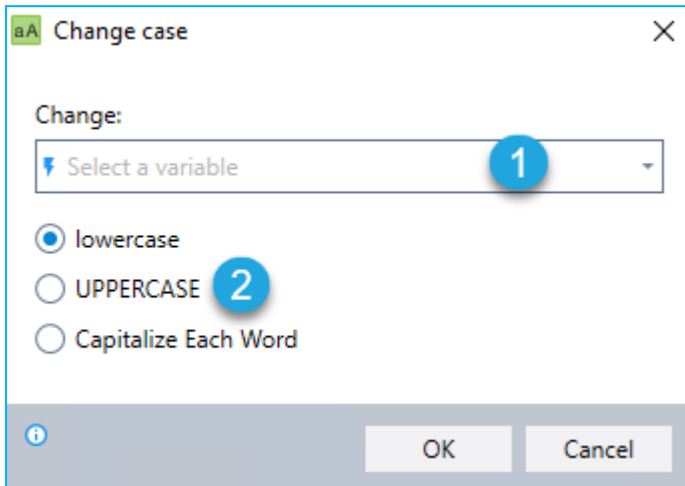
TIP

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

Change Case

Change the letter case within a variable.

Using the CHANGE CASE command



1. Enter the name of the variable for which you'd like to change letter case.
2. Select to change to all lowercase, to all uppercase, or to capitalize the first letter of each word.

Check Type

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

Using the CHECK TYPE command

- 1 Enter the name of the variable whose value you'd like to check; *and* Choose the type of value you'd like to check for (numeric or textual)
- 2 Enter the name of the variable into which you'd like 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

The screenshot shows a dialog box titled "Get random number". It has a close button (X) in the top right corner. The main content is divided into two sections. The first section is labeled "Get a random number between:" and contains two input fields separated by the word "and". A blue circle with the number "1" is positioned to the right of the second input field. The second section is labeled "Return the result 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 the dropdown menu. 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 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:

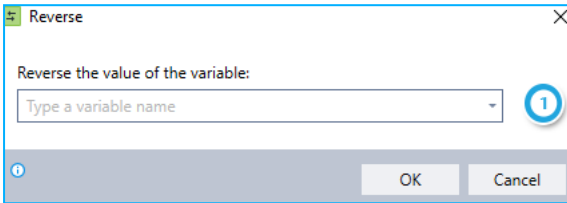
Creating single-use login codes for other applications

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

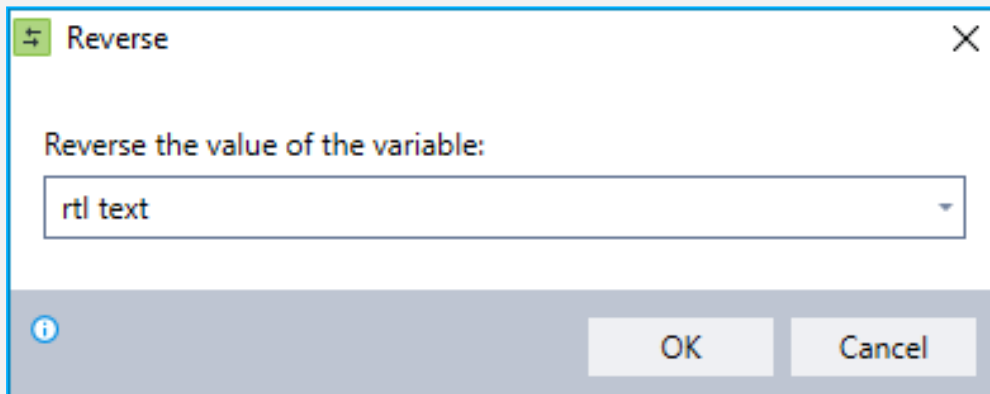


- 1 Enter the name of the variable whose value you'd like to reverse



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". 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 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

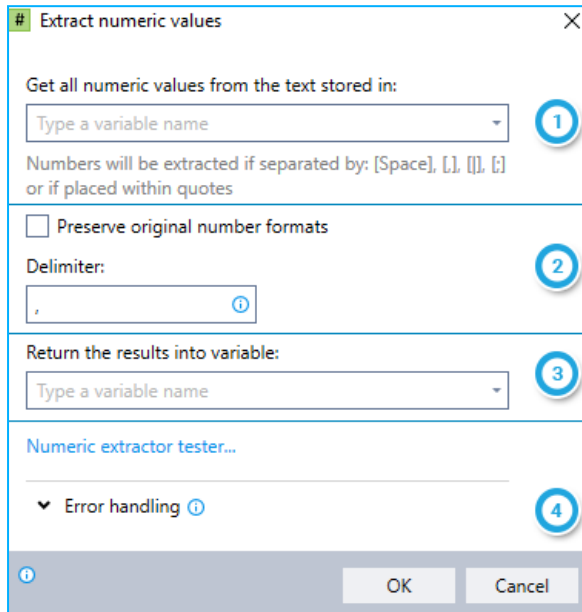
Why use GET LENGTH?

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.

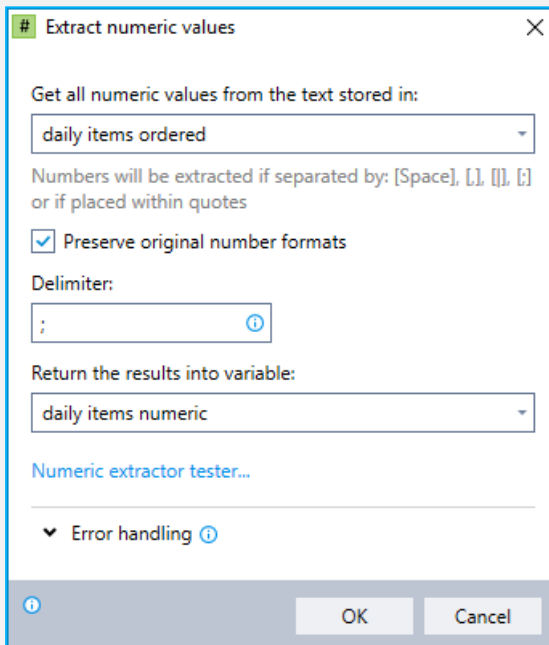
If you elect to preserve original number formats (and there is no space between a number and the currency symbol preceding it), the currency symbol will be preserved in the output.



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
```



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	66
Complex If Else	70
Multi-Value If Else	72
Loop	74
Loop: Break	76
Loop: Restart	77
Loop Table	78
Loop Items	80
Pause	82
Group	83
Global Functions	86

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

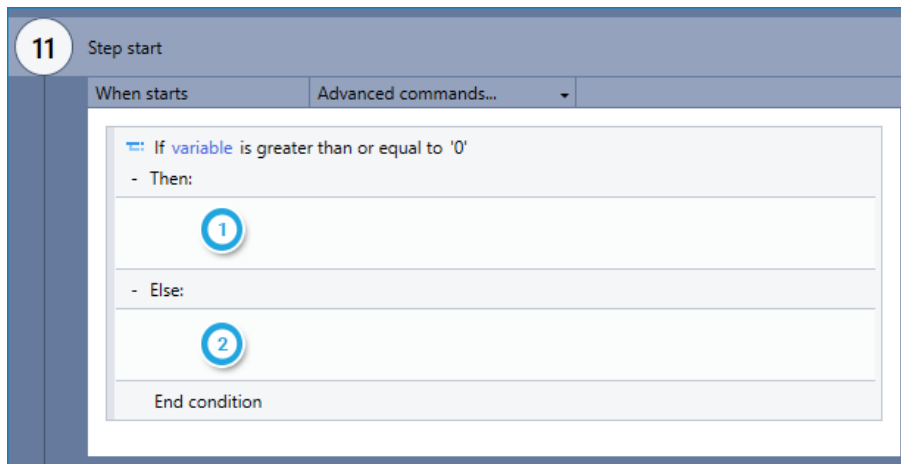
The first step in using the **IF ELSE** command is to define the condition (i.e., set up a comparison).

- 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.

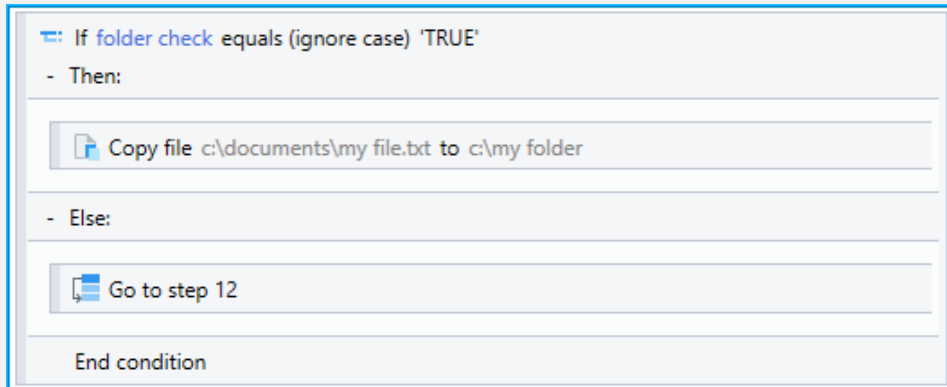
- If it does (the condition is TRUE), you want to copy a file to it
- If it doesn't (the condition is FALSE), you want to skip to the next step

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

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

- 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

If else

If
insured age is greater than or equal to 65
 Reverse comparison (If not)

Else if
insured age is greater than or equal to 25
 Reverse comparison (If not) Remove

Else if...

OK Cancel

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**

If insured age is greater than or equal to '65'

- Then:

Set rate table to 'RT1'

- Else if insured age is greater than or equal to '25'

Set rate table to 'RT2'

- Else:

Set rate table to 'RT3'

End condition

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).

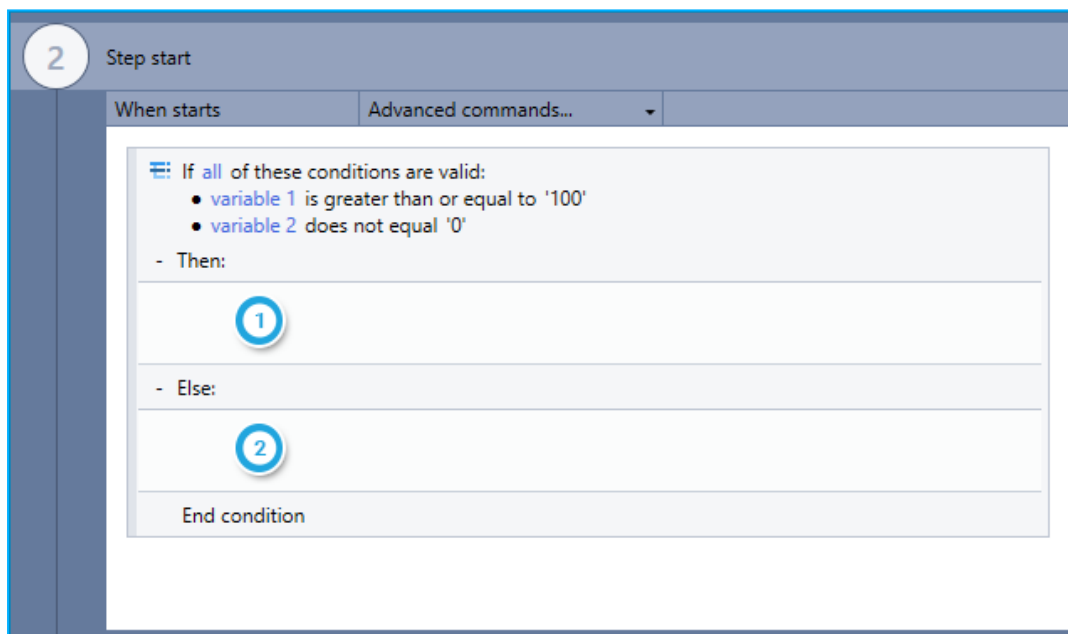
- 1 Choose whether **any** or **all** of the conditions must be true in order for the overall outcome to be TRUE
- 2 For each condition:
Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
 - equals (with options to 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

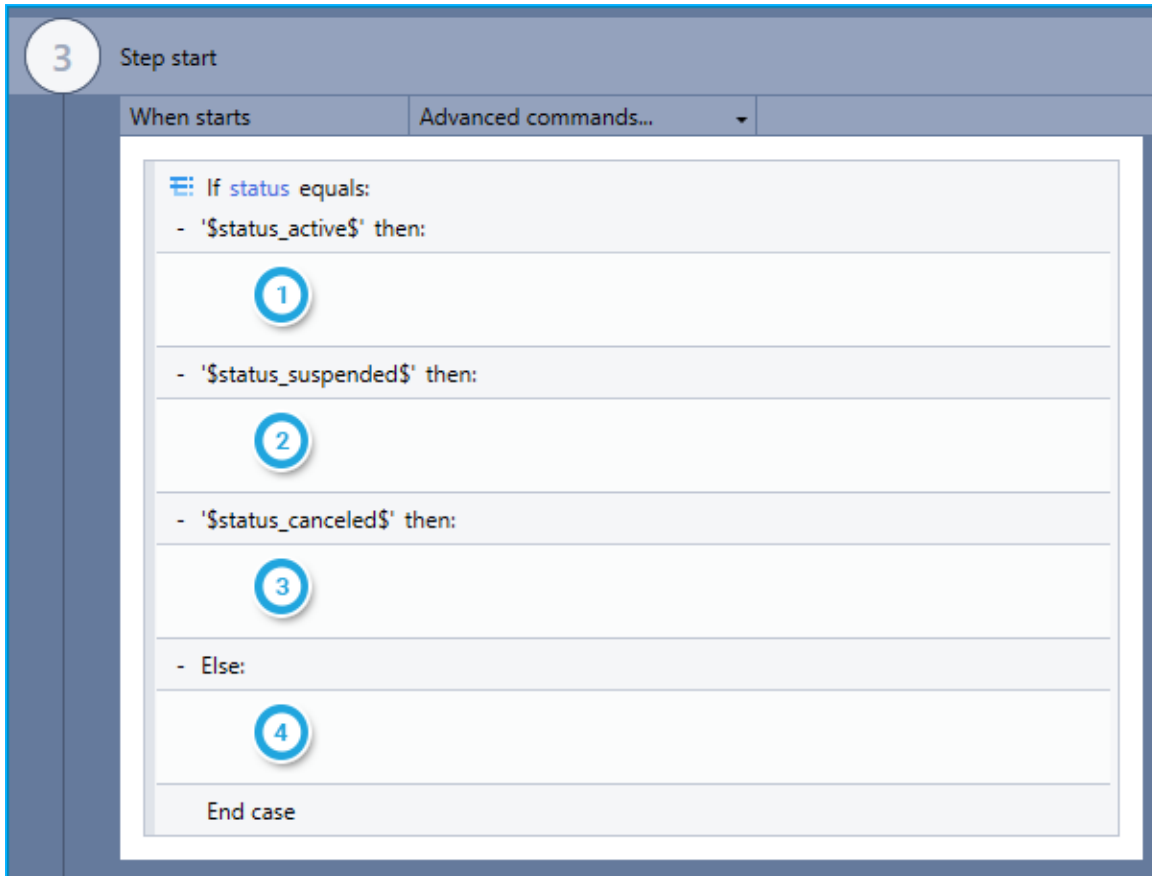
Step #1 - Define the conditions

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

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).

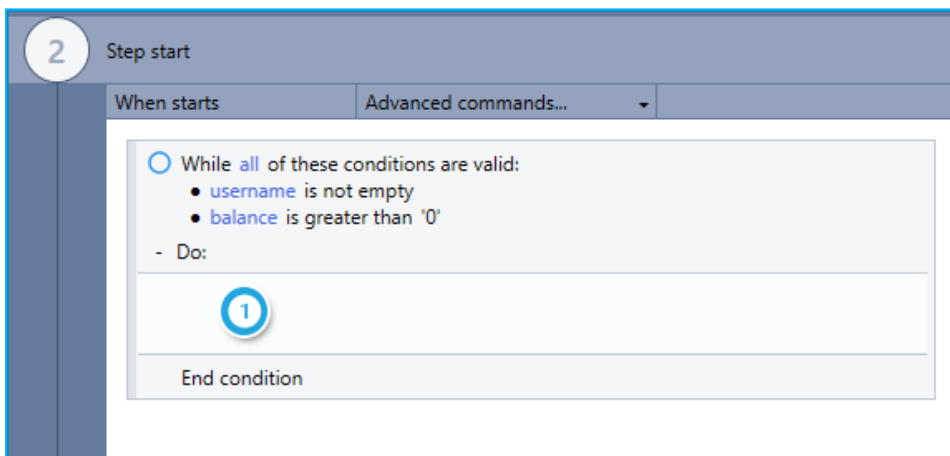
- 1 Choose whether **any** or **all** of the conditions must be true in order for the loop to continue
- 2 For each condition:
Enter the name of the variable whose value you wish to compare with another value
- 3 Select the type of comparison you wish to perform:
 - equals (with options to ignore letter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

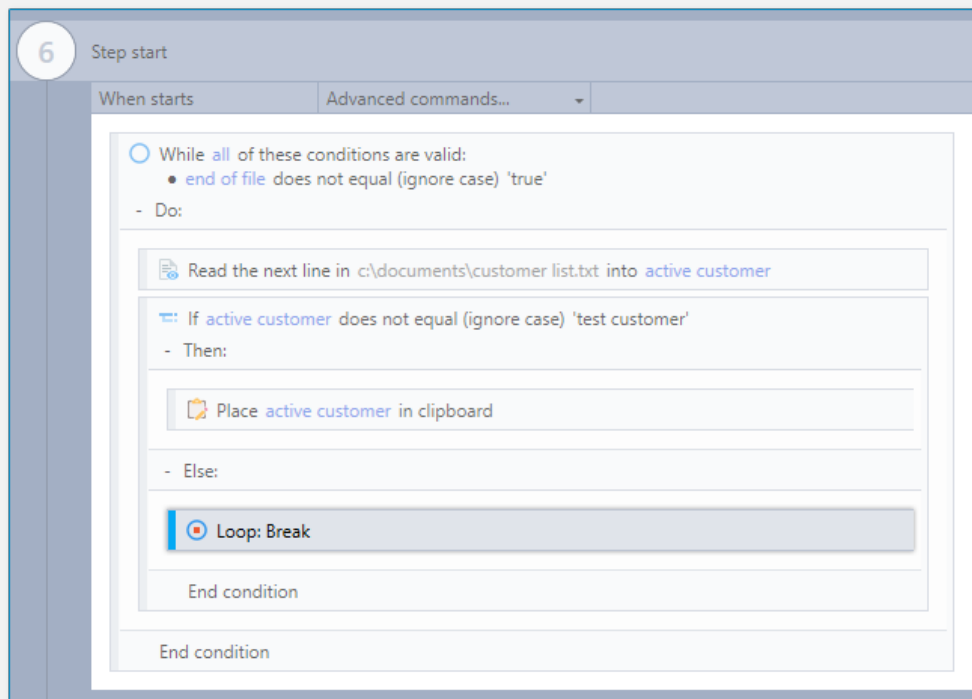


NOTE

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.

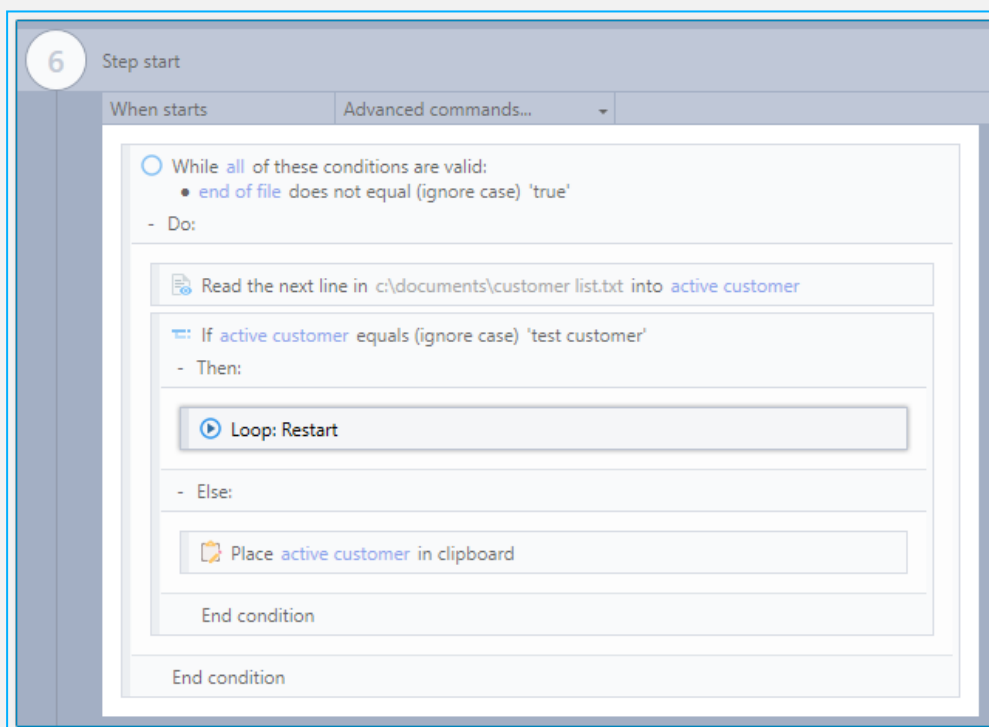


NOTE

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:

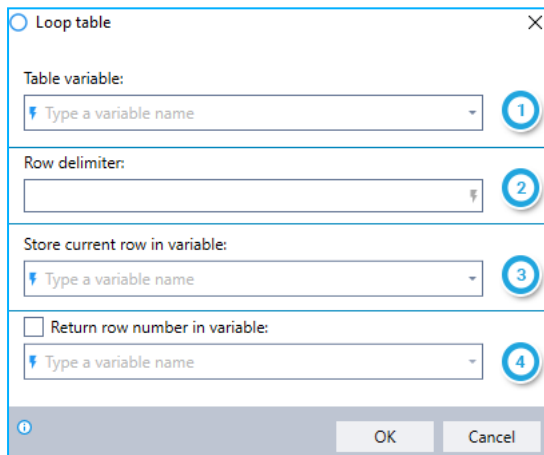
- Place the current row into a new or existing variable
- Optionally place the row number into a new or existing variable; **and**
- Perform a specified action or series of actions

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 sections, each with a numbered callout (1-4) in a blue circle:

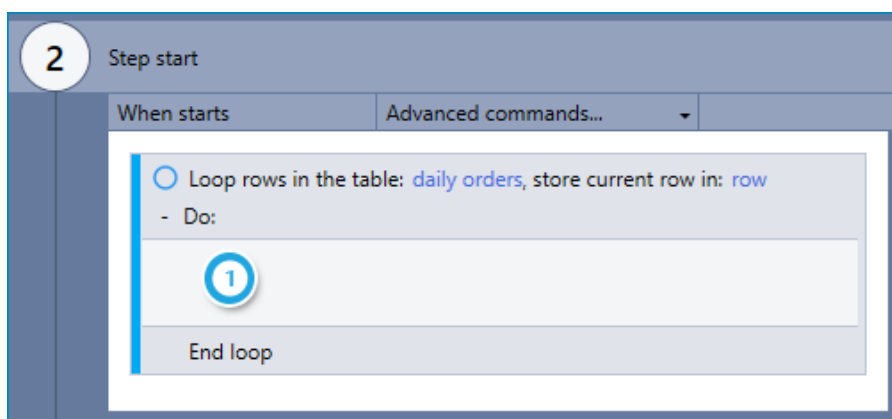
- 1** "Table variable:" followed by a dropdown menu containing the text "Type a variable name".
- 2** "Row delimiter:" followed by a text input field.
- 3** "Store current row in variable:" followed by a dropdown menu containing the text "Type a variable name".
- 4** "Return row number in variable:"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

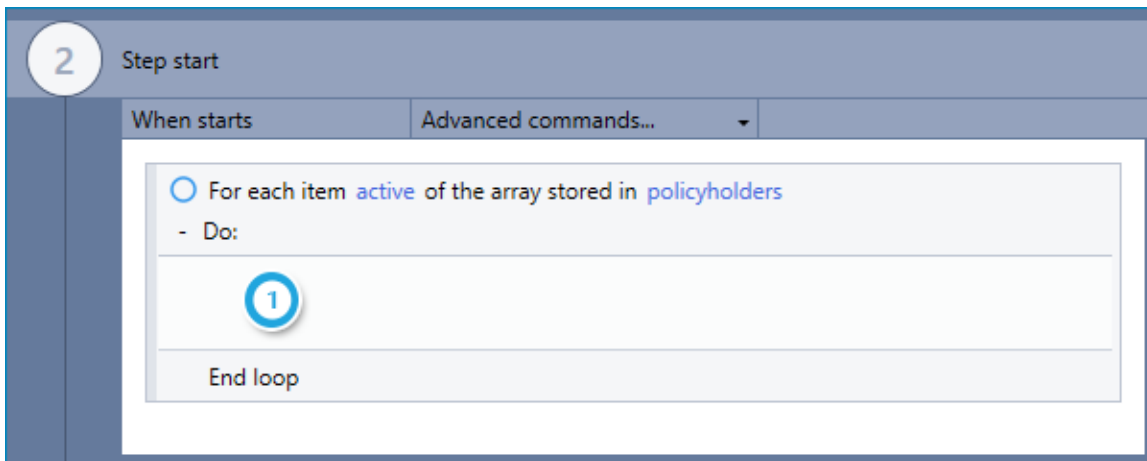
OK Cancel

- 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

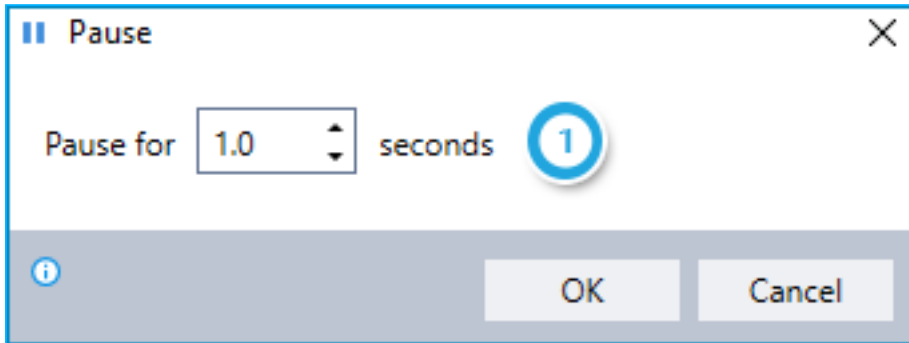
Break that loop!

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)

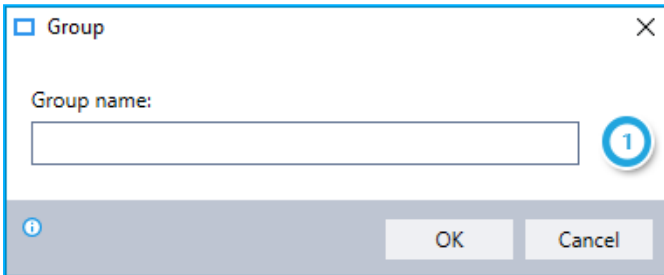
Group

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

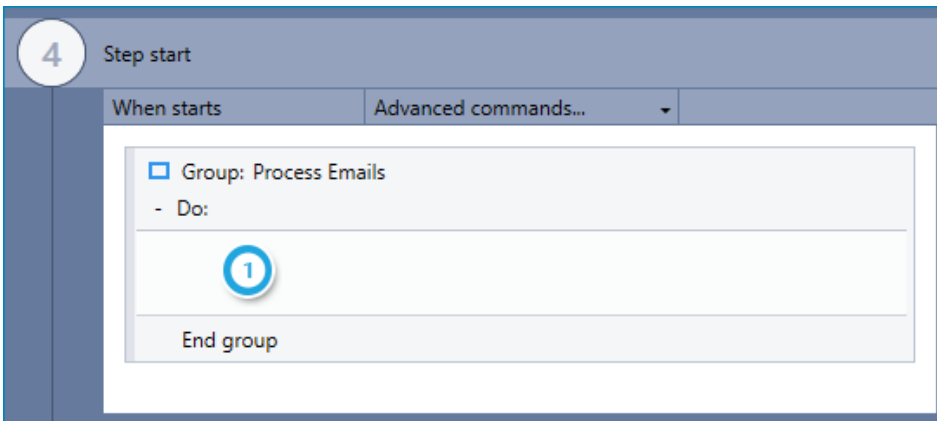
Step #1 - Name the group



- 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

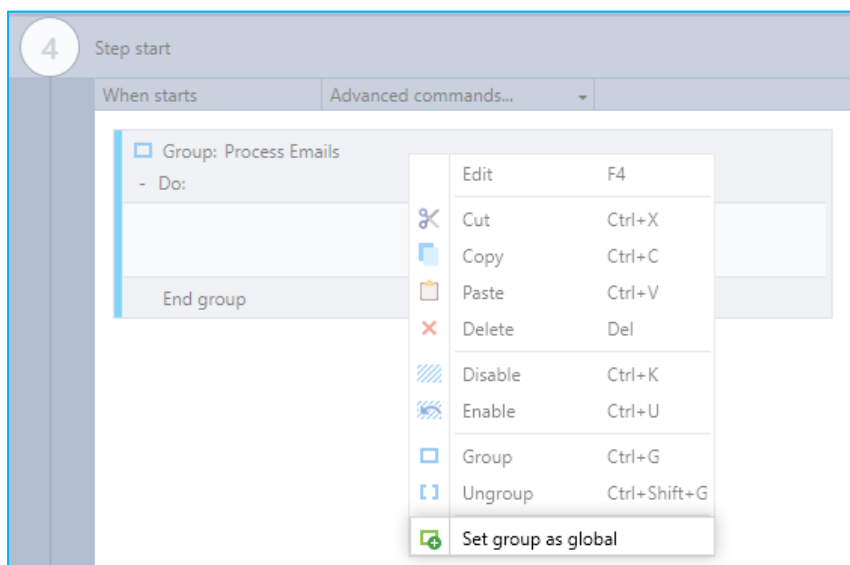
If the group you've created is one you expect to use numerous times throughout the wizard, you might want to make it global.

**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.

Global Functions

The Global Function feature lets you create a customized self-made advanced command that contains of a set of advanced commands of your choice. Once you create a Global Function, you can apply it to any Wizard in the Company.

USAGE

Imagine there is a repetitive fixed set of actions (a set of advanced commands) that need to occur more than once across a wizard, or across several wizards. Instead of re-creating and re-configuring the same set of advanced commands over and over again in the relevant wizard(s), you can create them *ONCE* as a Global Function, and apply the Global Function in different wizard (s).

KNOWN LIMITATION

Since the Global Function is created by compiling a set of advanced commands and *NOT* a recorded session, you cannot insert into it advanced commands that require only recorded steps.

- Workaround: Advanced commands that require recorded steps (e.g. mouse click) should be inserted into a Global Group.

AN OVERVIEW OF HOW IT WORKS

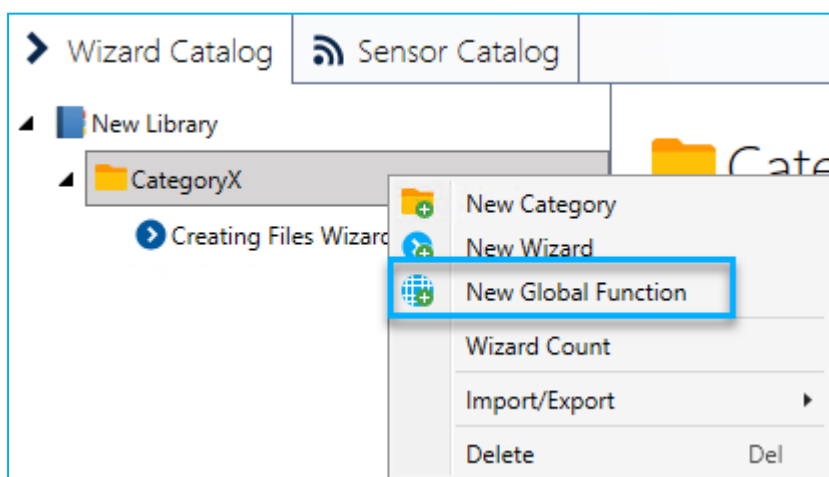
1. [Creating a Global Function](#)
2. [Adding the global function to the relevant wizard\(s\)](#)

Note: A Global Function is applicable across the wizards in the same Company only; a Global Function created under *CompanyA* cannot be applied in *CompanyB*.

Creating Global Function

STEP 1: CREATE A NEW GLOBAL FUNCTION

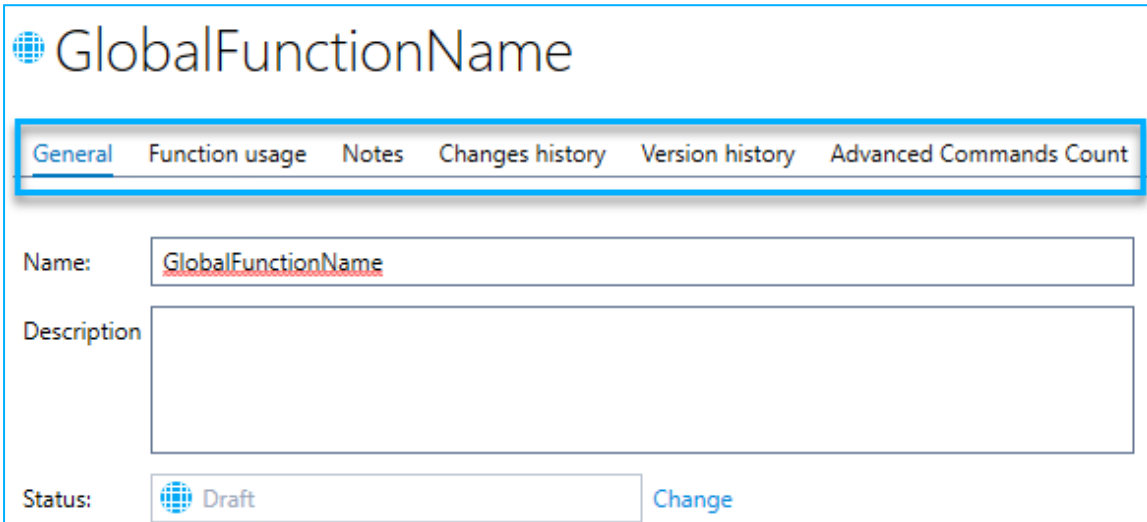
Option 1: Right-click a category > **New Global Function**



Option 2: Click a category > Select **New Global Function** at the bottom action pane



STEP 2: CONFIGURE AND SAVE THE GLOBAL FUNCTION



General Set the name of the Global Function.

Function usage: A view of published wizards that use this global function (this view is populated once the global function is applied in a wizard)

- Save your changes.

STEP 3: GET STARTED WITH THE GLOBAL FUNCTION

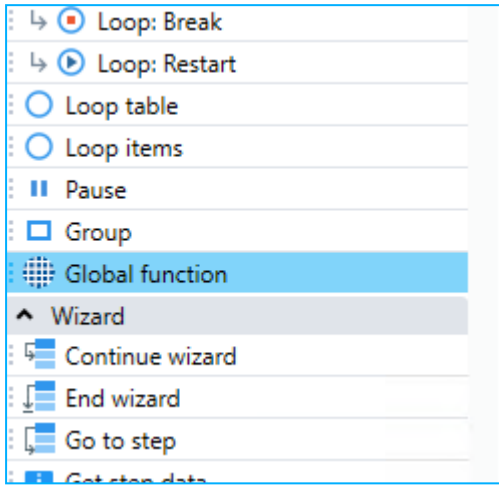
- a. Click **GET STARTED** to open the Global Function editor.
- b. Build the set of advanced commands that make the global function

Note: The Global Function editor is significantly simpler than the regular Wizard editor. It doesn't contain the regular wizard editor features, such as Bubbles or Core actions. It contains only two steps - a starting and an ending one - and a list of Advanced Commands you can use.

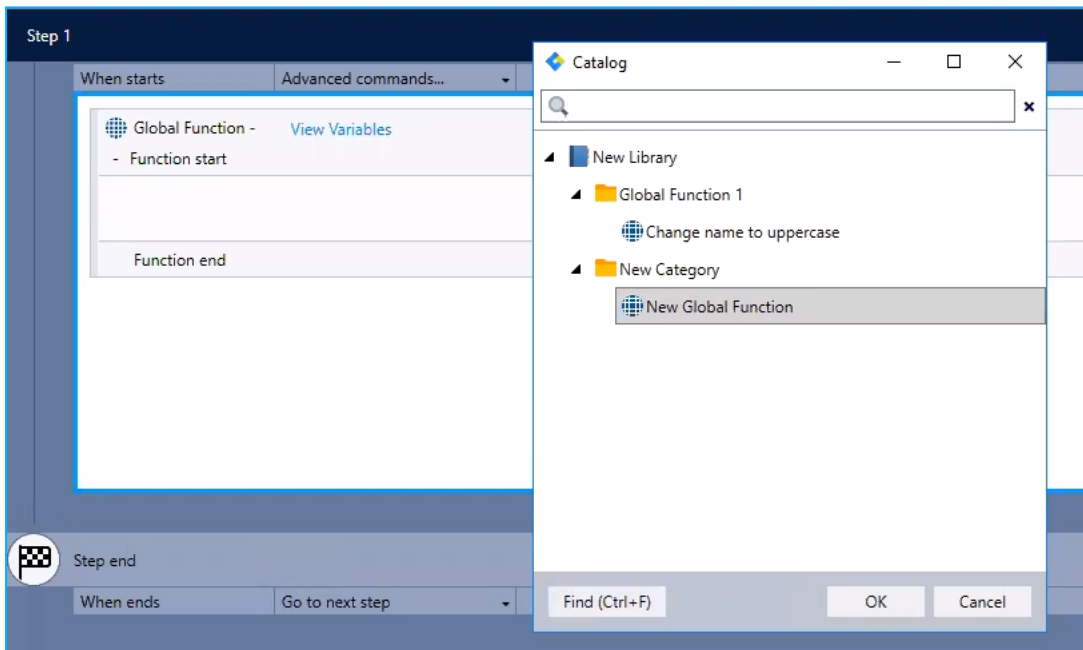
- c. Save
- d. Publish the global function by changing its status to **Published**.
 - **Note:** For the Global Function to be applicable and used in wizards, its status must be **Published**.

Adding the global function to wizards

- a. Open/Edit your wizard in the Advanced Command view.
- b. Drag the Global Function advanced command into the steps editor:

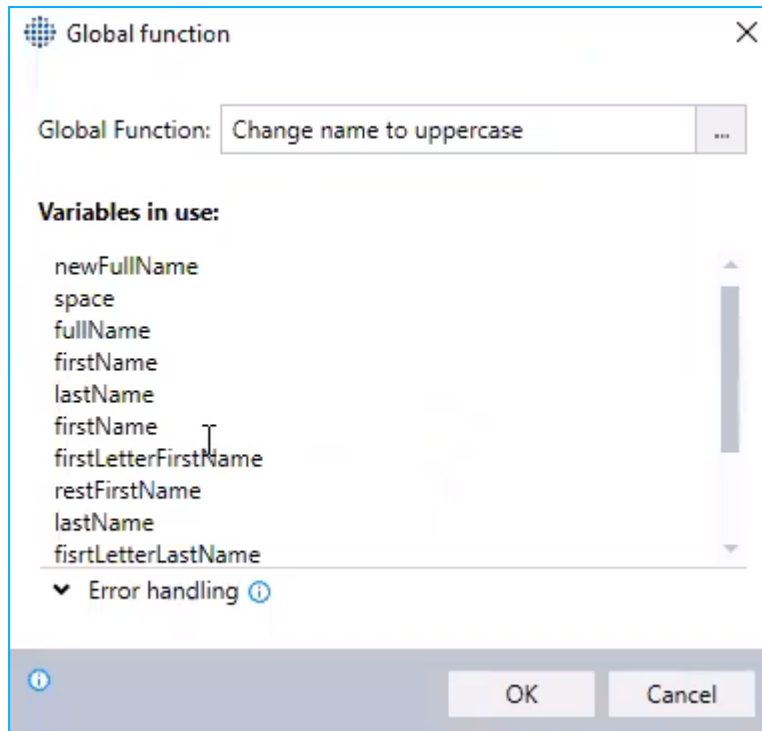


- c. Select the relevant global function (you can also search for it):



Note: The catalog presents only published global functions.

- d. Once you select a global function, its variables are presented for your reference.
Example:



Tip: You can highlight and copy the variables to your clipboard to use later in the wizard.

- e. Instruct the wizard how to handle any errors encountered - if relevant (you can also leave it as is).

Read more about [Error Handling](#).

- f. **Optional:** You can [further customize](#) the selected global function by adding to it more advanced commands. Any commands you drag into a global function in a wizard will be applicable only to this specific wizard.

Added commands will appear in the white container (between **Function start** and **Function end**).

Example of a global function that is customized in a wizard

Lets take a look at the following global function. Its purpose is to make sure that the first letter in the first and last name is capitalized.

Step 1

1 Step start


When starts	Advanced commands...
	<div style="background-color: #D9E1F2; padding: 2px; margin-bottom: 2px;"> 🔍 Get the character represented by the ASCII code 32 into space </div> <div style="background-color: #D9E1F2; padding: 2px; margin-bottom: 2px;"> ⋮ Split fullName into firstName and lastName (delimiter: '\$space\$') </div> <div style="background-color: #D9E1F2; padding: 2px; margin-bottom: 2px;"> ⋮ Split firstName into firstLetterFirstName and restFirstName </div> <div style="background-color: #D9E1F2; padding: 2px; margin-bottom: 2px;"> ⋮ Split lastName into firsrtLetterLastName and restLastName </div> <div style="background-color: #D9E1F2; padding: 2px; margin-bottom: 2px;"> aA Change firstLetterFirstName to uppercase </div> <div style="background-color: #D9E1F2; padding: 2px;"> aA Change firsrtLetterLastName to uppercase </div>


🏁 Step end

When ends	Advanced commands...
	<div style="background-color: #D9E1F2; padding: 2px;"> 📄 Set newFullName to '\$firstLetterFirstName\$\$restFirstName\$\$space\$\$firsrtLetterLastName\$\$restLastName\$' </div>


Now lets open the wizard where this global function is applied in. As you can see, there is an additional advanced command to backup the old full name in this specific wizard.

Step 1


 Set `fullName` to 'eylon lieberman'

 Global Function - Change name to uppercase [View Variables](#)

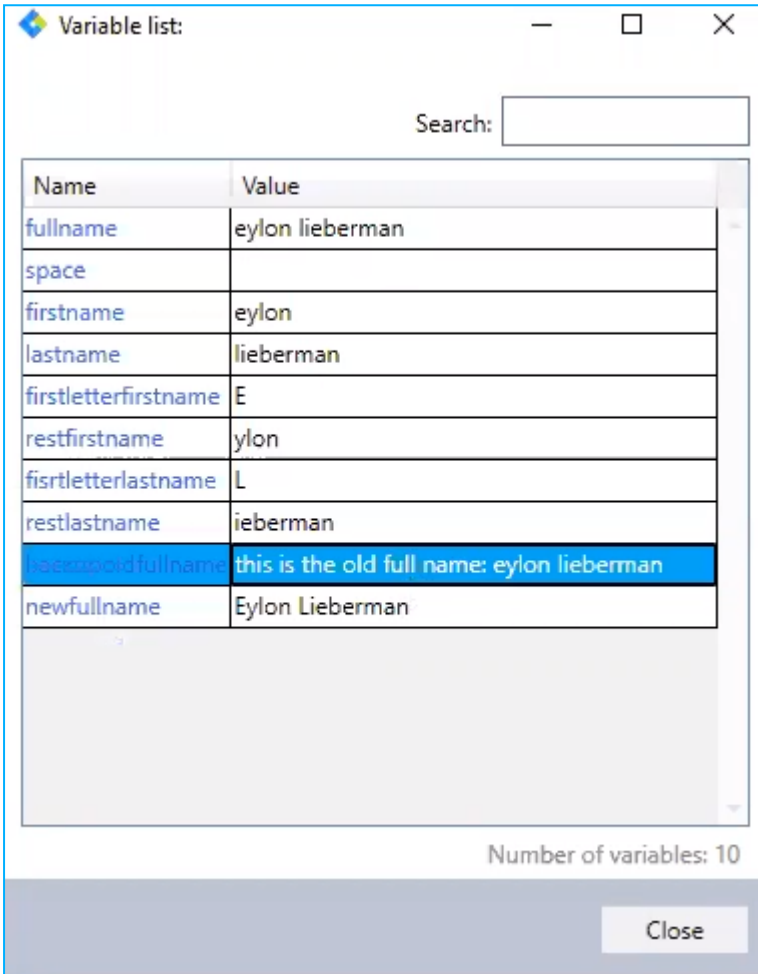
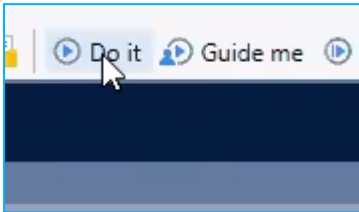
- Function start

 Set `backupOldFullName` to 'this is the old full name: \$fullName\$'

Function end

 View variable list:

If you click **Do it**, the process will run. When it is done, you'll see the process **Variable list** (as set in the original global function) and the additional customized Set value command (that is added to this specific wizard):



So what happened here? An example of a possible scenario:

Making sure the first letter of the first and last name is capitalized is a task we want to have applied across different wizards / multiple times, for every excel file that contains full names. Therefore, we've created the whole repetitive process one time, as a global function. We started applying this global function in different wizards - whenever and wherever it is relevant. In one of the wizards, we decided that there is a need to backup the original names as well, as they were originally typed. Therefore, we added an additional relevant advanced command into the container of the global function

CHAPTER 3: Wizard Commands

In this chapter:

Continue Wizard	94
End Wizard	95
Go To Step	96
Get Step Data	98
Get Wizard Data	99
Check Application	100
Get User Data	101
Check Run Mode	102
Check Video Recording Mode	103
Set User Interrupt Mode	104
Show Message	105
Raise Wizard Error	107
Get Last Failure Type	108
Resume Error	109
Report Wizard Output	110

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.

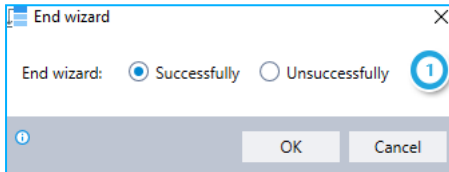
End Wizard

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

The screenshot shows a dialog box titled "Get wizard data". It has a close button (X) in the top right corner. The first section is labeled "Get the current" and contains a dropdown menu with "wizard ID" 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 are "OK" and "Cancel" buttons.

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

Check 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 (X) and a help icon (i). The main area contains the following elements:

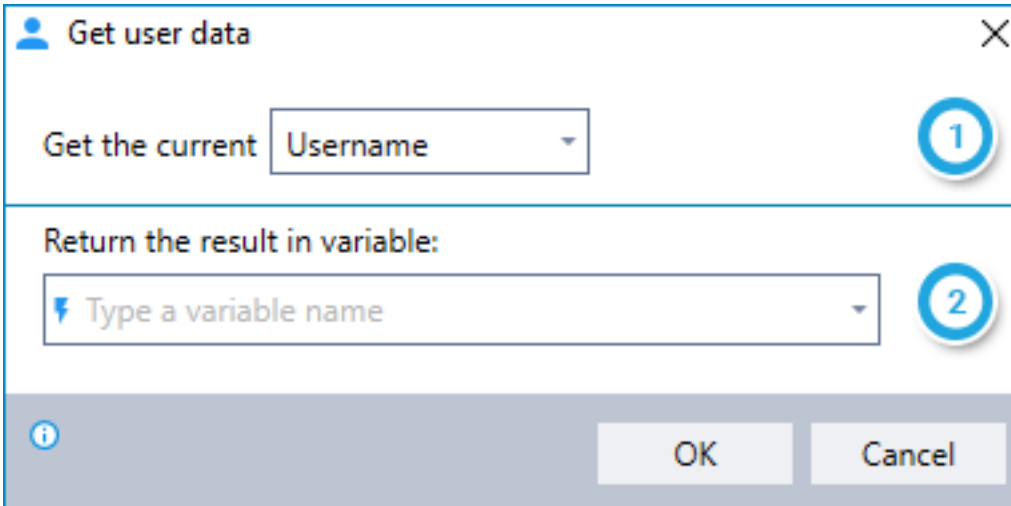
- Header: 'Enter values manually or use the Service Discovery tool' with a 'Discover Services...' button (callout 1).
- Web Service URL: A text input field (callout 2).
- Options: Radio buttons for 'Call Method' (selected) and 'Send SOAP xml'.
- Protocol: A dropdown menu showing 'HttpPost' (callout 3).
- Service: A text input field.
- Method: A text input field.
- Credentials: Radio buttons for 'No credentials' (selected), 'Manual', and 'From vault' (callout 4).
- Parameters: A table with columns 'Name', 'Type', and 'Value'. The first row is a header, and the second row contains a '+' icon and the text 'Enter new parameter name'.
- Return result in variable: A dropdown menu with the placeholder text 'Type a variable name'.
- Service timeout: A spinner box set to '30' with the unit 'seconds'.
- Error handling: A collapsed section with a help icon (i).
- Buttons: 'OK' and 'Cancel' buttons at the bottom right.

- 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



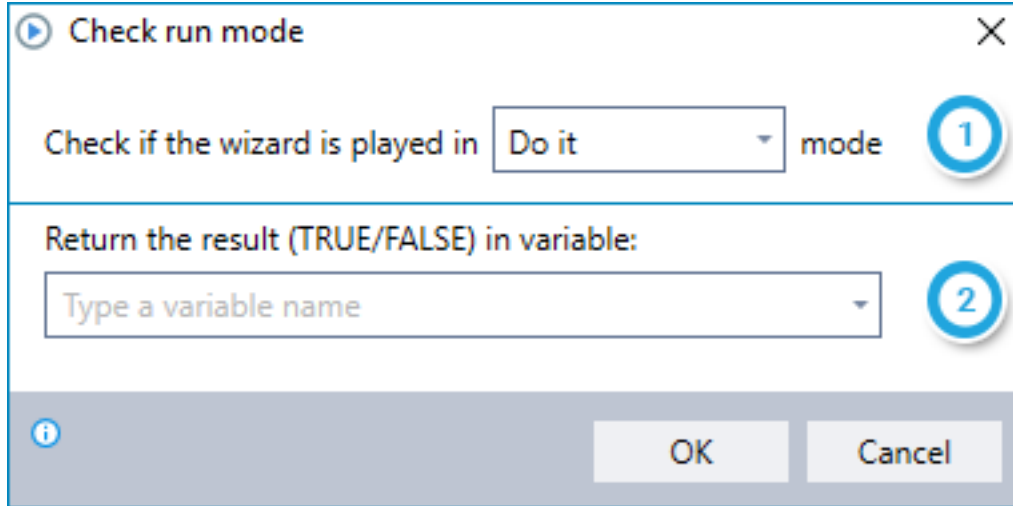
The screenshot shows a dialog box titled "Get user data". It contains two main sections. The first section is labeled "Get the current" and has a dropdown menu with "Username" selected. A blue circle with the number "1" is next to it. The second section is labeled "Return the result in variable:" and has a text input field with "Type a variable name" entered. A blue circle with the number "2" is next to it. At the bottom, there are "OK" and "Cancel" buttons.

- 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:

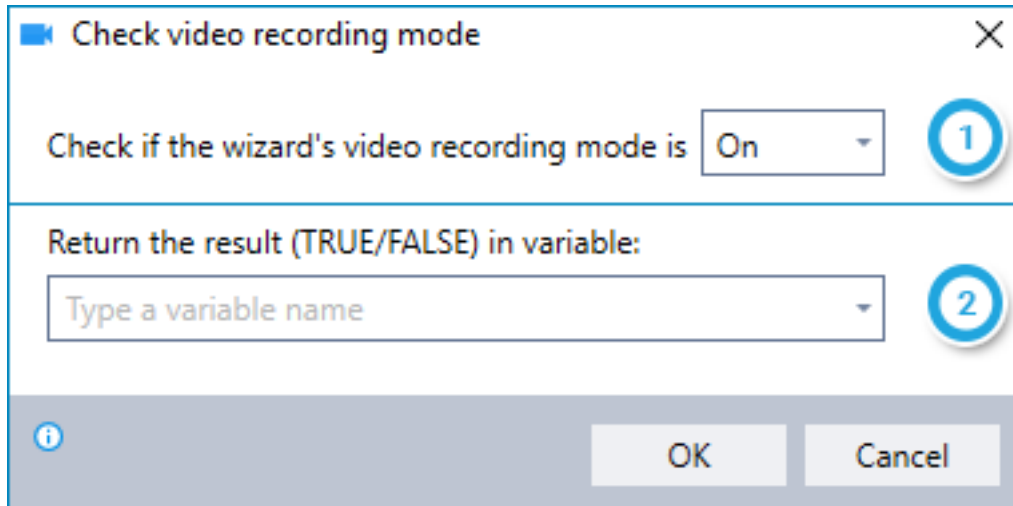
2

- 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

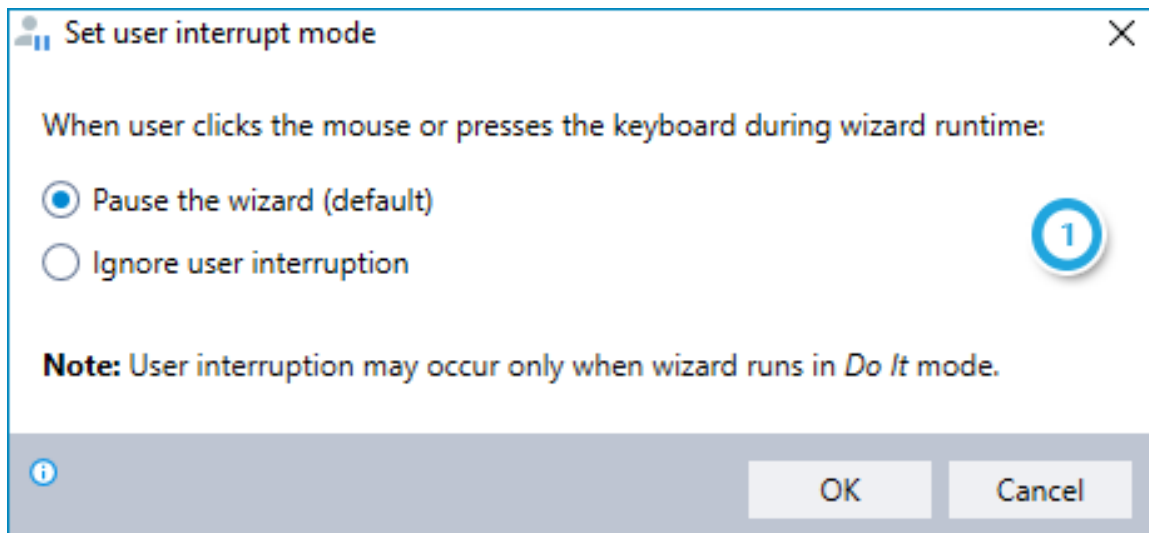
Select whether or not to pause a wizard when the end user clicks the mouse or uses the keyboard while it's running.



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



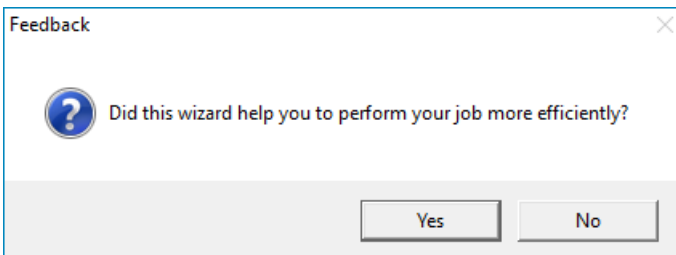
1

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

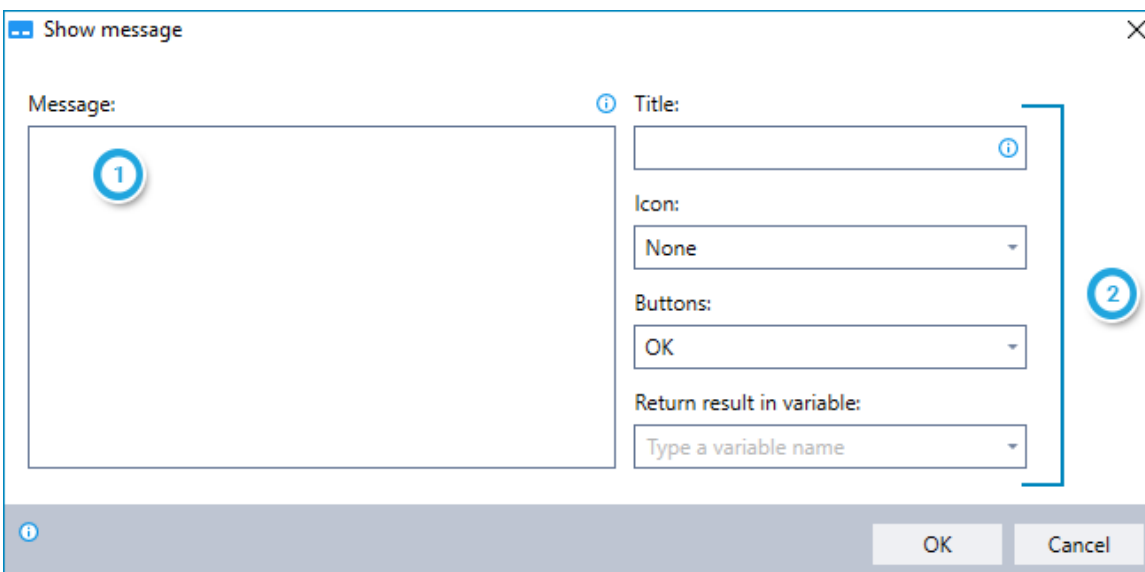
- Pause the wizard (allow user interruption); or
- Continue playing the wizard (ignore user interruption)

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?

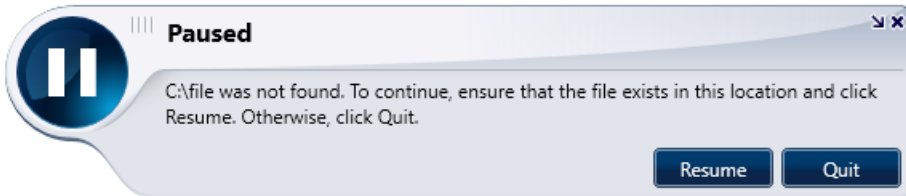
While placing the result of **SHOW MESSAGE** into a variable is optional, there are times when it can be particularly useful:

- 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.

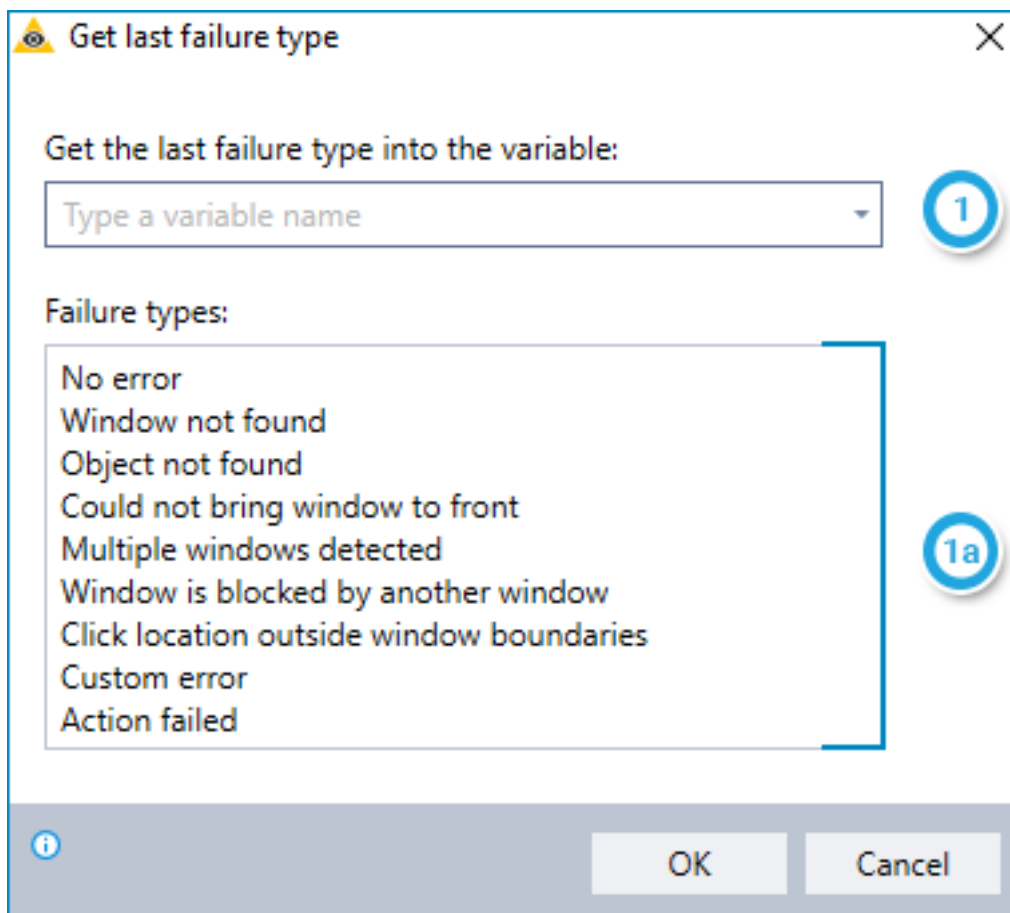
Recommended: The message should instruct the user how to correct the error before he clicks **RESUME**.

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

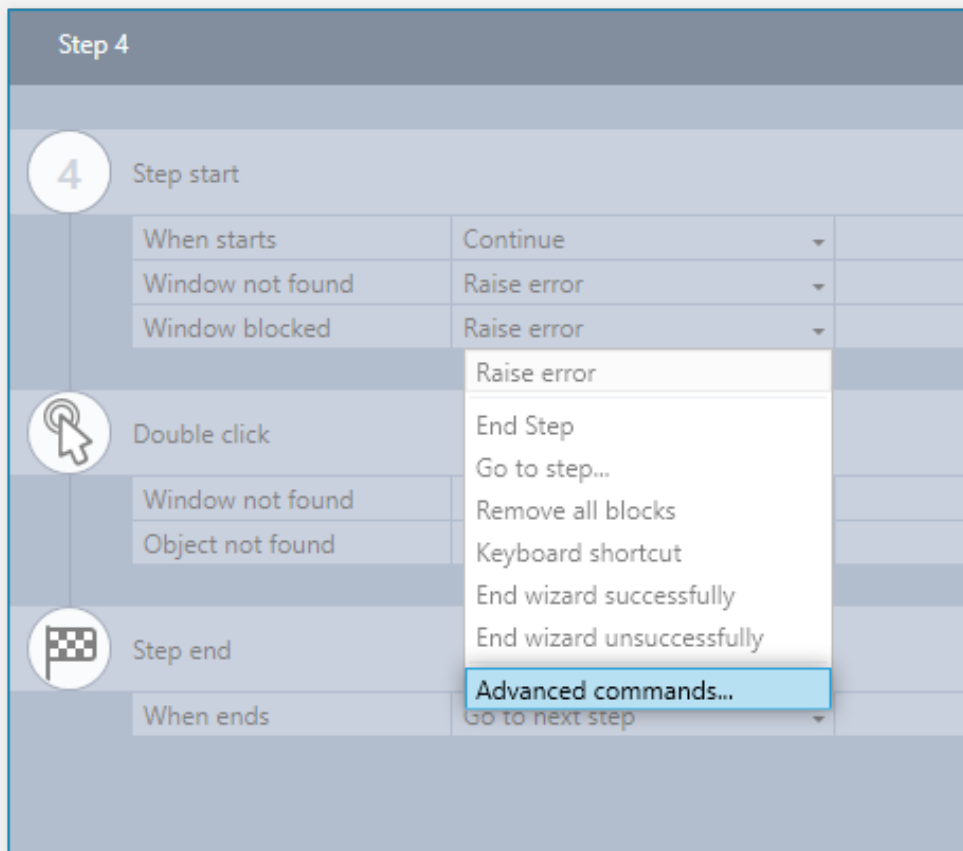
Designed to work in conjunction with Kryon's fallback features (i.e., an action or series of actions you have defined for when an error occurs), this command instructs the wizard to return to the default procedures for handling an error after performing the fallback procedures you have defined. For more 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	112
Input Text	113
Get Mouse Position	114
Drag & Drop	115
Mouse Click	117
Wait for Busy Cursor	118
Set Caps Lock State	119
Get Caps Lock State	120

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

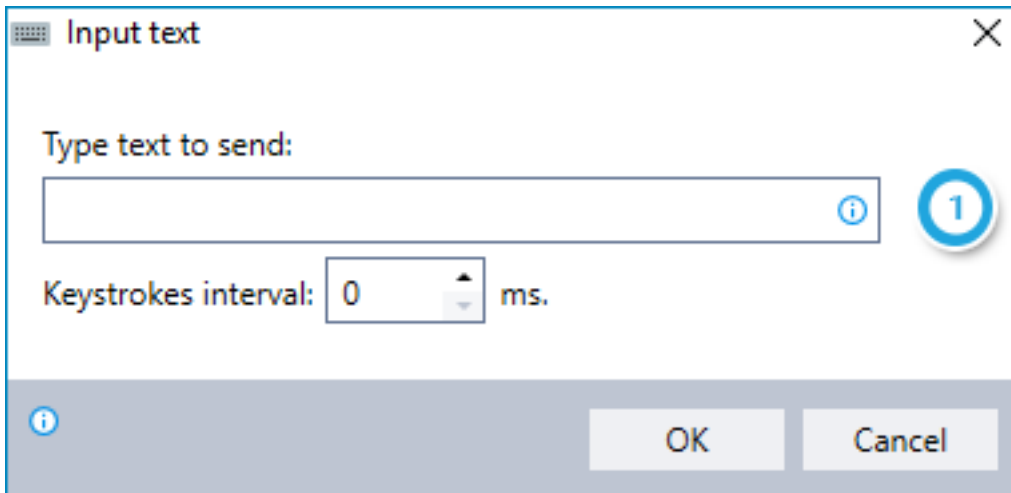
Send the text you've entered (or the text stored in a variable) to the active application.



NOTE

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

Using the INPUT TEXT command



1

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

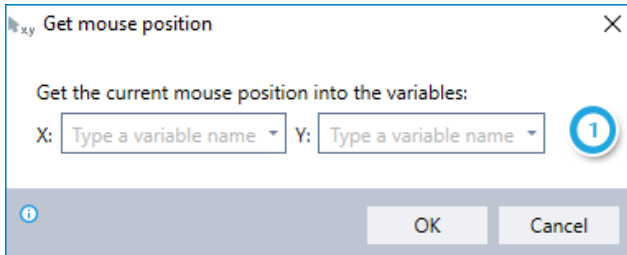
- The longer the interval selected, the slower the keystrokes are sent
- Applications/configurations can vary widely in how quickly they are able to accept data, so finding the best interval may require a bit of experimentation

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

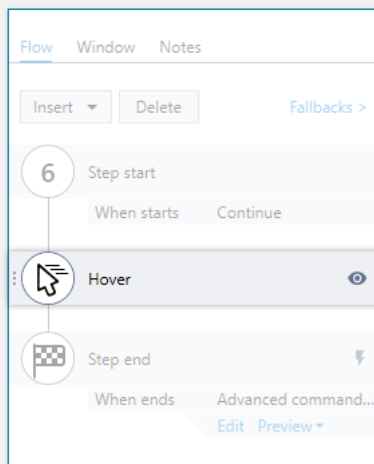


TIP

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.



NOTE

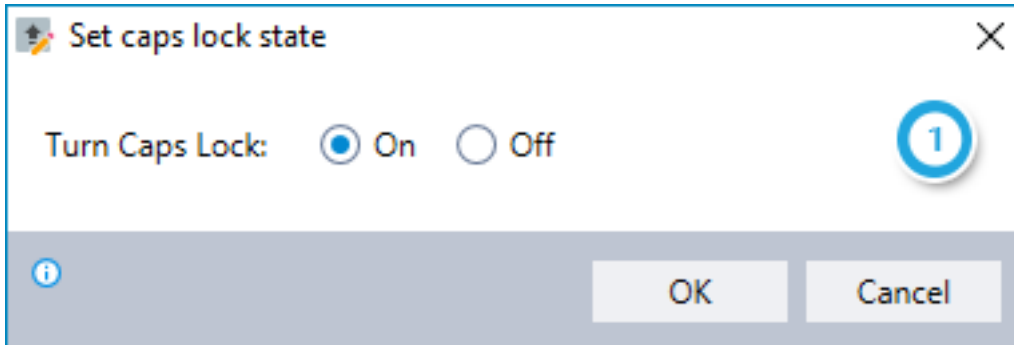
The mouse Click action has a default timeout of four clicks. This means that, if after four clicking attempts, the click action doesn't process (the click isn't executed for some reason), the action is terminated and an error is raised. See Appendix D: Click Blocked

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



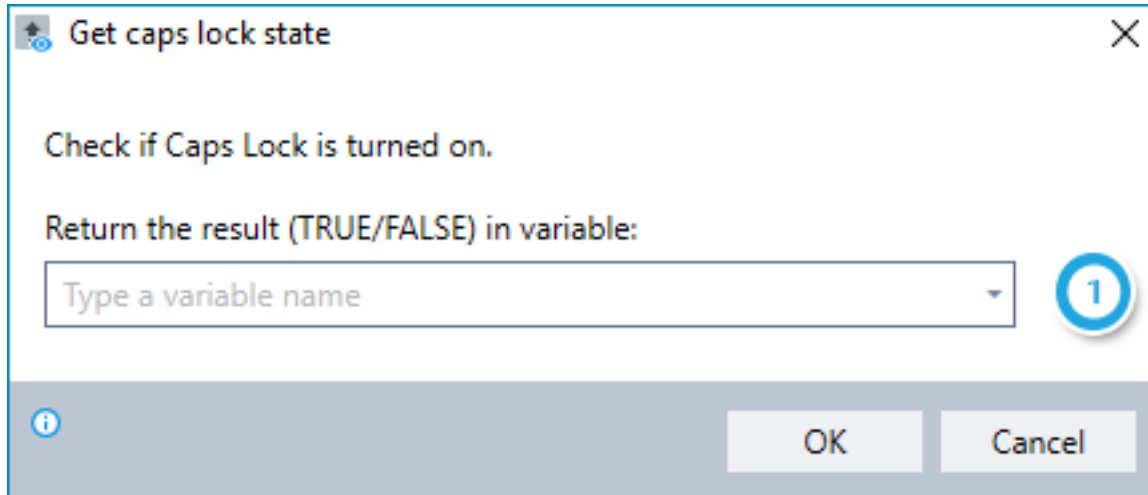
- 1 Choose whether to set **Caps Lock** to **On** or **Off**

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.

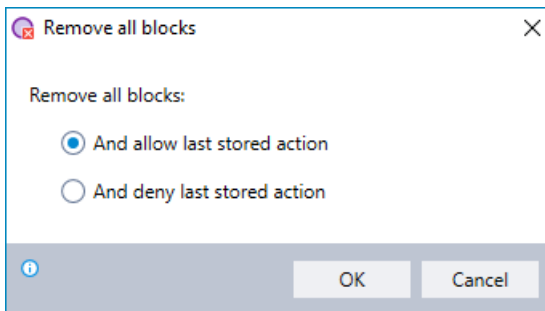
Deny Last Stored Action

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	125
Validate Date	126
Get Day of Month	127
Compare Dates	129
Add/Subtract Date	131
Calculate Date Range	133
Check Day of Week	135
Format Date	137
Get Current Time	140
Compare Time	141
Add/Subtract Time	143
Calculate Time Range	144
Convert Between Time Zones	146



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)

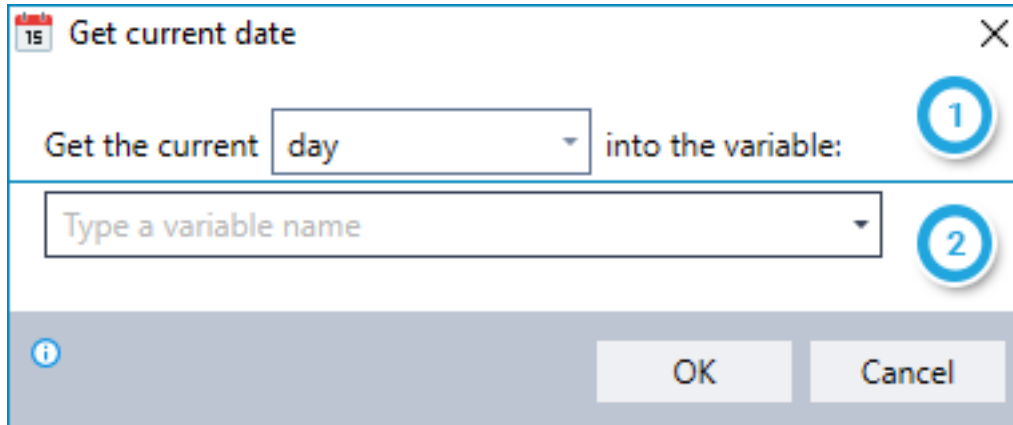
Time

- **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" with a close button in the top right corner. The main content area contains the text "Get the current" followed by a dropdown menu currently set to "day", and "into the variable:". Below this is a text input field with the placeholder text "Type a variable name". At the bottom of the dialog are "OK" and "Cancel" buttons. Two numbered callouts (1 and 2) are present: callout 1 points to the dropdown menu, and callout 2 points 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:

- Section 1:** "Get the First day of:" followed by two dropdown menus labeled "Month:" and "Year:", both containing the placeholder text "Type a variable name". A blue circle with the number "1" is positioned to the right of these fields.
- Section 2:** A checkbox labeled "Count working days only:" followed by seven checkboxes 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.
- Section 3:** "Return the result in variable:" followed by a dropdown menu containing the placeholder text "Type a variable name". A blue circle with the number "3" is positioned to the right of this field.
- Section 4:** A dropdown menu labeled "Error handling" with an information icon (i) to its right. A blue circle with the number "4" 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 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

The screenshot shows the 'Add/subtract date' dialog box with the following fields and callouts:

- 1** Input date type: Use current day
- 2** Input date: Day/s, Month/s, Year/s (each with a text input field containing 'Enter value or \$variable\$') and Output date: Day, Month, Year (each with a dropdown menu containing 'Select a variable')
- 3** Input region: Use a different regional format than the operating system (checkbox) and Output region: Use a different regional format than the operating system (checkbox)
- 4** Error handling: Error variable (dropdown menu containing 'Select a variable') and General error value (text input field containing 'ERROR')

Buttons: OK, Cancel

1. Select the Input date type (Current date, use variables for date, or Custom format).
2. Enter the number of days/months/years you would like to add to or subtract from the existing date.
Note: 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. Select the date regional format to be used for the input and output (when applicable).
4. 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:	Type a variable name	Day:	Type a variable name
	Variable value: 1 - number of days in month		
Month:	Type a variable name	Month:	Type a variable name
	Variable value: 1 - 12		
Year:	Type a variable name	Year:	Type a variable name
	Variable value: 1 - 9999		

Count working days only

Mon Tue Wed Thu Fri Sat Sur

Return the result in variable:

Type a variable name

▼ 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

1. Select the input type for the date to be formatted and provide additional information as required (varies by input type)
 - **Current date:** The wizard will format the current date (according to the system clock of the machine on which the robot is running)
No additional information is required
 - **Use day/month/year variables:**
Enter the names of the variables that contain the day, month, and year of the date to be formatted

- **Custom format:**

Enter the date the wizard should format

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)

In other words, whether **6/5/2020** 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. Select the format to which the specified date should be converted.
3. Select the date regional format to be used for the input and output (when applicable).
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

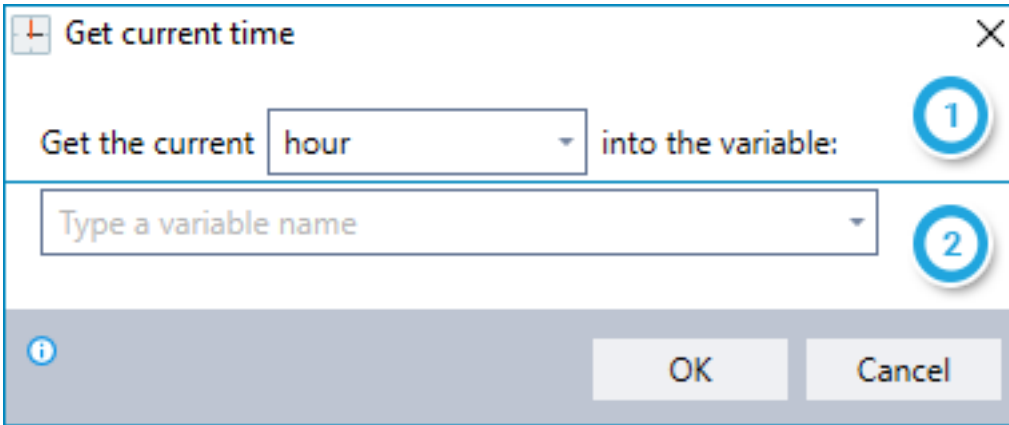
MMMM dd, yyyy → June 05, 2017

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

- 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	148
Find Matching Window Handles	149
Get Active Application	151
Control Window State	152
Check Window State	154
Get Active Window/Web Page	155
Capture Step Window Image	156

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

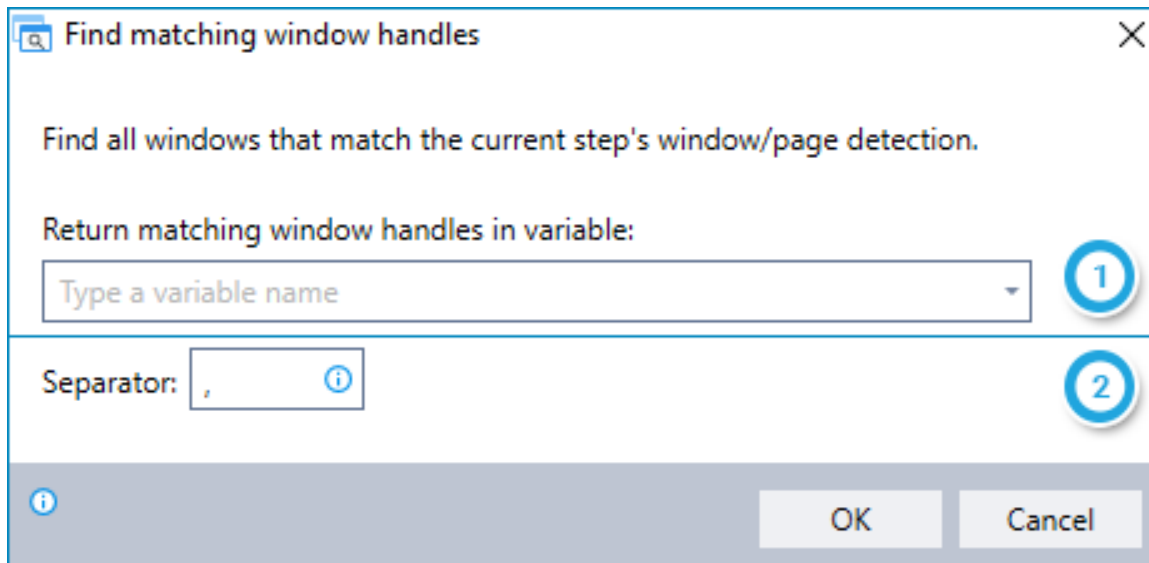
- 1 Enter the name of the variable into which you'd like to place the handle
- 2 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

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



- 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 for a wizard step. At the top, there are tabs for 'Flow', 'Window', and 'Notes', with 'Window' selected. Below the tabs are 'Insert' and 'Delete' buttons. The main area is titled 'Window data' and contains two checked items: 'Class name' with the operator 'Equals' and the value 'XLMAIN', and 'Caption' with the operator 'Contains' and the value 'Excel'. Below this is an 'Options' section with a checkbox for 'Wait for window to appear (1 sec.)' which is unchecked. There is a 'Bring window to front' dropdown menu set to 'Automatically'. At the bottom, there is a checkbox for 'Custom window name' which is unchecked, followed by an empty text input field.

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

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

Get Active Application

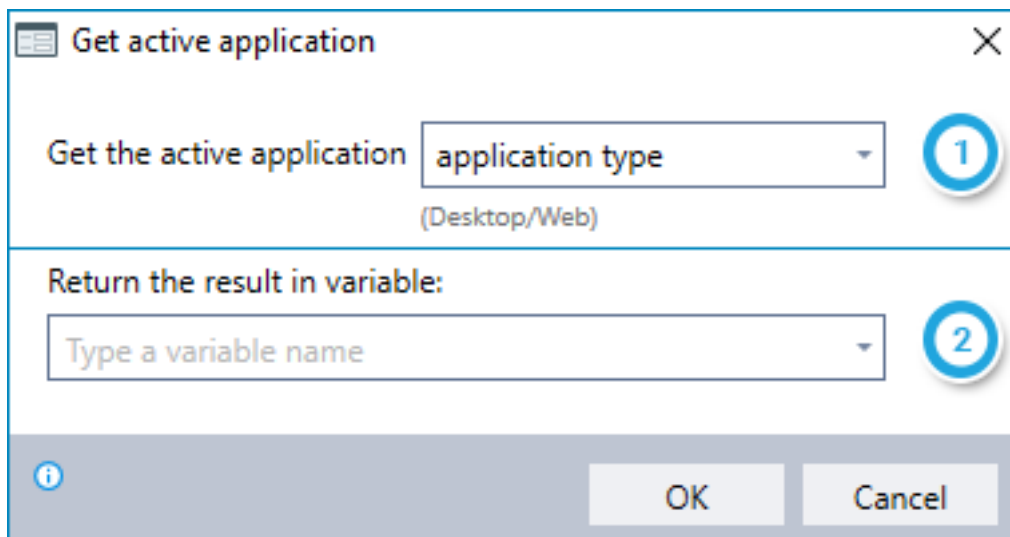
Retrieve information about the application on which the current wizard is being run and place it into a new or existing variable.

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

Control window state

Close the window specified by handle: Select a variable

^ Error handling ⓘ

Error variable: Select a variable

Enable timeout monitoring

Raise error after: 1800 seconds

Timeout error text: TIMEOUT_ERROR

OK Cancel

Select the action you want to take on the relevant window (close, minimize, maximize, bring to front, etc.), and enter the name of the variable in which the handle of the window is stored.

- **Error variable:** Enter a designated error variable to get an indication if the command fails.
- **Enable timeout monitoring (optional):** Enable timeout monitoring and set the seconds and timeout error text.

See [Timeout Monitoring](#) Support.



TIP

Why enable timeout monitoring for the “Control window state” command?

For example, if the target window is blocked by a dialog, the Wizard might keep attempting to execute the command ‘endlessly’ unless Timeout Monitoring is enabled.

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

The screenshot shows a dialog box titled "Check window state". It has a title bar with an information icon, the text "Check window state", and a close button. The main area contains two rows of input fields. The first row is labeled "Check if the window specified by handle:" and contains a dropdown menu with "Type a variable name" and another dropdown menu with "Exists". A blue circle with the number "1" is next to this row. The second row is labeled "Return the result (TRUE/FALSE) in variable:" and contains a dropdown menu with "Type a variable name". A blue circle with the number "2" is next to this row. At the bottom right, there are "OK" and "Cancel" buttons. A blue circle with the number "1" is also next to the "OK" button.

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

Get Active Window/Web Page

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

You can choose to obtain the following types of information:

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

Using the Get ACTIVE WINDOW/WEB PAGE command

Get active window/web page

Get the active into the variable: 1

2

OK Cancel

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

Capture Step Window Image

Take a screen capture of the window on which the 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	158
Read From Text File	160
Write to Text File	161
Does File Exist	162
Copy a File	163
Move a File	164
Rename a File	165
Delete a File	166
Delete File(s)	168
Monitor File Changes	170

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 existing 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.

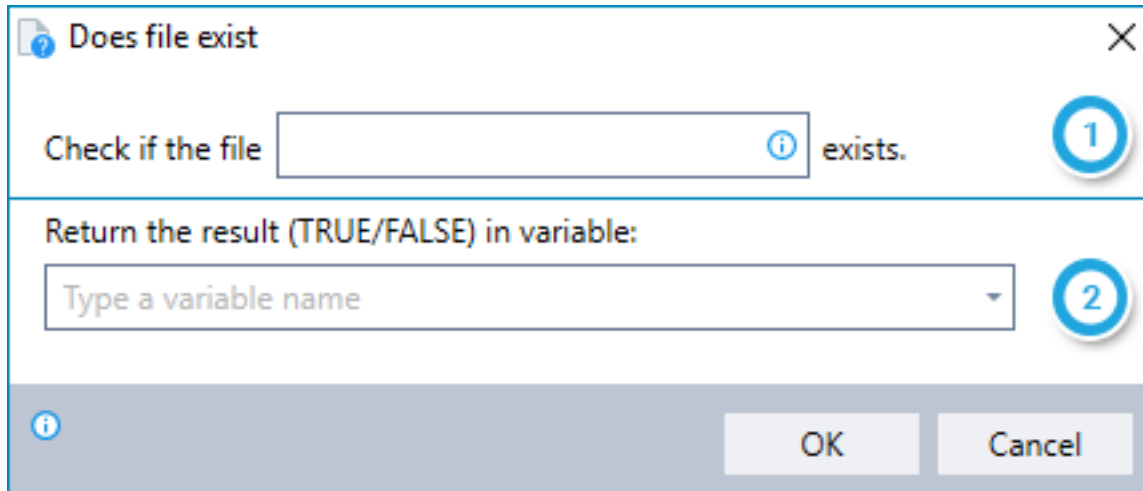
Using the WRITE TO TEXT FILE command

- 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



Does file exist

Check if the file exists. 1

Return the result (TRUE/FALSE) in variable:

2

OK Cancel

- 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 'Copy a file' dialog box with the following elements:

- File to copy:** A text input field with an information icon (i) to its right, marked with a circled '1'.
- Target folder:** A text input field with an information icon (i) to its right, marked with a circled '2'.
- Override already existing files in target folder:** A checkbox, marked with a circled '3'.
- Error handling:** A dropdown menu with a downward arrow and an information icon (i) to its right, marked with a circled '4'.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right, and an information icon (i) at the bottom left.

- 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 the 'Move a file' dialog box with the following fields and callouts:

- 1** File to move: Enter full path or variable
- 2** Target folder: Enter full path or variable
- 3** Return moved file path in variable: Select a variable
- 4** Error handling:
 - Error variable: Select a variable
 - General error: GeneralError
 - File not found: FileNotFound
 - Access denied: AccessDenied

Buttons: OK, Cancel

1. Enter the **full file path and file name** of the file to move.
Example: `C:\LeoSource\Text.txt`
2. Enter the **full path** of the folder to which to move the file.
Example: `C:\LeoDest`
NOTE: 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. **Return moved file path in variable:** (optional): enter a new file path (move to target folder file) to store a new full path of the moved file in a variable for further usage.
Example: `C:\LeoDest\Text.txt`
4. 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

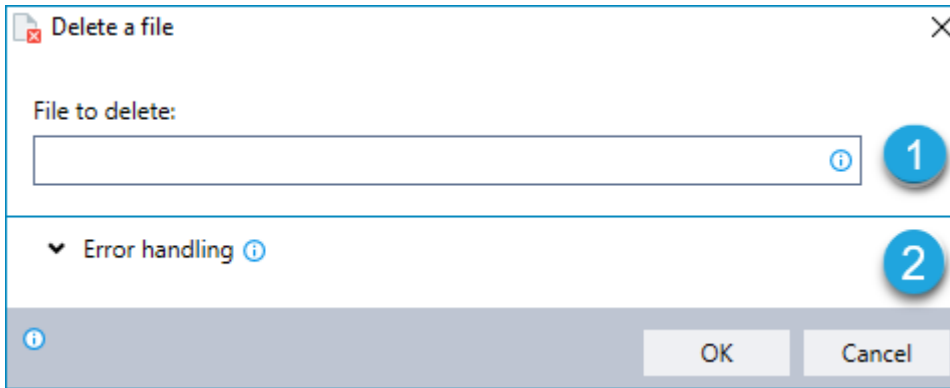
- 1 Enter the **full file path and file name** of the file to rename
- 2 Enter the new name for the file, including the file extension
 - If you enter only the file name (with no path), the file will be renamed and remain in its current location
 - If you enter a new file path along with the file name,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



NOTE

Using the "Delete file", "Delete files", and "Delete folder" commands by at end of a running wizard

When using the "delete file(s)/folder" commands at the end of a running wizard, the Windows OS might prevent the delete-action due to the files being in-use/open. To delete the file/folder after the wizard ends, you have two (2) options:

Option 1: Manually delete the desired file/folder.

Option 2: End the wizard and start a new wizard with the "delete" command:

1. End the wizard to let it automatically close all the open and in-use processes/files/folders.
2. Create a new continuing wizard
3. Add the relevant "Delete" command at the beginning of the continuing wizard and configure it to delete the desired file(s)/folder.
4. End the wizard or use it to execute additional steps.

Delete File(s)

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

Using the DELETE FILE(S) command

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

- 1** Specify a folder name to delete from: [Input field]
- Make sure the folder is accessible for all end users.
- Include sub-folders
- 2** File to delete: [Input field]
- Enter file name or pattern (use stars: *). Example: *.xlsx
- 3** Return the number of deleted files in variable (optional): [Dropdown menu]
- Return the number of files that failed to delete in variable (optional): [Dropdown menu]
- 4** Error handling [Dropdown arrow]
- Buttons: OK, Cancel

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.



NOTE

Using the "Delete file", "Delete files", and "Delete folder" commands by at end of a running wizard

When using the "delete file(s)/folder" commands at the end of a running wizard, the Windows OS might prevent the delete-action due to the files being in-use/open. To delete the file/folder after the wizard ends, you have two (2) options:

Option 1: Manually delete the desired file/folder.

Option 2: End the wizard and start a new wizard with the "delete" command:

1. End the wizard to let it automatically close all the open and in-use processes/files/folders.
2. Create a new continuing wizard
3. Add the relevant "Delete" command at the beginning of the continuing wizard and configure it to delete the desired file(s)/folder.
4. End the wizard or use it to execute additional steps.

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 the following fields and options:

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

At the bottom, there are "OK" and "Cancel" buttons.

- 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	173
Get Folder Location	174
Get Files	175
Does Folder Exist	177
Is Folder Empty	178
Copy a Folder	179
Move a Folder	180
Rename a Folder	181
Delete a Folder	182
Monitor Folder Changes	185

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 'Get files' dialog box with the following fields and options:

- 1** Root folder: [Text input field]
- 2** File name to retrieve (optional): [Text input field]
- 3** Include sub-folders
- 4** Include folder name in output
- 5** Sort by: [File name] (dropdown), Ascending, Descending
- 6** Return results (file names) in variable: [Type a variable name] (dropdown)
- 7** Result delimiter: [Text input field]

Buttons: OK, Cancel

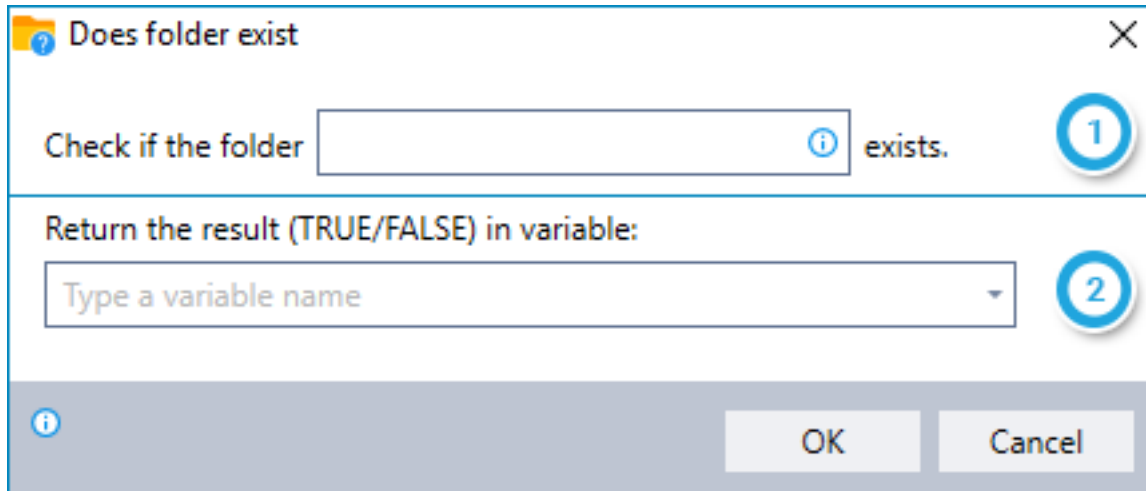
- 1** Enter the **full path** of the root folder from which to retrieve a list of files
- 2** **To retrieve a list of all files:** Leave this field blank;
To retrieve a single file: Enter the name of the file; *or*
To retrieve a list of matching files: Enter a naming pattern for the files, using asterisks (*) 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

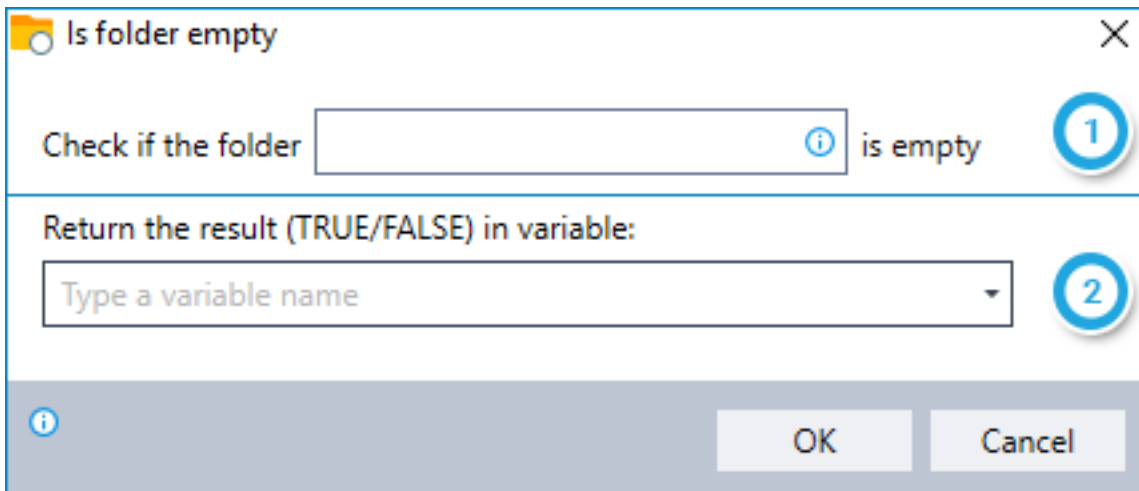


- 1 Enter the **full path** of the folder for which to check
- 2 Enter the name of the variable into which to place the result of the check. (The result will be either TRUE or FALSE, as applicable.)

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". The dialog has a title bar with a close button (X) 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, then "is empty". A blue circular callout with the number "1" points to the text input field. 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 circular callout with the number "2" points to the dropdown menu. At the bottom of the dialog, there is a grey bar containing an information icon (i) 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

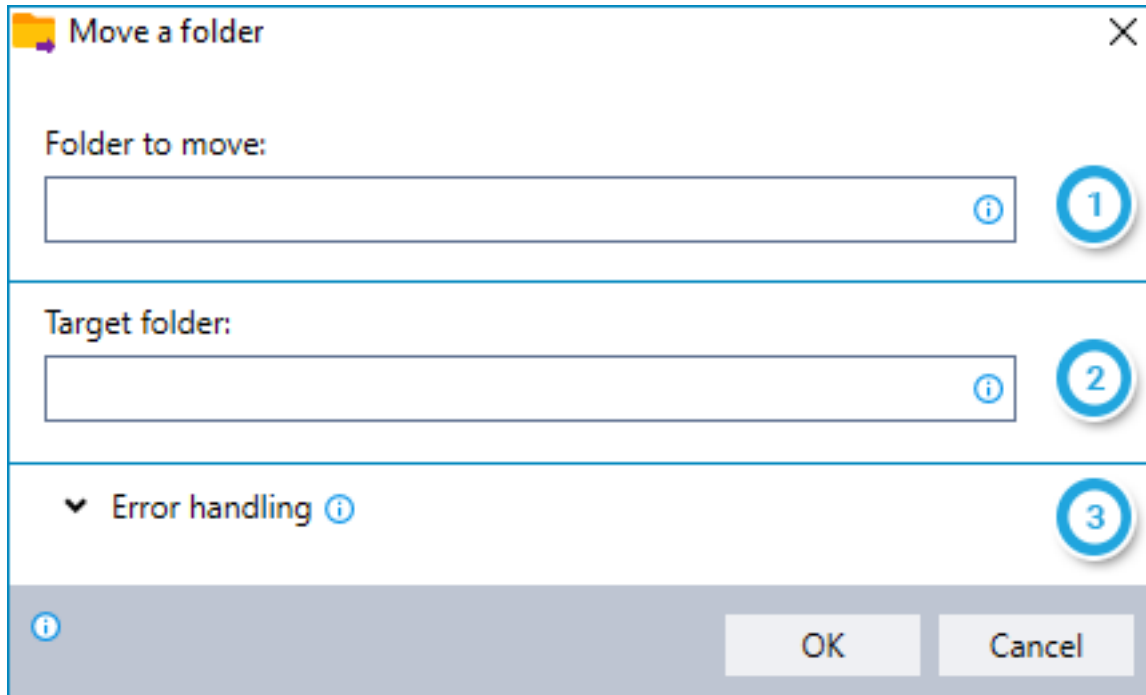
The screenshot shows a dialog box titled "Copy a folder" with a close button in the top right corner. The dialog is divided into three sections. The first section, "Folder to copy:", has an empty text input field with an information icon to its right and a callout bubble with the number "1". The second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

- 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



- 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.

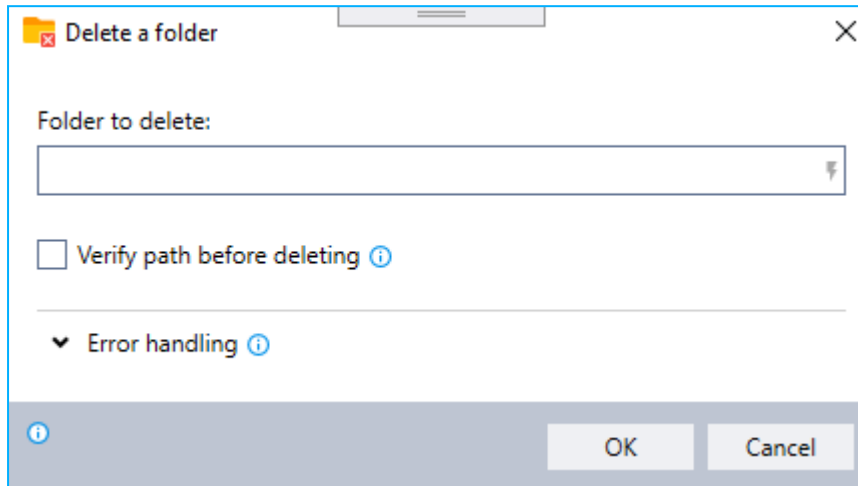
Using the RENAME A FOLDER command

- 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.
Note: Check "**Verify path before deleting**" to make sure folder delete isn't executed in case one of the variables in the path is empty.
2. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).



NOTE

Using the "Delete file", "Delete files", and "Delete folder" commands by at end of a running wizard

When using the "delete file(s)/folder" commands at the end of a running wizard, the Windows OS might prevent the delete-action due to the files being in-use/open. To delete the file/folder after the wizard ends, you have two (2) options:

Option 1: Manually delete the desired file/folder.

Option 2: End the wizard and start a new wizard with the "delete" command:

1. End the wizard to let it automatically close all the open and in-use processes/files/folders.
2. Create a new continuing wizard
3. Add the relevant "Delete" command at the beginning of the continuing wizard and configure it to delete the desired file(s)/folder.
4. End the wizard or use it to execute additional steps.



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*"
- 3:** Two checkboxes: "Folder created (or renamed)" and "Folder deleted".
- 4:** "Return folder path in variable:" followed by a dropdown menu with the text "Type a variable name".
- 5:** "Timeout: 5 minutes" (the number 5 is in a spinner box).
- 6:** A collapsed section "Error handling" with 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?	188
Azure Licenses	190
OCR: Documents	191
Get Text (SmartScan)	195
Get Value	196
Get Selection Element State	198
Does Word Exist (SmartScan)	200
Get Date/Time	201
Save as Image	203
Save to Excel	204
OCR: Printed and Handwritten Text	206
Get Text (MS Azure)	209
Does Word Exist (MS Azure)	210
Form Recognizer	211
Get Receipt Data	214
Text Analytics: Analyze Sentiment	216
Text Analytics: Detect Language	217
Text Analytics: Identify Key Phrases	218
Convert to Text (SmartScan+)	219
Convert to Text (Tesseract)	220

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

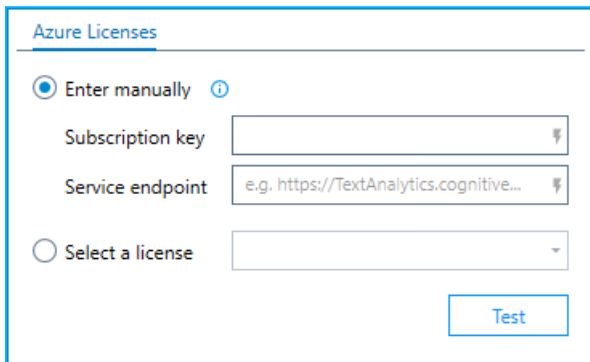
The following Advanced Command offers OCR capabilities as part of the standard Kryon installation and does not require an additional license:

- 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.



Azure Licenses

Enter manually ⓘ

Subscription key

Service endpoint

Select a license

Test

- 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: 1

2

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.

3

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

4

▼ 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 of the 'Document Separation' wizard. It features three radio button options: 'Analyze all pages' (selected), 'Analyze specific pages:' (with a text input field and a help icon), and 'Ignore pages with any of the following words:' (with a text input field and a help icon). Below the 'Analyze specific pages:' field, it says 'Comma separated. Example: 1, 5, 7-9, 13-*'. Below the 'Ignore pages...' field, it says 'Comma separated.'.

Document Separation:

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

The screenshot shows the 'Document Separation' tab of the wizard. It has three radio button options: 'Do Not Separate' (with a stack of papers icon and description 'Analyze the entire file as a single document'), 'By Page' (with three separate page icons and description 'Analyze each page as a separate document'), and 'By Inner Document' (with a grid of page icons and description 'Split each page to inner documents and analyze each document separately'). The 'By Inner Document' option is selected.

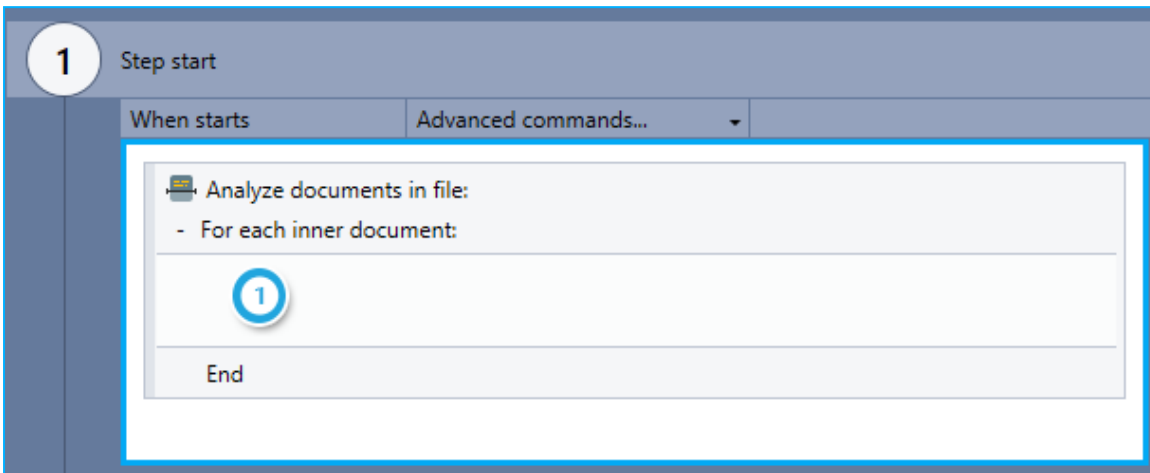
3 Indicate if you wish to place information about total pages/documents and current page/document into variables

- **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

Get text (SmartScan)

Set document text into a variable:

Type a variable name

Preserve formatting

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

OK Cancel

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

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 fields and options:

- 1** Search for the word: [Text input field]
- 2** And read value from: A central diagram with four arrows pointing up, down, left, and right. Each arrow has a radio button next to it. The 'Right side' radio button is selected. The labels are 'Above', 'Below', 'Left side', and 'Right side'.
- 3** Read entire line
- 3** Allow close match ⓘ
- 3** Required accuracy [Slider] 0%
- 4** Value type: [Dropdown menu showing 'Text']
- 5** Set the value in variable: [Text input field with a lightning bolt icon and placeholder text 'Type a variable name']
- 6** Error handling [Dropdown menu]

At the bottom, there is an information icon ⓘ, and '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

Get selection element state (SmartScan) [X]

Determines if a selection element is selected ⓘ

Selection element is located to the right of the words: ①

Specify all possible words (comma separated).

Allow close match ⓘ ②

Required accuracy [Slider] 100% ③

Add another condition ④

Set the value in variable:

Select a variable ⑤

▼ Error handling

ⓘ [OK] [Cancel]

- 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:" with 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 dropdown arrow (v) on the left.

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

- 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

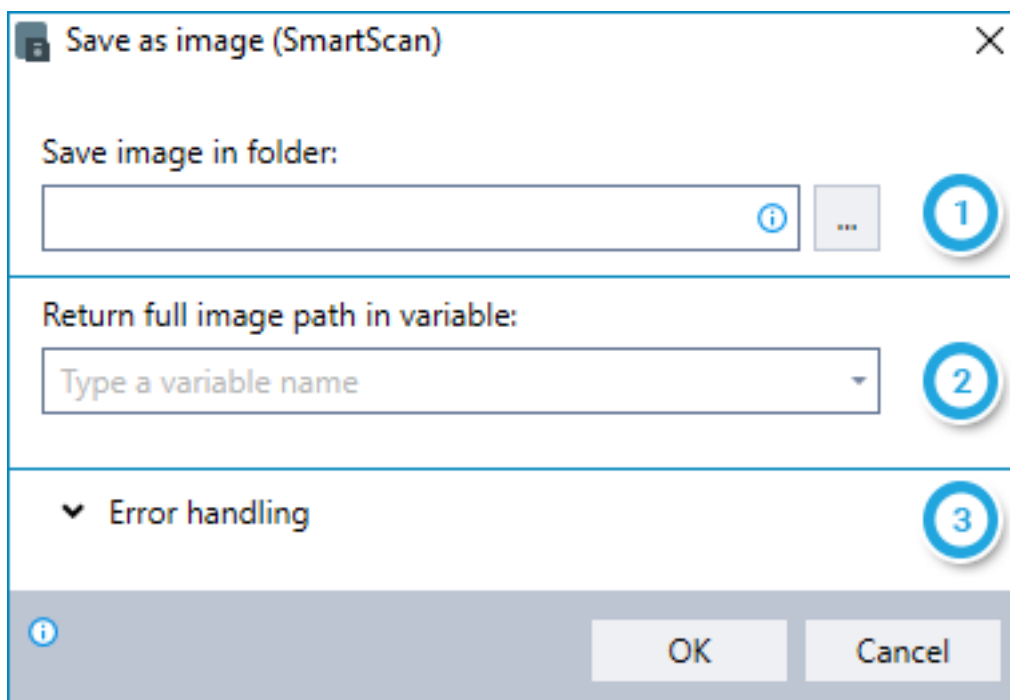
Save a scanned document as an image file.



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

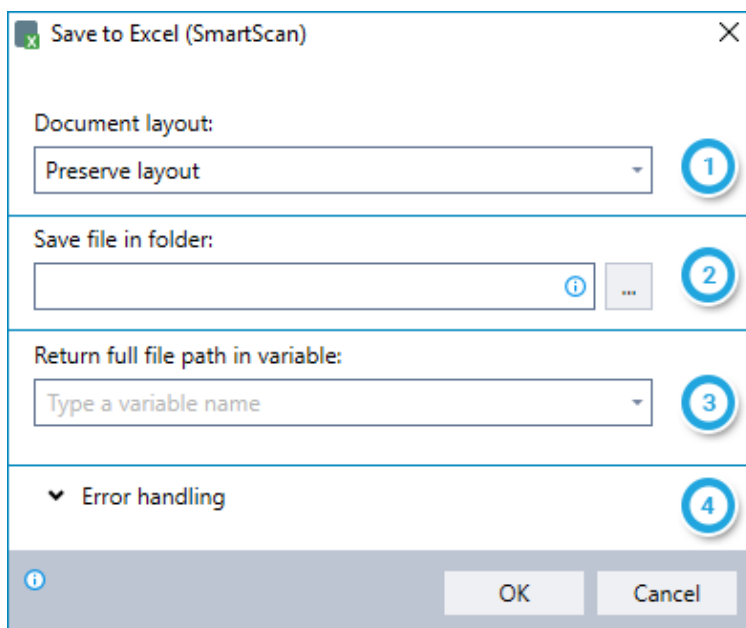
Save text from a scanned document into an Excel spreadsheet.



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

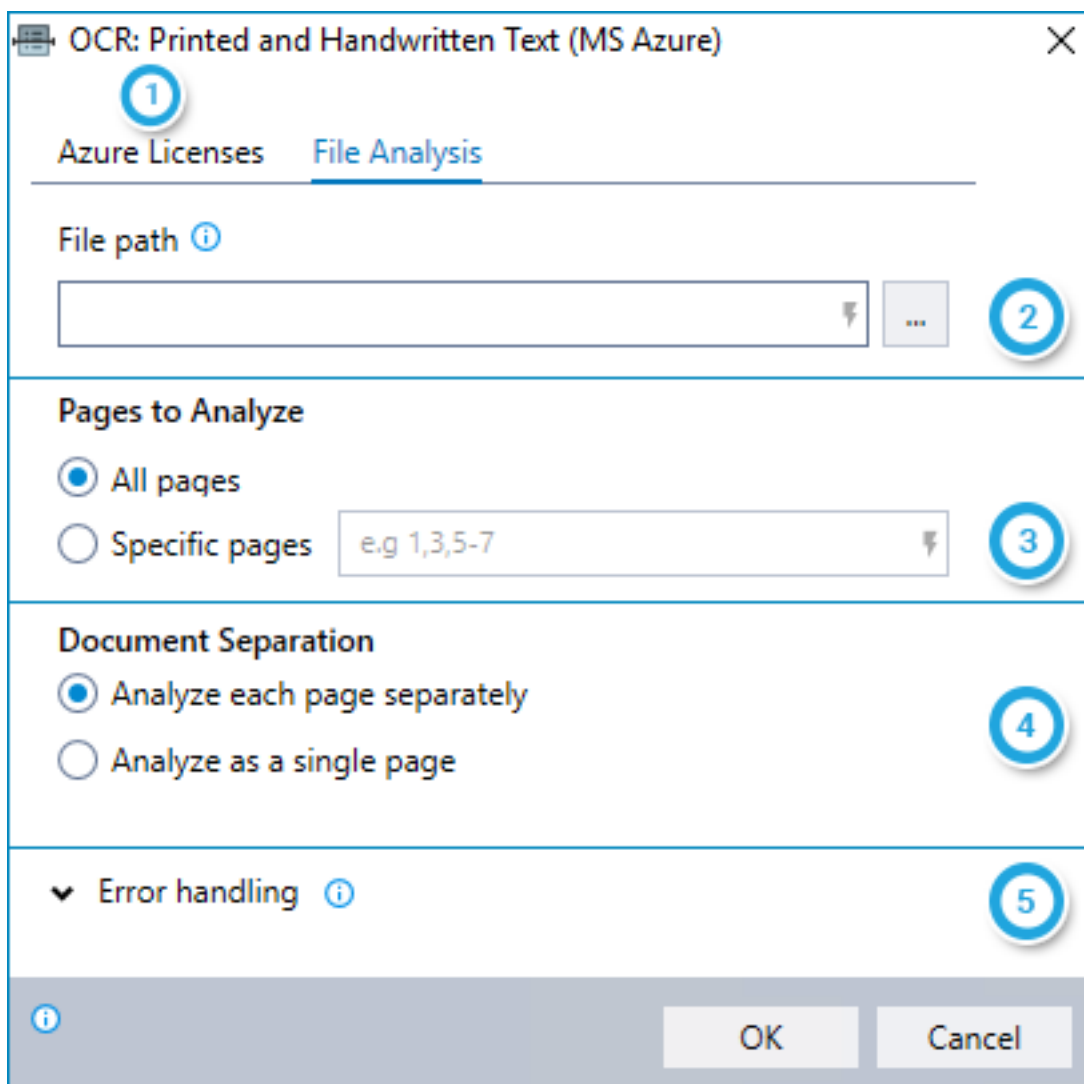
AI Booster

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

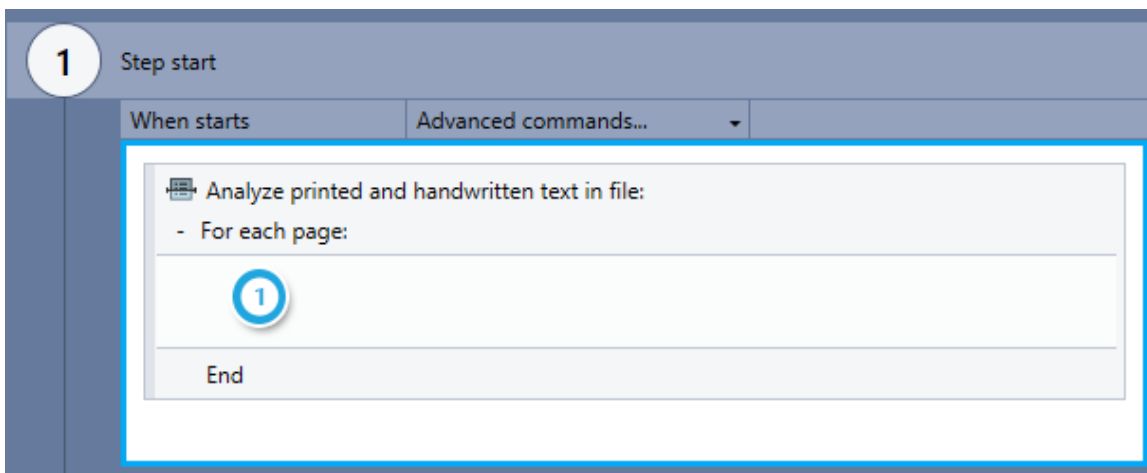
The first step in using the **OCR: PRINTED AND HANDWRITTEN TEXT** command is to analyze and separate the file on which the specified actions will be performed.



- 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

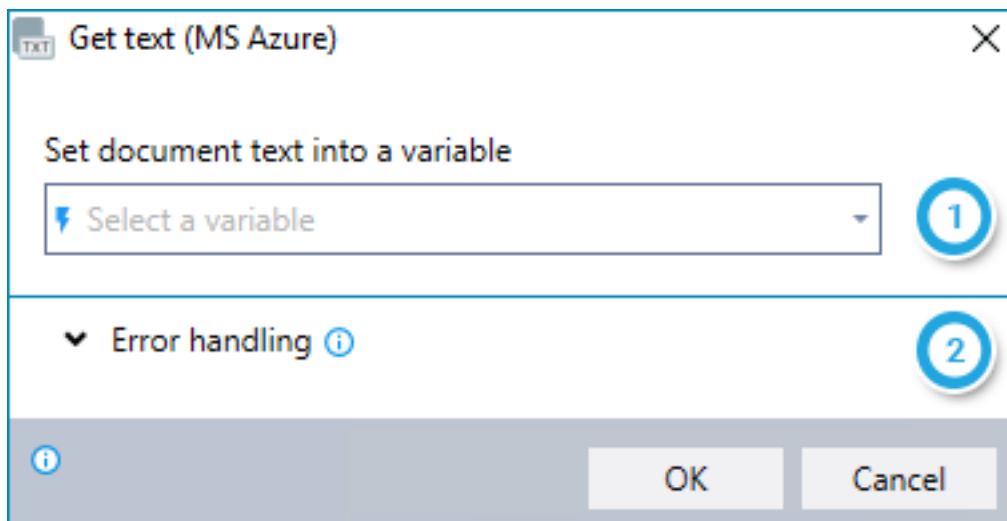
Place text from a printed or handwritten document into a variable.



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

Does word exist (MS Azure)

Search for the text:
Specify all possible terms (comma separated) ⚡ 1

Return result (TRUE/FALSE) in variable:
⚡ Select a variable ▾ 2

▼ Error handling ⓘ 3

ⓘ OK Cancel

- 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

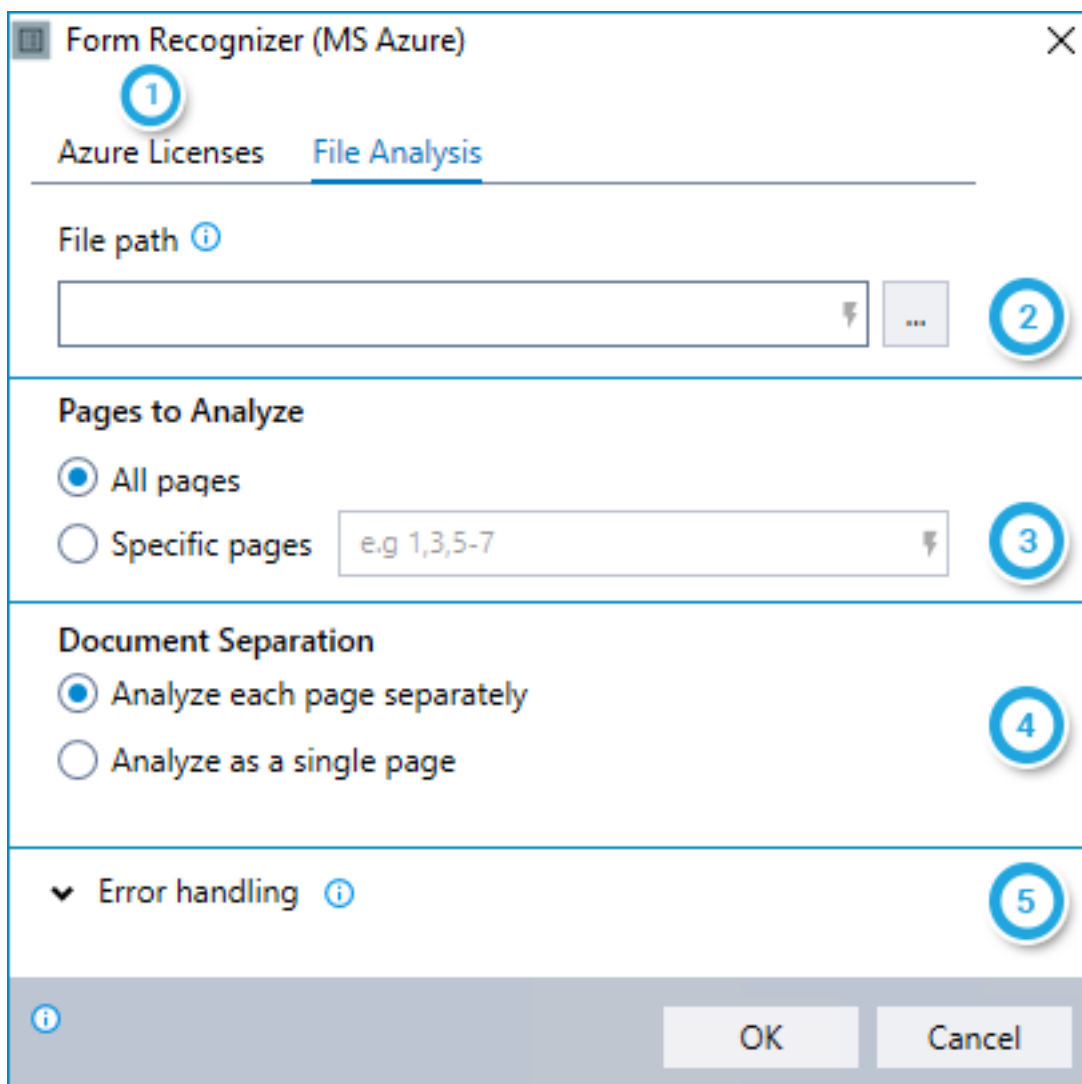
AI Booster

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

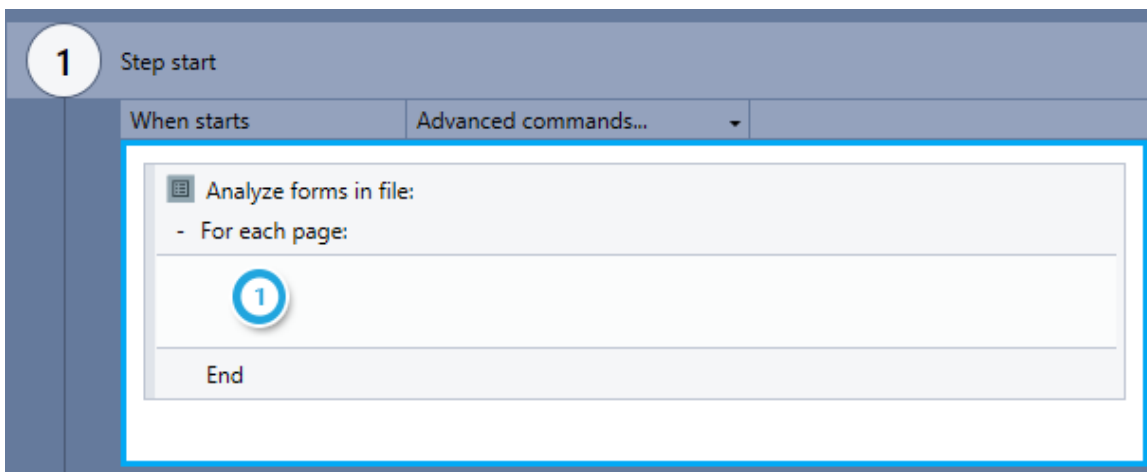
The first step in using the **FORM RECOGNIZER** command is to analyze and separate the file on which the specified actions will be performed.



- 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

- 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

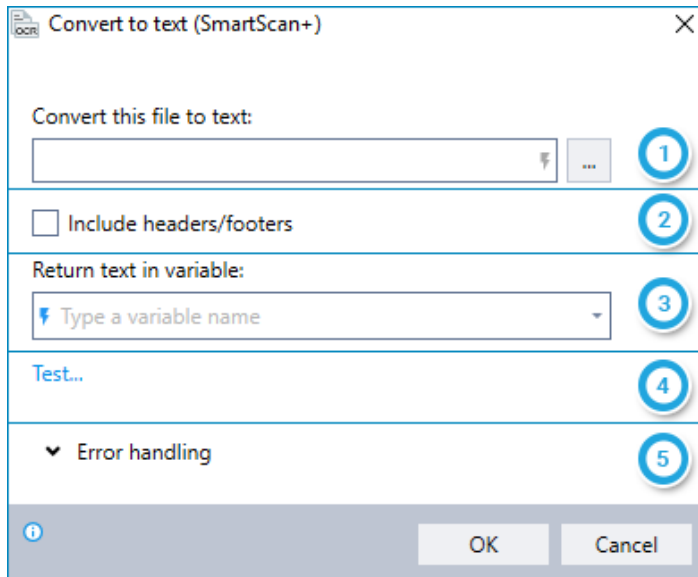
- 1 See [Azure Licenses](#) for the information required to complete this tab
- 2 Enter the text to analyze:
 - Can be free text or a variable
 - Maximum characters: 5,120
 - Supported languages: English, Chinese-Simplified, French, German, Spanish
- 3 Specify the delimiter(s) by which the detected phrases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

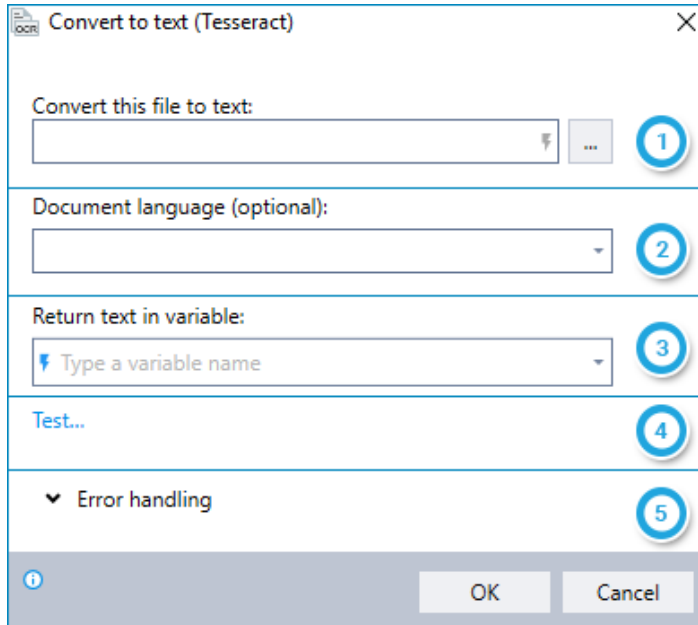


- 1 Select the PDF or image file to convert
- 2 Indicate whether you'd like to include the text of the original document's headers/footers (if any)
- 3 Enter the name of the variable into which you'd like to place the recognized text
- 4 (Optional) Test the text recognition results
- 5 Instruct the wizard 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	222
Page: Get Text	225

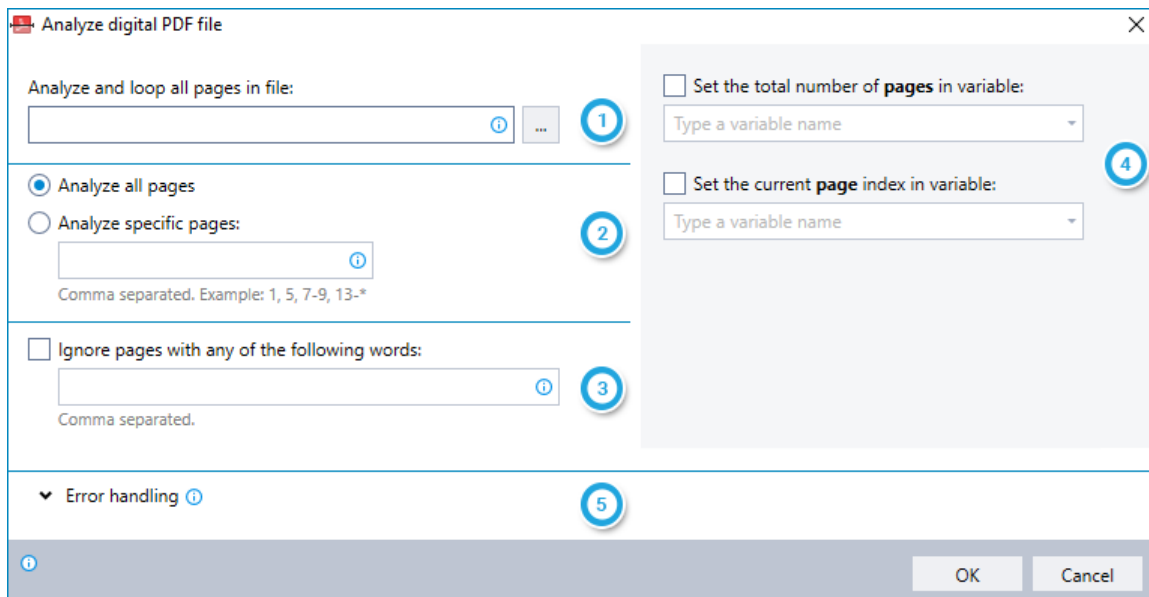
Analyze Digital PDF File

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

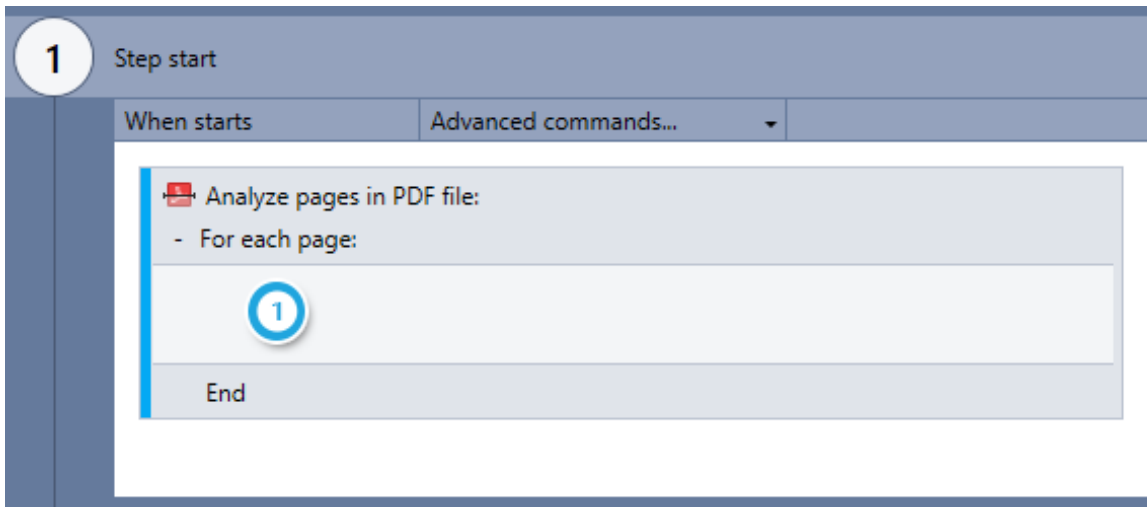
The first step in using the **ANALYZE DIGITAL PDF FILE** command is to analyze and identify the pages on which the specified actions will be performed.



- 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

Page: Get text

Get the current page text into a variable:

Type a variable name

Simple text

Formatted text

OK Cancel

- 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	227
Clear Clipboard	227
Place in Clipboard	229
Copy Active Field Value	230
Get OS Version	231
Get Input Language	232
Read From Registry	233
Remove From Registry	235
Write to Registry	236
Get Environment Variable	237
Get Windows Event Log Data	238
Set BI Field	240
Log an Action	241
Query XML	242
Query JSON	244
Call REST API Method	246

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

Get from clipboard

Get the clipboard data into the variable:

Type a variable name

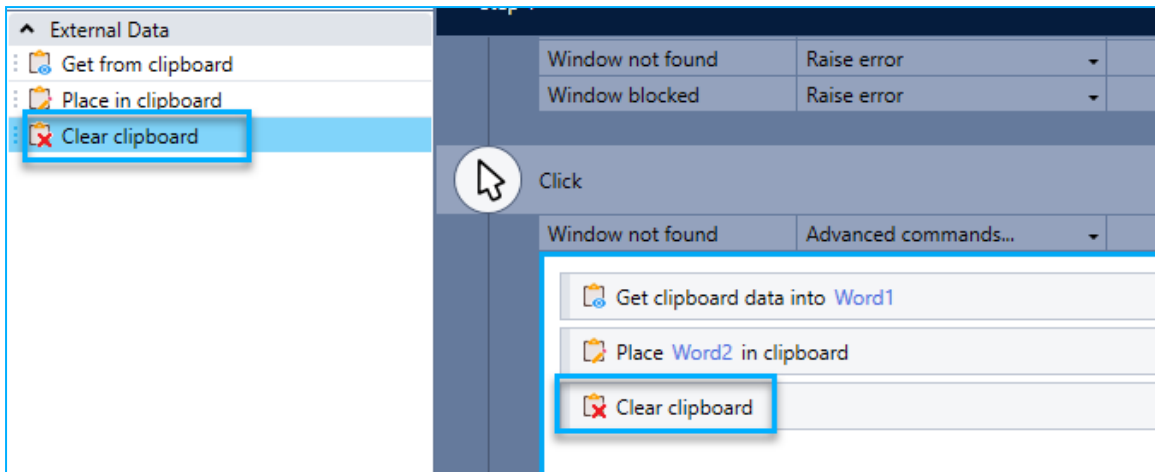
▼ Error handling ⓘ

OK Cancel

- 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](#).

Clear Clipboard

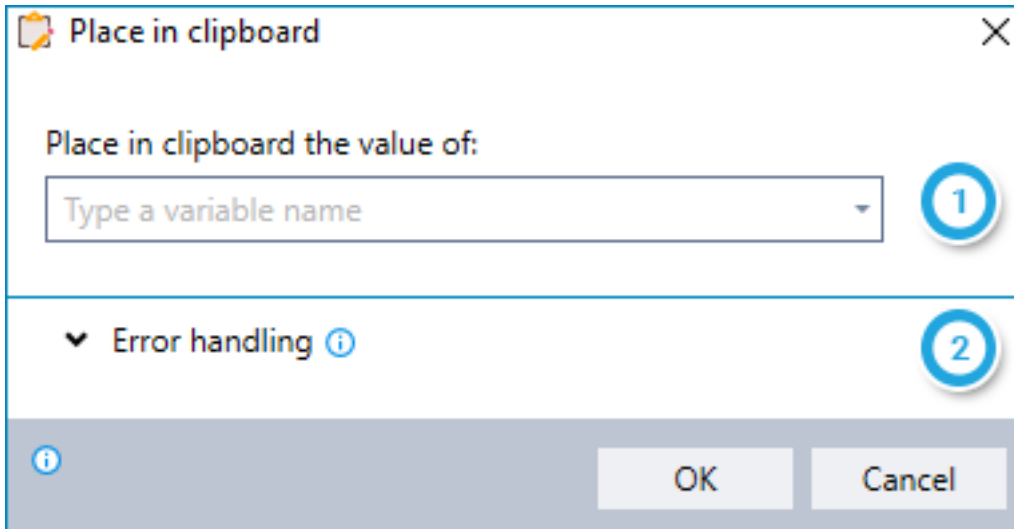
Using the clipboard is essential to copy information. Use the **Clear Clipboard** command to clear the clipboard whenever the information in the clipboard is no longer relevant to the task (or to next step). This way you can ensure the Wizard is using the relevant and updated information from the clipboard. The command can be useful in case the copy command fails. In this case, the Wizard won't paste information that is old or irrelevant to the task since the clipboard has been cleared.



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.

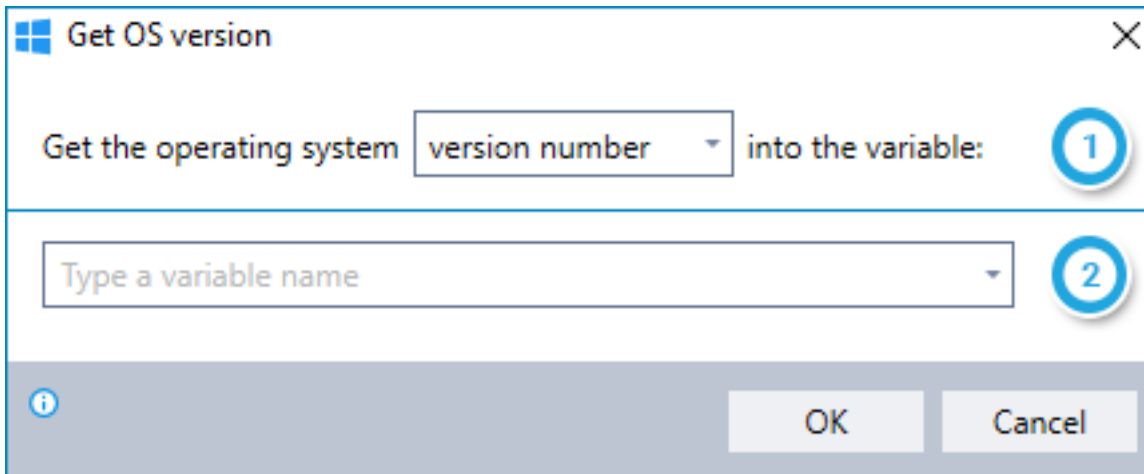
Using the COPY ACTIVE FIELD VALUE command

- 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

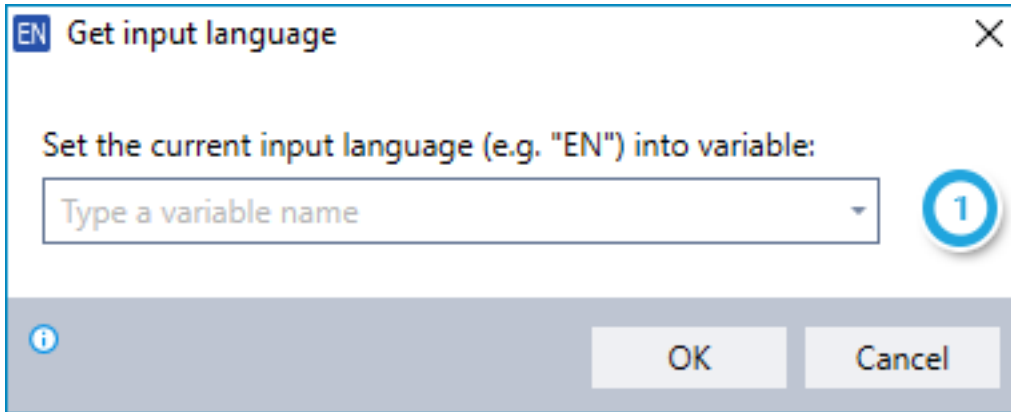


- 1 Choose whether you'd like to retrieve the Windows version number or version name
- 2 Enter the name of the variable into which you'd like to place the result

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

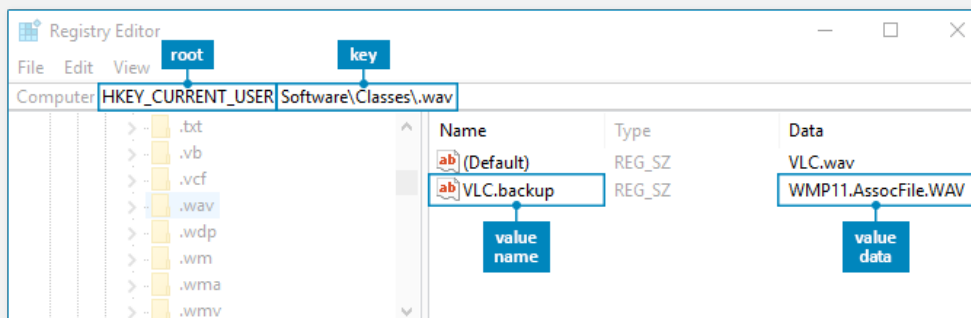
The anatomy of the Windows Registry

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.

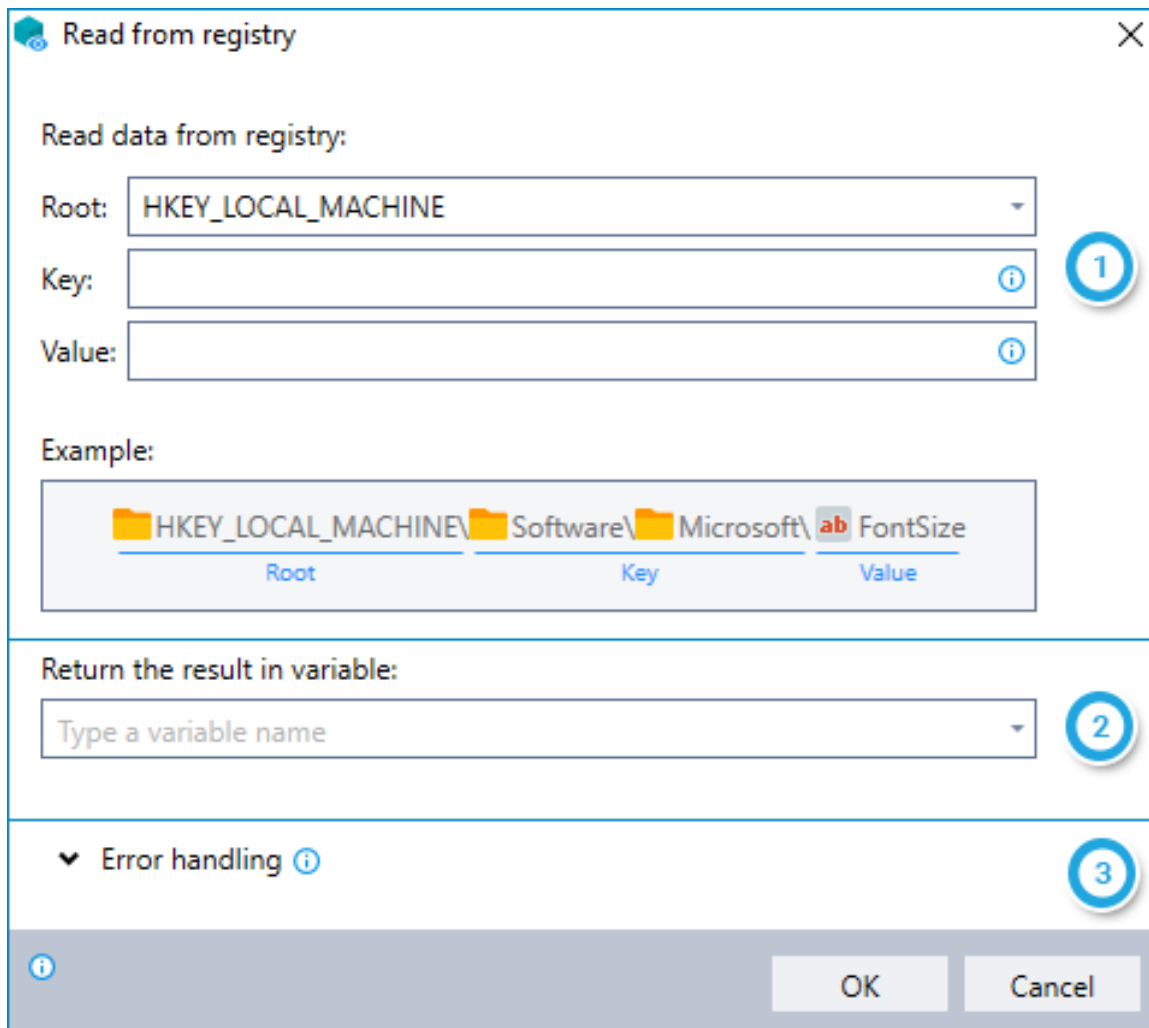
Registry Editor is the Windows component that allow you to view and edit the Registry. To access it, type `Run` from the Windows Start Menu, then `regedit` in the dialog box .

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

Get environment variable

Get the value of the environment variable:

Return the result in variable:

Type a variable name

OK Cancel

- 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.

Level	Date and Time	Source	Event ID	Task Categ...
Warning	6/15/2017 6:01:05 PM	Security-Kerberos	14	None
Information	6/15/2017 5:52:51 PM	GroupPolicy (Microsoft-Windows-...	1501	None
Warning	6/15/2017 5:00:58 PM	Security-Kerberos	14	None
Information	6/15/2017 4:41:56 PM	GroupPolicy (Microsoft-Windows-...	1500	None
Information	6/15/2017 4:04:50 PM	GroupPolicy (Microsoft-Windows-...	1501	None
Warning	6/15/2017 4:00:26 PM	Security-Kerberos	14	None
Information	6/15/2017 3:54:22 PM	Time-Service	37	None
Warning	6/15/2017 3:44:43 PM	DNS Client Events	8016 (1028)	
Warning	6/15/2017 3:44:43 PM	DNS Client Events	8015 (1028)	
Warning	6/15/2017 3:44:31 PM	DNS Client Events	8016 (1028)	
Warning	6/15/2017 3:44:31 PM	DNS Client Events	8015 (1028)	
Warning	6/15/2017 3:44:14 PM	DNS Client Events	1014 (1014)	
Warning	6/15/2017 3:44:07 PM	DNS Client Events	8015 (1028)	
Warning	6/15/2017 3:44:02 PM	Time-Service	131	None
Warning	6/15/2017 3:44:02 PM	Time-Service	131	None
Warning	6/15/2017 3:43:53 PM	DNS Client Events	1014 (1014)	
Warning	6/15/2017 3:43:50 PM	DNS Client Events	8016 (1028)	

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

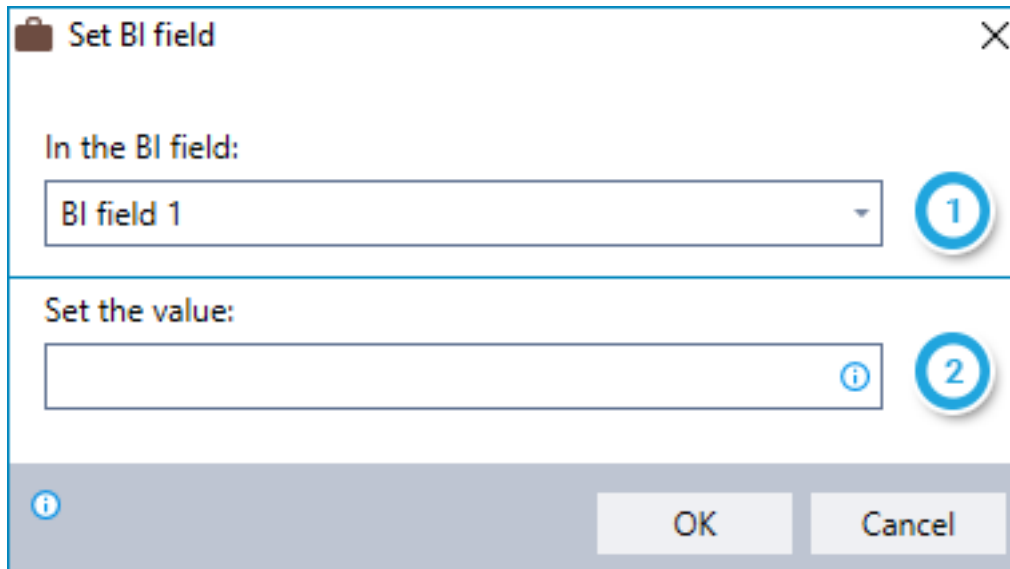
3 Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).

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 main sections. The first section, labeled "In the BI field:", contains a dropdown menu with "BI field 1" selected. A blue circle with the number "1" is positioned to the right of the dropdown. The second section, labeled "Set the value:", contains a text input field. A blue circle with the number "2" is positioned to the right of the input field. 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 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 ConsoleX 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 'XPATh Tester...' link and an information icon.
- 3** Radio buttons for 'First result', 'Last result', and 'All results' (which is selected). Below them 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 'JSONPATH Tester...' link to its right and a blue circle with the number 2 next to it.
- 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 is a 'Delimiter' field with a comma character and a blue circle with the number 4 next to it.
- 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 a blue circle with the number 5 next to it.

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 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

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	249
Get Email Messages	256
Email: Get Data	263
Email: Move to Folder	264
Email: Forward	266
Email: Reply	268
Email: Delete	269
Email: Mark as Read/Unread	270
Email: Save Attachments	271
Email: Save Message	273

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

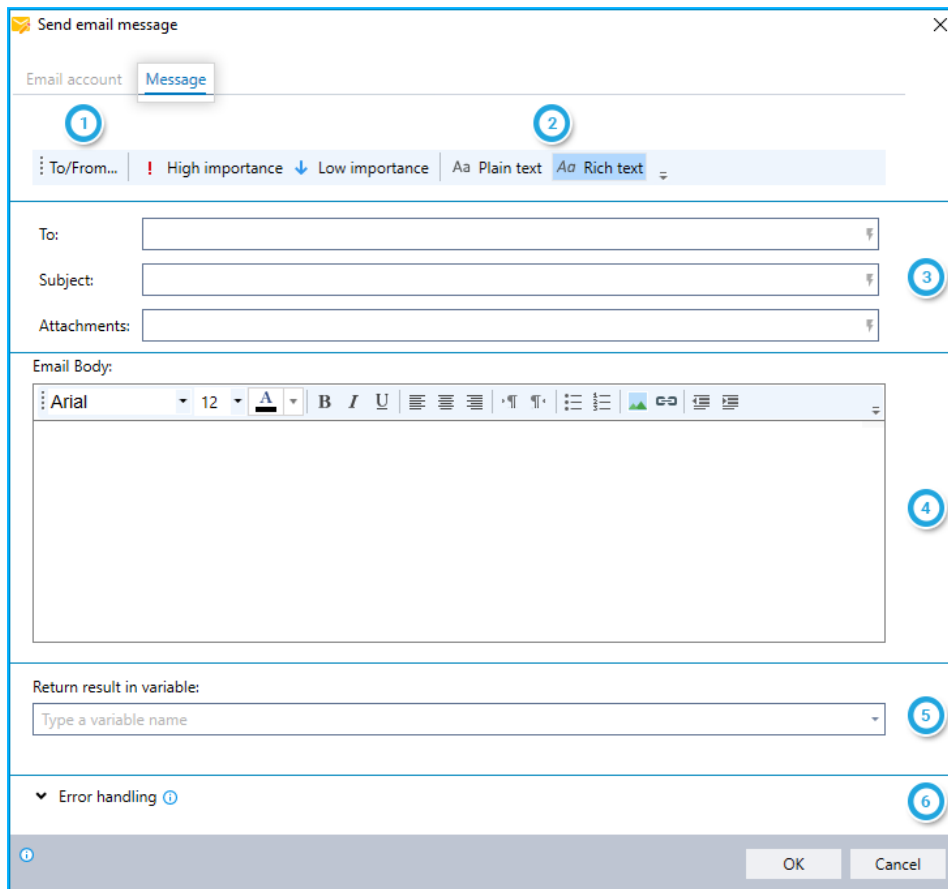
3 Mailbox:

- 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

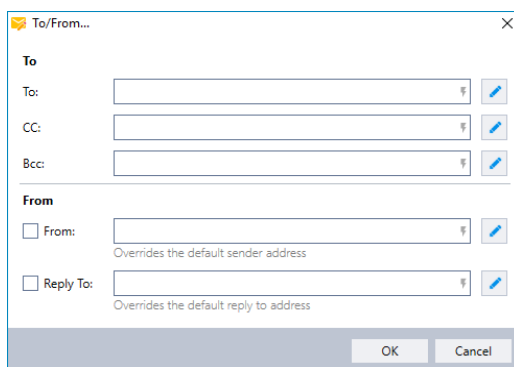
Note: The mailbox data entered will provide the default **FROM:** and **REPLY TO:** addresses for the message. These addresses can be overridden by utilizing [advanced address options](#) in the **Message** tab.

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.



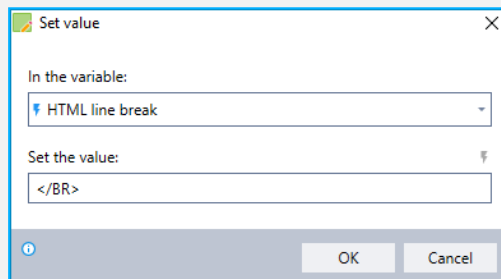
TROUBLESHOOTING TIP

Line breaks in rich text emails

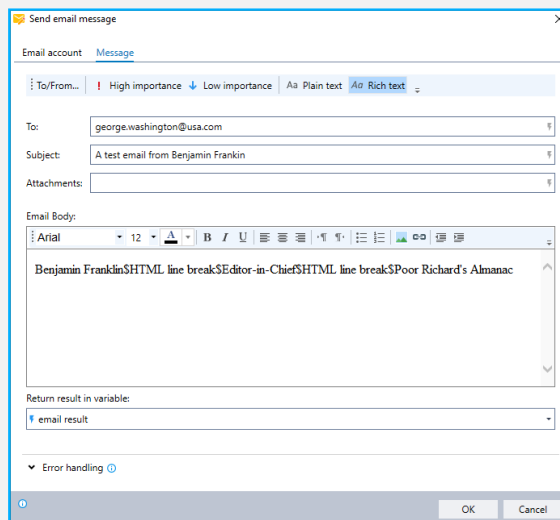
When sending an HTML/rich text email that contains "hard" line breaks, create a special character variable named, for example, **HTML line break**, and set its value to `</BR>`. Then use this variable to replace actual line breaks in your email text.

Example:

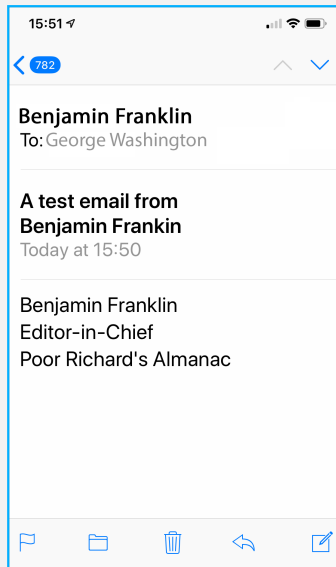
1. Set a variable to define the line break



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: [Empty text box]

Use SSL Port: [Empty text box]

Auto (dropdown menu)

Test (button) (3)

Credentials (2)

From credentials vault: [Dropdown menu] Add new

Enter credential by variable: [Dropdown menu]

Enter manually:


Username: [Text box]

Password: [Text box]

▼ 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**

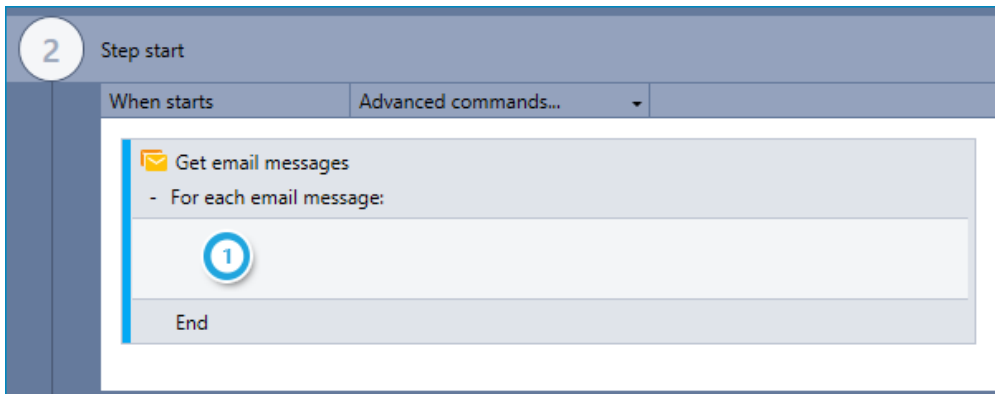
Get messages tab

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

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

- 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 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

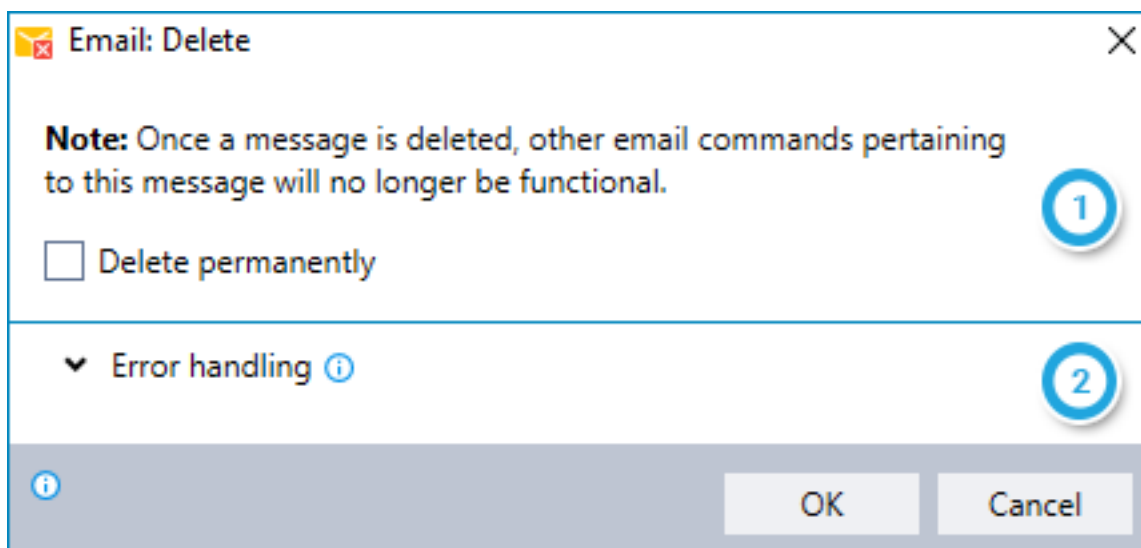
This command can be used only within a [GET EMAIL MESSAGES](#) container.



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.

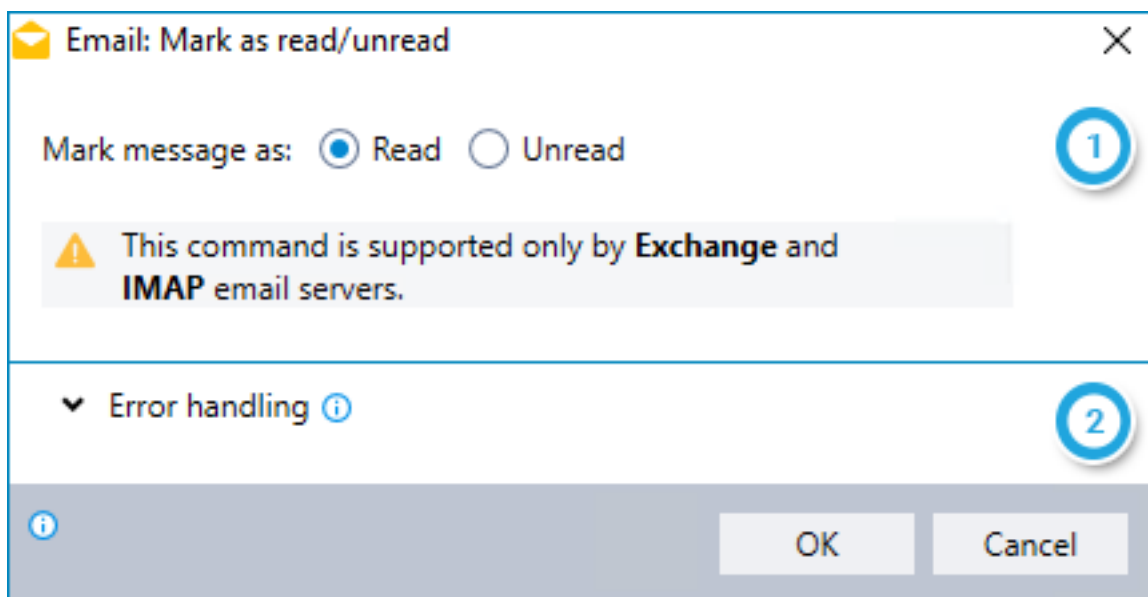


NOTE

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)

- 3** Enter the name of the variable into which you'd like to place the name(s) of the saved files
 - Multiple filenames will be returned in the variable separated by commas
- 4** Instruct 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	275
Run .NET Plugin Method	278
Call Web Service Method	280
Run Script	281
Open URL Link (beta)	284
Run curl Command	285
Get Web Page HTML	286
Run JavaScript on Page	287

Run Program

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

What are command line parameters?

Command lines parameters are 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, Microsoft's [command line parameters for Excel](#).

Using the RUN PROGRAM command

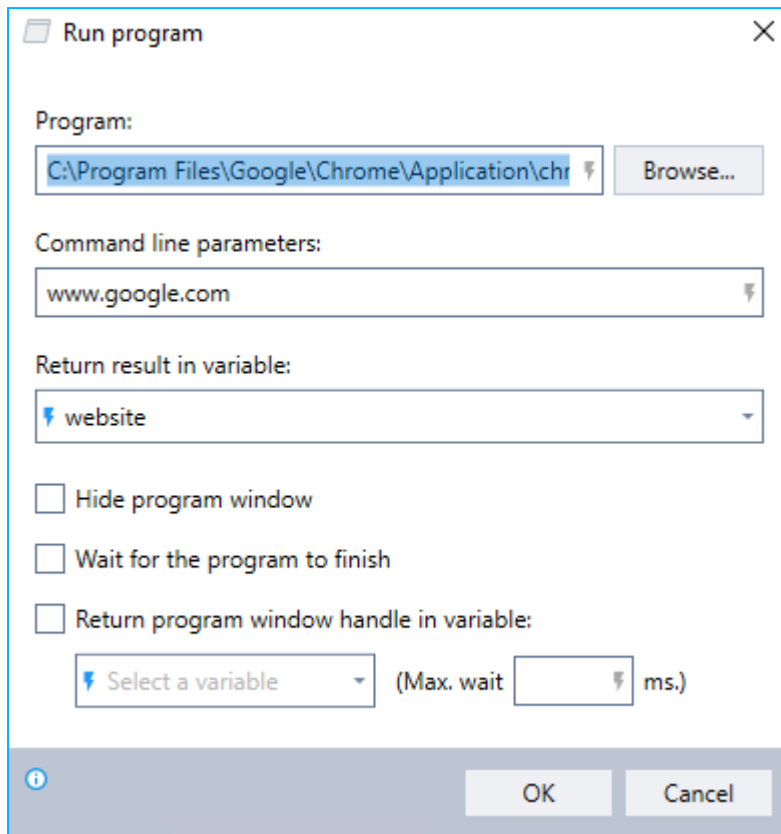
The screenshot shows the 'Run program' dialog box with the following fields and options:

- 1**: Program: [Text input field] [Browse... button]
- 2**: Command line parameters: [Text input field]
- 3**: Return result in variable: [Dropdown menu with 'Type a variable name' selected]
- 4**: [Hide program window checkbox] [Wait for the program to finish checkbox] [Return program window handle in variable: checkbox]
- [Return program window handle in variable: dropdown menu with 'Type a variable name' selected] (Max. wait [Text input field] ms.)
- [OK button] [Cancel button]

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

EXAMPLE:



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.

Method to Run:

String downloadOnlyFileInFolder(String userID, String folderID, PathTo...

Parameters:

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

Return result in variable:

result

Error handling

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

Important: If your script contains special characters, please use a file to run them. If not, the special character will be removed. This is a known limitation.

Using the RUN SCRIPT command

The screenshot shows a 'Run script' dialog box with the following fields and controls:

- 1**: A large text area for 'Script Code' with a 'Load from file...' link below it.
- 2**: A 'Parameters' text input field.
- 3**: A 'Language' dropdown menu currently set to 'VbScript'.
- 4**: A 'Timeout' spinner box set to '60' with the unit 'seconds'.
- 5**: A checkbox for 'Use Custom Encoding' and a corresponding dropdown menu.
- 6**: A 'Return result in variable' dropdown menu with the text 'Select a variable'.

Below the dropdowns, there is a note: 'Use WScript.StdOut.Write("My return value") to return value from script'. At the bottom, there is an 'Information' section stating 'Script parameters are space separated' and 'OK' and 'Cancel' buttons.

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.

NOTE: Custom encoding *isn't* relevant nor supported for Python.

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.

curl Run curl command

Use the curl command to transfer data from or to a server, using one of the supported protocols.

curl command:

Return result in variable (optional):

Return exit code in variable (optional):

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.

[Online documentation](#) | [Manual \(offline\)](#) | [How to use \(offline\)](#)

OK Cancel

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

Run your own JavaScript code and change the target website behavior; manipulate the behavior of the target website as needed with maximum flexibility.



NOTE

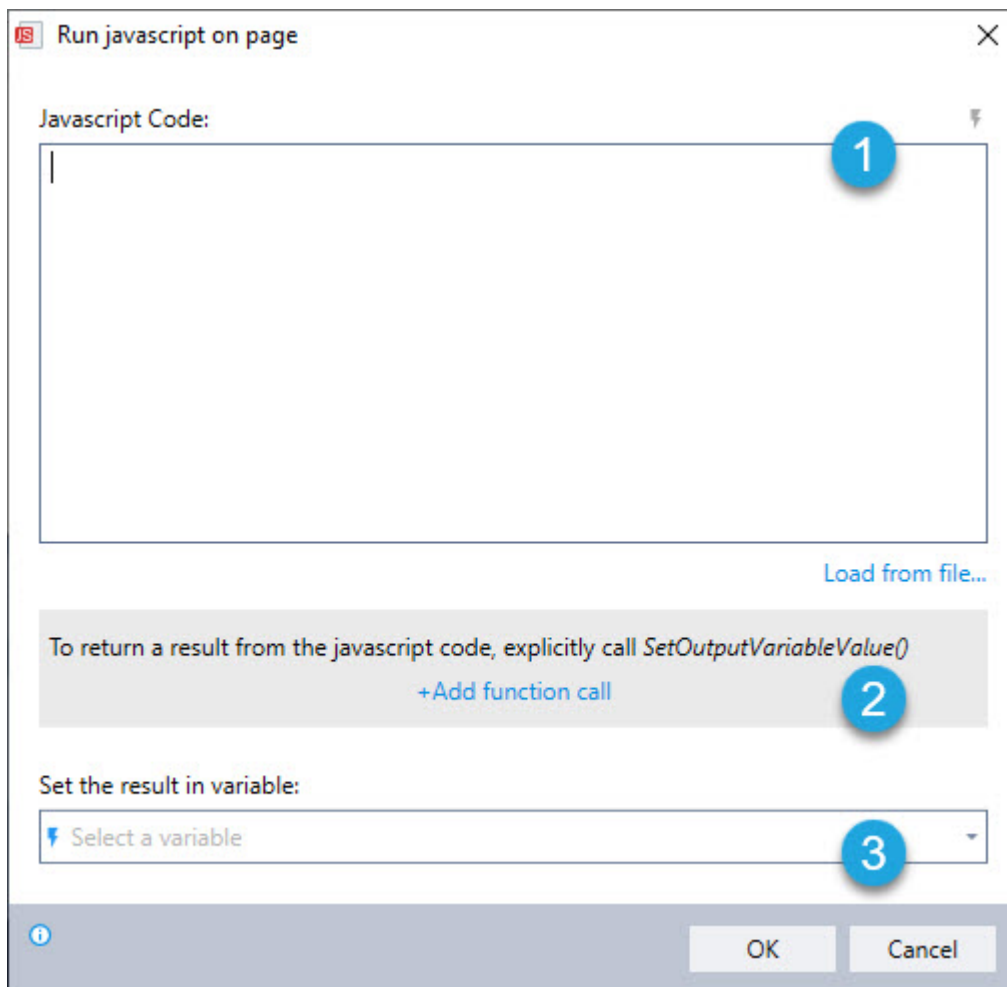
Make sure it's the right browser!

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



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



NOTE

Steps 2 and 3 are relevant only for a script that is designed to return a result.



NOTE

The command has a built-in timeout that stops the command from running if processing it takes longer than expected.

In **Firefox/Chrome**: the timeout is 5 seconds

In **Internet Explorer**: the timeout is 15 seconds

[Read more](#) about Timeout Monitoring.

CHAPTER 15: Database Commands

In this chapter:

Execute SQL Query	290
Monitor Database Changes	293

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.

The screenshot shows the 'Execute SQL query' dialog box with the 'Predefined Query' tab selected. The 'Query name' is 'Stocks Starts With Returns a list of stocks starts with'. The 'Query content' is 'select * from Stock where StockName like \$Prefix;'. The 'Parameters' table is as follows:

Name	Value	Type
Prefix	\$First Letter\$%	Automatic

Other settings include 'Return result in variable' set to 'Results', 'Query output' set to 'All rows', 'Row delimiter' set to '###', and 'Column delimiter' set to '::'. The 'Error handling' section is expanded at the bottom.

Custom query

The screenshot shows the 'Execute SQL query' dialog box with the 'Custom Query' tab selected. Numbered callouts (1-6) highlight the following fields:

- Connection string
- Get username and password from credentials vault
- Query content
- Error handling (General error value: ERROR, SQL error value: DBError)
- Return result in variable
- Query output (All rows selected), Row delimiter (Line break), and Column delimiter (Comma)

The 'OK' and 'Cancel' buttons are visible at the bottom right.

- 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

The screenshot shows the 'Monitor database changes' dialog box with the following configuration areas highlighted by numbered callouts:

- 1**: Connection string field with a 'Builder...' link.
- 2**: Check box for 'Get username and password from credentials vault' and an 'Add new' link.
- 3**: 'Table to monitor' dropdown menu and checkboxes for 'Insert', 'Update', and 'Delete'.
- 4**: 'Required table columns' section with dropdowns for 'UID', 'Update Date', and 'Is Deleted'.
- 5**: 'WHERE clause' text area with an example: 'Optional. Example: COUNTRY_CODE = 'US''.
- 6**: 'Polling cycle' (30 seconds) and 'Timeout' (5 minutes) fields.
- 7**: 'Data to fetch' text area and radio buttons for 'Fetch up to 100 rows' (selected) and 'Fetch first changed row only'.
- 8**: 'Return result in variables' section with dropdowns for 'Inserted rows', 'Updated rows', and 'Deleted rows'.
- 9**: 'Error Handling' dropdown menu.

- 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	296
Insert Password Into Active Field	297
Generate New Password	298
Revert Password	299

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	301
Get Automation Task Status	304
Get File Trigger Input	306
Get Folder Trigger Input	307
Get Email Trigger Input	308
Get Database Trigger Input	309

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

The screenshot shows the 'Add automation task to queue' dialog box. It contains the following elements:

- 1**: 'Task name' text input field.
- 2**: 'Queue Priority' dropdown menu, currently set to 'Normal'.
- 3**: 'Return initiated task ID in variable:' dropdown menu with the placeholder text 'Type a variable name'.
- 4**: 'Add this task to the user's robot task queue viewer' checkbox, which is currently unchecked. Below it is a descriptive text: '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.'
- 5**: 'When task ends, place task's output in a variable (optional):' dropdown menu with the placeholder text 'Type a variable name'.
- 6**: 'If ended **successfully**:' dropdown menu, currently set to 'No further action'.
- 7**: 'If ended **unsuccessfully**:' dropdown menu, currently set to 'No further action'.
- 7**: 'If **failed**:' dropdown menu, currently set to 'No further action'.
- 7**: 'Error handling' section, which is currently collapsed.

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

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. 1

File action
 Type a variable name
 Expected values: NEW / MODIFIED / DELETED 2

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

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

1

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

Row delimiter
Type a variable name

Column delimiter
Type a variable name

Database action
Type a variable name
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	312
Paste to Excel	318
Delete from Excel	324
Excel Worksheet Actions	327
Excel Row Actions	329
Excel Column Actions	332
Convert Excel to CSV	335
Create New Excel File	337
Query From Excel	338
Run Macro	340



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 fields and callouts:

- 1**: 'Where to copy from:' dropdown menu.
- 2**: 'Password:' text input field.
- 3**: 'What to copy' section, including 'Worksheet' dropdown and 'Select in Excel' button.
- 4**: 'Test' button.
- 5**: 'Copy method' dropdown menu.
- 6**: 'How to save' dropdown menu.
- 7**: 'Error handling' section, including 'Error variable' dropdown and various error type text inputs.
- 8**: 'Enable timeout monitoring' checkbox.

Other fields include 'Browse...', 'Column', 'Row', 'Column delimiter', and 'Row delimiter'.

1. Click **Browse** 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*
 - [entering them manually](#) (this method can be particularly useful when you want to use variable values to select cells)

1. Click **Test** to display a sample of the data that will be copied based on your selections.

Month	Product	State	Sales \$	
January	Volcano Blend	California	955	
January	Turkish Delight Roast	California	152	
January	Turkish Delight Roast	Oregon	319	
January	Breakfast Blastoff	Oregon	983	
January	Best Blend of Arabia	California	489	
January	Kona Koast	California	48	
January	Best Blend of Arabia	Oregon	218	

The above data is a sample of your selection

OK Cancel

2. Select whether to copy the values as **Actual Values or Formatted Text**.
3. 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
4. Instruct the wizard how to handle any errors encountered.
Read more about [Error Handling](#).
5. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).



TIP

Should I use Actual Values or Formatted Text as the copy method?

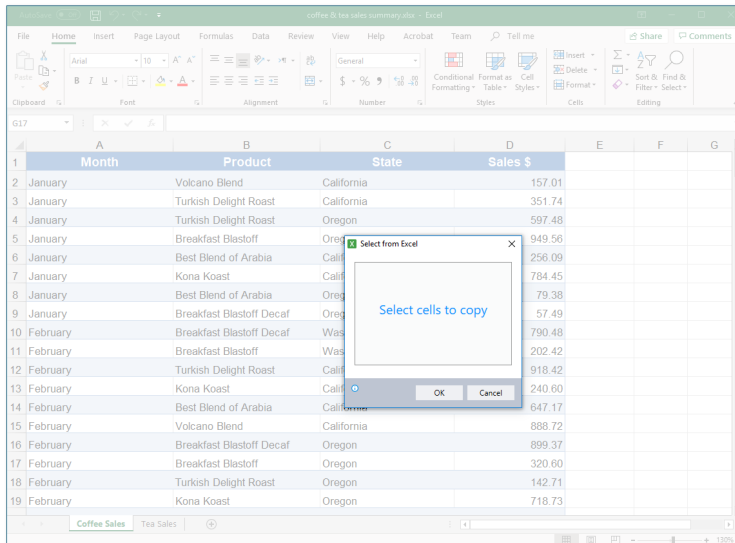
It's important to use the **Formatted Text** option when the relevant cells are formatted as dates. Otherwise, the values returned will be the serial numbers for the dates of the Excel. (For example: June 21, 2017 = serial number 42907)

For reading all other cells, use the **Actual Values** option.

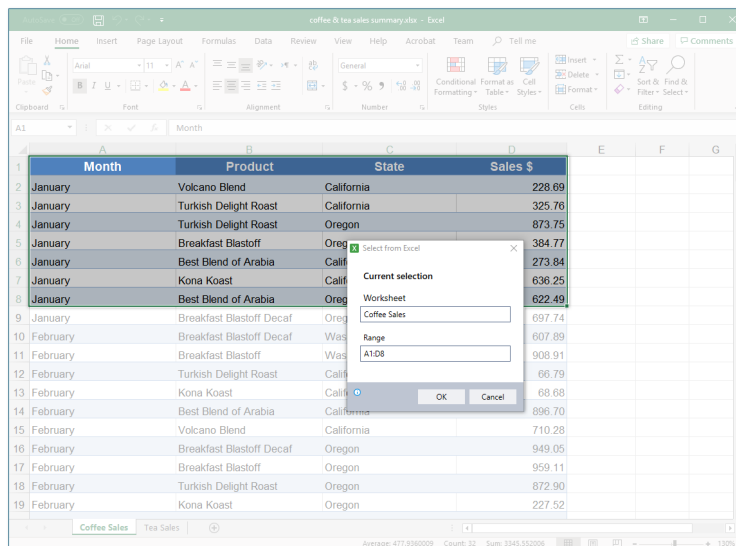
Also, **Formatted Text** provides the exact same text as in the excel. To avoid having fields presented as #####, the columns need to have the proper length - which you can do directly in the excel or by using Macro. You can also use the [Run Macro](#) advanced command to perform this action.

Selecting worksheet & cells directly in Excel

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



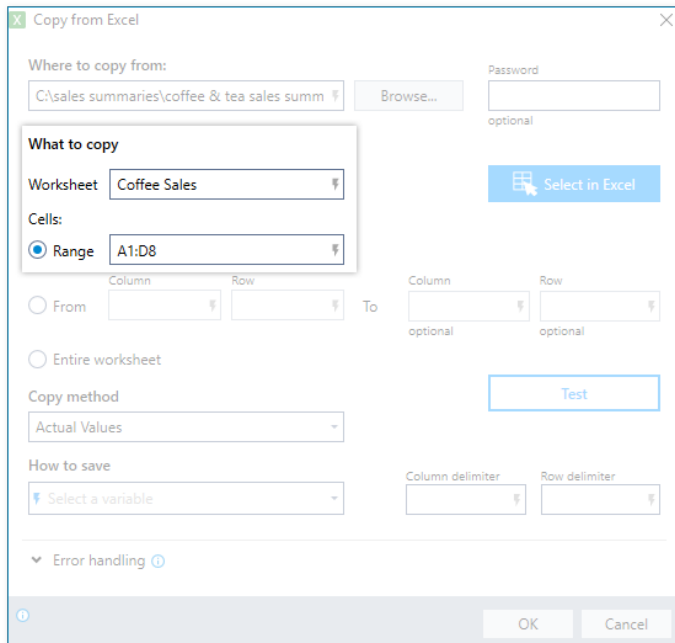
3. Click directly in the Excel file and select the worksheet (tab) and cells you want to copy from.
 - The worksheet name and cell range you have selected will appear in the selector window's **Current selection** fields



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 **Select in Excel** 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 numbered callouts:

- 1**: Radio button for 'Paste a table'.
- 2**: 'Select a variable' dropdown menu.
- 3**: 'Actual Values' dropdown menu under 'How to paste'.
- 4**: 'Browse...' button next to the 'Where to paste' file selection.
- 5**: 'Password' text input field.
- 6**: 'Example A2' text input field under 'Start pasting here: Cell address'.
- 7**: 'Error handling' section header.
- 8**: '1800' text input field for 'Raise error after'.

Other visible elements include: 'Paste a single value' radio button, 'Column delimiter' and 'Row delimiter' dropdowns, 'Create a new file if not exist' checkbox, 'Worksheet:' dropdown, 'Select in Excel' button, 'Column/Row' and 'Column' radio buttons with their respective dropdowns, and error handling options like 'Invalid path: INVALID_PATH', 'File not found: FILE_NOT_FOUND', 'Cell error: CELL_ERROR', 'File read only: FILE_READ_ONLY', 'General error: ERROR', and 'Timeout error text: TIMEOUT_ERROR'. 'OK' and 'Cancel' buttons are at the bottom right.

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

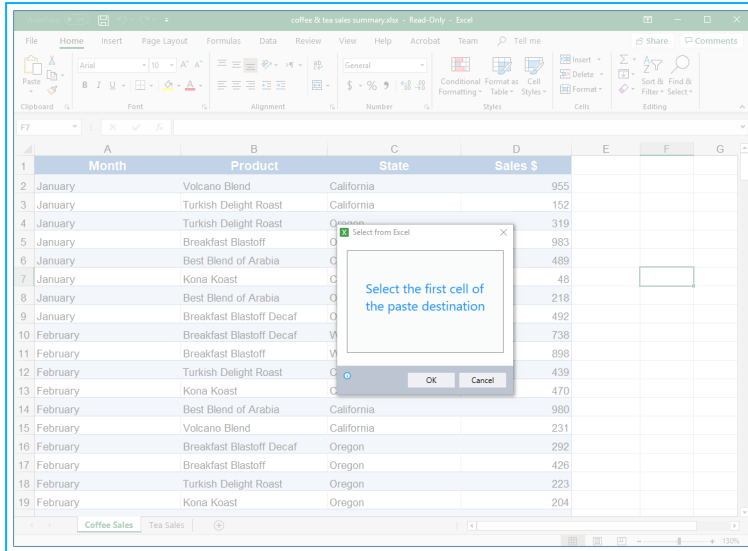
3. Select whether to paste the data as **Actual Values** or as **Formulas**.
4. Enter a variable, the path and name of the file to paste into or click **Browse** and navigate to the folder and choose the name of the Excel file within it that you want to paste the data into.

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 as well.
7. Instruct the wizard how to handle any errors encountered.
Read more about [Error Handling](#).
8. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).

Selecting worksheet & cells directly in Excel

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



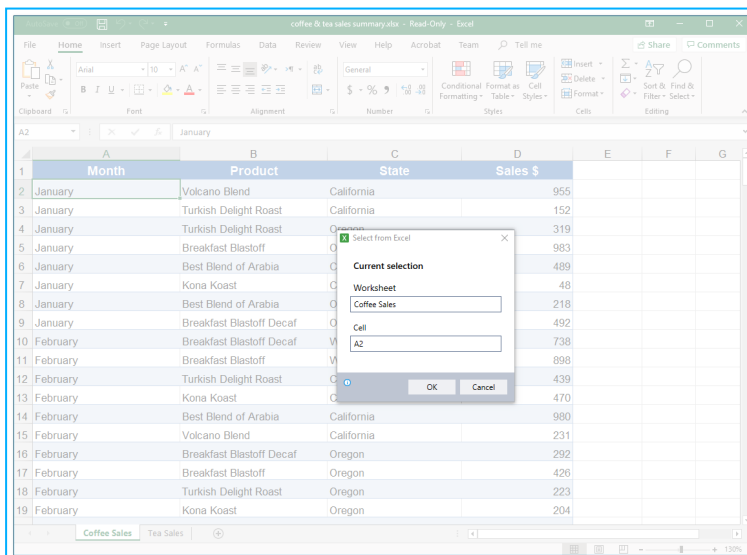
TIP

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.

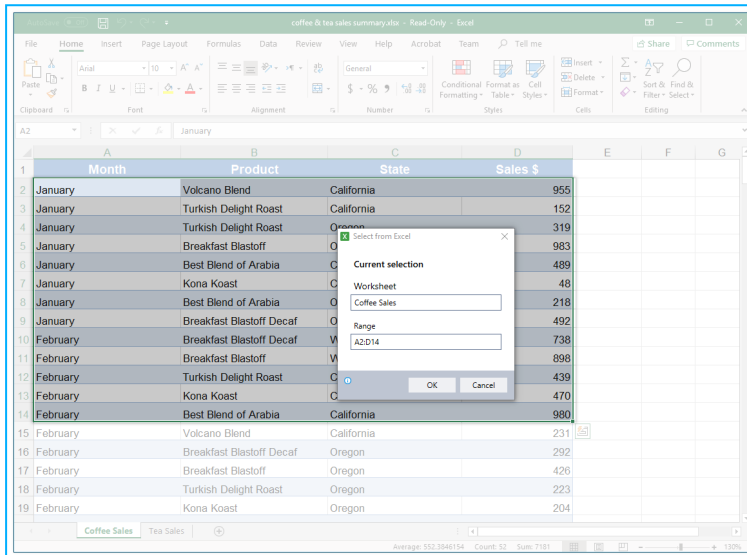
- The worksheet name and cell you have selected will appear in the selector window's **Current selection** fields.



- 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 **Select in Excel** 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 the three methods below for specifying where to **start pasting**:

- Cell address** – Designate the first destination cell using "standard" Excel format, e.g., A2 (this is the method used by the **Excel selector**).
- 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).
- 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 the two methods below 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

The screenshot shows the 'Delete from Excel' dialog box with the following fields and options:

- Where to delete from:** A dropdown menu with 'Select an Excel file to delete from' and a 'Browse...' button. (Callout 1)
- Password:** A text input field. (Callout 2)
- What to delete:**
 - Worksheet:** A dropdown menu with 'Type worksheet name'. (Callout 3)
 - Cells:**
 - Range:** A dropdown menu with 'Example A2:C20'.
 - From:** Radio button selected. Includes 'Column:' and 'Row #' dropdowns.
 - To:** Includes 'Column #' and 'Row:' dropdowns, both labeled 'optional'.
 - Entire worksheet:** Radio button unselected.
- Error handling:** (Callout 4)
 - Error variable:** A dropdown menu with 'Select a variable'.
 - File not found:** Text input field with 'FILE_NOT_FOUND'.
 - Worksheet not found:** Text input field with 'SHEET_NOT_FOUND'.
 - Cell error:** Text input field with 'CELL_ERROR'.
 - General error:** Text input field with 'ERROR'.
 - Enable timeout monitoring:** Check box unselected. (Callout 5)
 - Raise error after:** Text input field with '1800' and 'seconds' label.
 - Timeout error text:** Text input field with 'TIMEOUT_ERROR'.

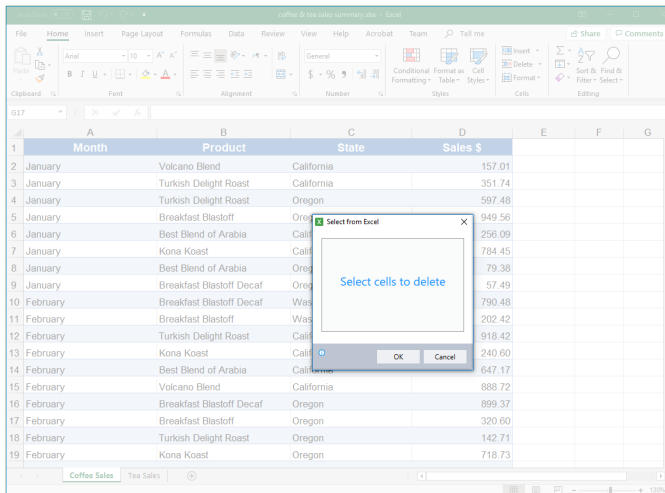
At the bottom right are 'OK' and 'Cancel' buttons. A 'Select in Excel' button is also present next to the Password field.

1. Click **Browse** 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:

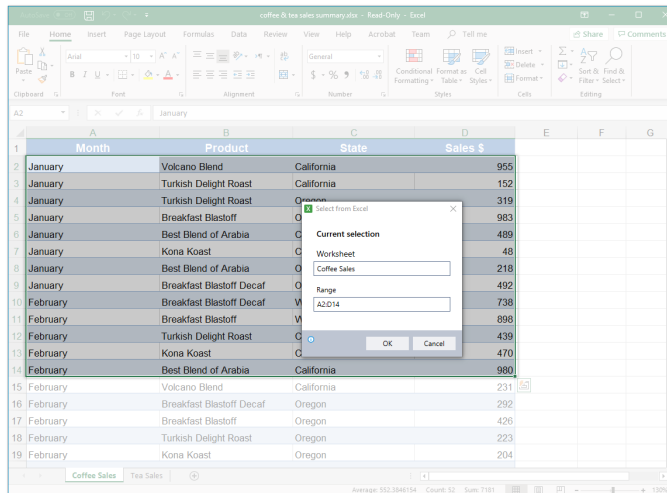
- by selecting them directly in Excel; **or**
 - by entering them manually (this method can be particularly useful when you want to use variable values to select cells).
4. Instruct the wizard how to handle any errors encountered.
Read more about [Error Handling](#).
 5. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).

Selecting worksheet & cells directly in Excel

1. Click **Select in Excel** 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.

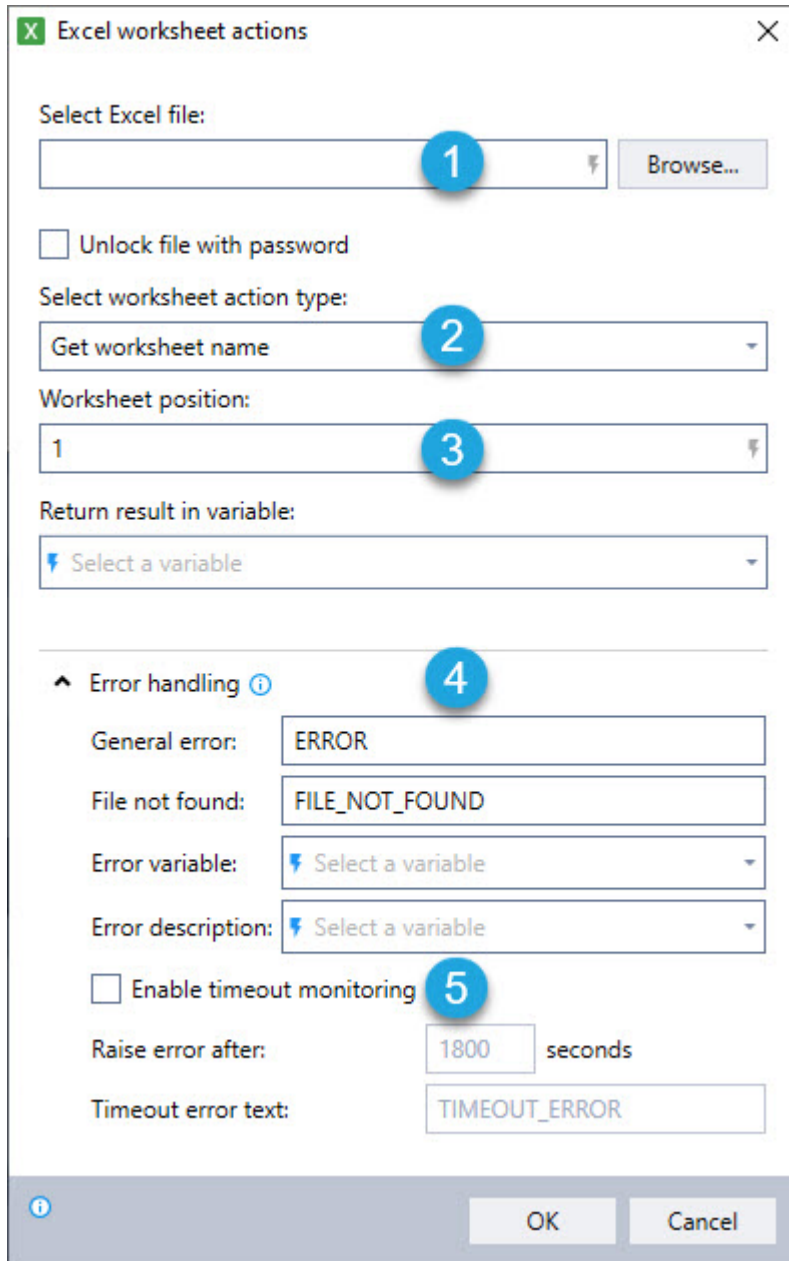
Entering worksheet & cells manually

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](#).
 5. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).

Excel Row Actions

- Retrieve the number of rows within a specified column and place it into a new or existing variable; *or*
- Insert or delete rows

Using the EXCEL ROW ACTIONS command

The screenshot shows the 'Excel row actions' dialog box with the following fields and callouts:

- 1**: 'Select Excel file:' text label.
- 2**: 'Worksheet name' radio button.
- 3**: 'Action:' dropdown menu.
- 4**: 'Number of empty cells allowed:' text label.
- 5**: 'Error handling' section header.
- 6**: 'Enable timeout monitoring' checkbox.

Other visible fields include: 'Browse...' button, 'Unlock file with password' checkbox, 'Worksheet position' dropdown (value: 1), 'Column index or letter:' dropdown (value: 1), 'Return result in variable:' dropdown (value: Select a variable), 'General error:' text box (value: ERROR), 'File not found:' text box (value: FILE_NOT_FOUND), 'Worksheet not found:' text box (value: WORKSHEET_NOT_FOUND), 'Error variable:' dropdown (value: Select a variable), 'Error description:' dropdown (value: Select a variable), 'Raise error after:' text box (value: 1800) followed by 'seconds', and 'Timeout error text:' text box (value: TIMEOUT_ERROR). At the bottom are 'OK' and 'Cancel' buttons.

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 rows within a specified column.
 - You can 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.
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](#).
6. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).

Excel Column Actions

- Retrieve the number of 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 callouts:

- 1**: 'Select Excel file:' text label above a text input field.
- 2**: 'Worksheet name' radio button and its corresponding text input field.
- 3**: 'Action:' dropdown menu, currently set to 'Get column count'.
- 4**: 'Number of empty cells allowed:' text input field, currently containing '1'.
- 5**: 'Error handling' section, specifically the 'General error:' text input field containing 'ERROR'.
- 6**: 'Raise error after:' text input field containing '1800' and the label 'seconds'.

Other visible fields include: 'Unlock file with password' (checkbox), 'Worksheet position' (radio button and text input field containing '1'), 'Row index' (text input field containing '1'), 'Return result in variable:' (dropdown menu), and 'Error variable:' (dropdown menu). The dialog also features 'OK' and 'Cancel' buttons at the bottom.

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 columns within a specified row.
 - You can 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](#).
6. Enable timeout monitoring (optional). Read more about [Timeout Monitoring](#).

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](#).
 5. Enable timeout monitoring (optional).
Read more about [Timeout Monitoring](#).

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

The screenshot shows the 'Create new Excel file' dialog box. It includes a text input field for the file name (marked with a blue circle 1), a 'Browse...' button, and a note: 'File extension must be one of the following: *.xls | *.xlsx | *.xlsm'. Below this is a checkbox for 'Overwrite existing file' (marked with a blue circle 2). The 'Error handling' section is expanded (marked with a blue circle 3) and contains the following fields:

- Error variable: Select a variable
- General error: ERROR
- File in use: FILE_IN_USE
- Invalid file extension: INVALID_FILE_EXTENSION
- File already exists: FILE_EXISTS

Below the error handling section is a checkbox for 'Enable timeout monitoring' (marked with a blue circle 4), a 'Raise error after' field with a value of 1800 seconds, and a 'Timeout error text' field with the value TIMEOUT_ERROR. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

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](#).
4. Enable timeout monitoring (optional).
Read more about [Timeout Monitoring](#).

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**: File selection field with a 'Browse...' button.
- 2**: 'Table data' section with radio buttons for 'All values in worksheet' (selected) and 'A specific range:'. Below is a range input field with 'Example: A2:D10'.
- 3**: 'Define parameters for your query:' section with a '+ Add' button and a table with columns 'Name', 'Value', and 'Type'.
- 4**: 'Return values in variable:' dropdown menu, 'Column delimiter:' and 'Row delimiter:' input fields.
- 5**: 'Error handling' section with fields for 'General error:', 'File not found:', and 'Error description:'.
- 6**: 'Raise error after:' input field set to '1800' seconds.

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](#).
 6. Enable timeout monitoring.
Read more about [Timeout Monitoring](#).

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

Run Macro on file:

1

Unlock file with password

Macro is embedded in file 2

Custom macro [Edit](#)

Module name:

Procedure name:

Set the macro function returned result in the variable:

3

Save file when macro ends 4

^ Error handling ⓘ

Error variable:

5 File not found:

File read only:

Code error:

General error:

6 Enable timeout monitoring

Raise error after: seconds

Timeout error text:

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 VPA editor of the Excel.
 - If written in the Advanced Command, click **Edit** 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](#).
 6. Enable timeout monitoring (optional).
Read more about [Timeout Monitoring](#).

CHAPTER 19: SAP Commands

In this chapter:

Get SAP Object Text	345
Get SAP Object Value	347
Get SAP Object Location	348
Set SAP Object Value	350



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

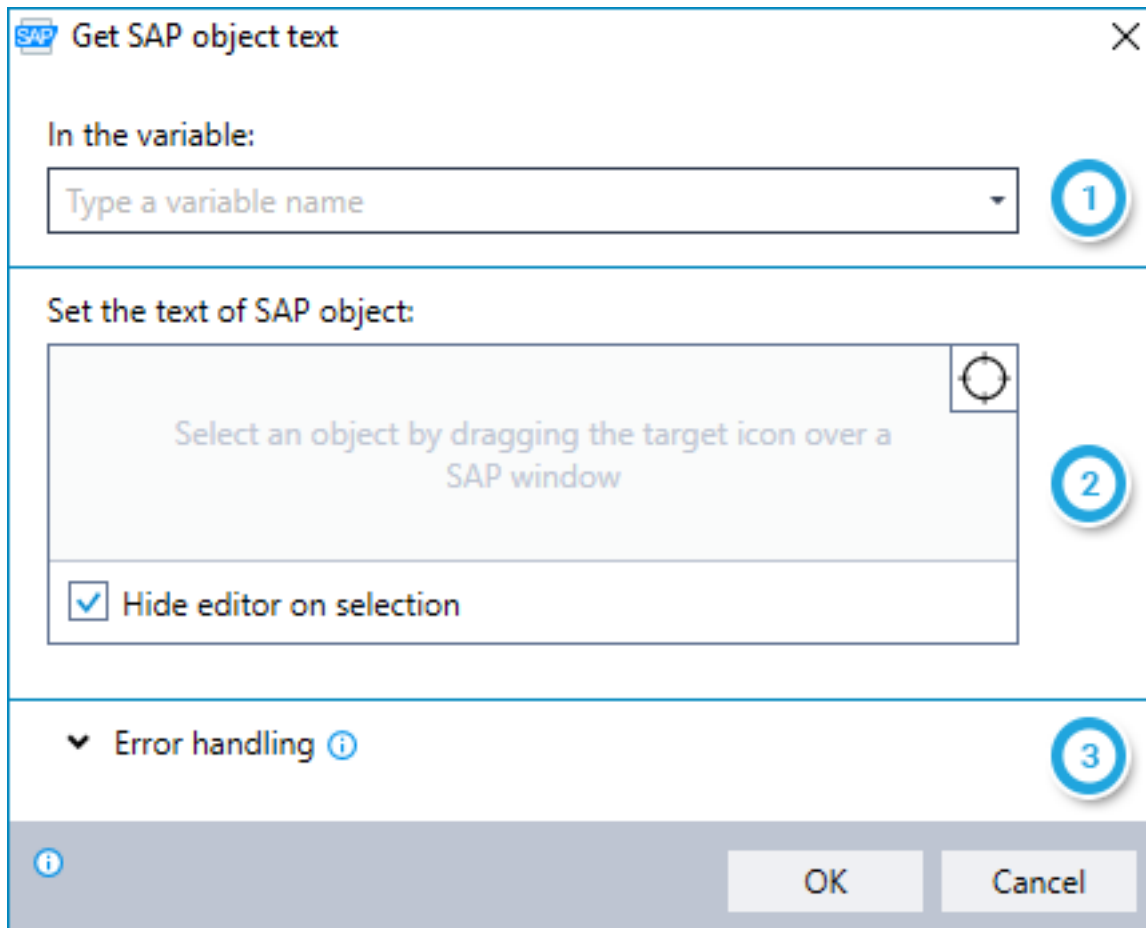
Use **SAP COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.





The supported SAP version is 7.4 *and later*.

Get SAP Object Text

Retrieve the text of an object in the active SAP window and place it into a new or existing variable. The supported SAP version is 7.4 and later.

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 Krypton 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



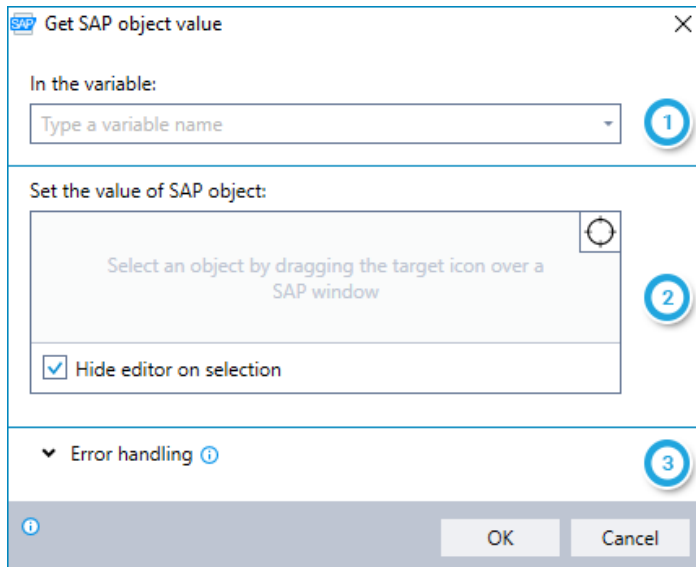
Instruct the wizard how to handle any errors encountered. Read more about [ERROR HANDLING](#).



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.

The supported SAP version is *7.4 and later*.

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

The supported SAP version is 7.4 and later.



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. It has a title bar with the SAP logo and a close button. The dialog is divided into several sections:

- In the variables:** This section contains four dropdown menus for 'Left', 'Top', 'Width', and 'Height', each with the placeholder text 'Type a variable name'. A blue circle with the number '2' is positioned to the right of these fields.
- Set the** 'rectangle' **of selected SAP object**: A dropdown menu is set to 'rectangle'. A blue circle with the number '1' is to the right.
- Select an object by dragging the target icon over a SAP window**: A large text area with a target icon in the top right corner. A blue circle with the number '3' is to the right.
- Hide editor on selection**: A checkbox that is checked.
- Error handling**: A dropdown menu with a blue circle with the number '4' to its right.

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

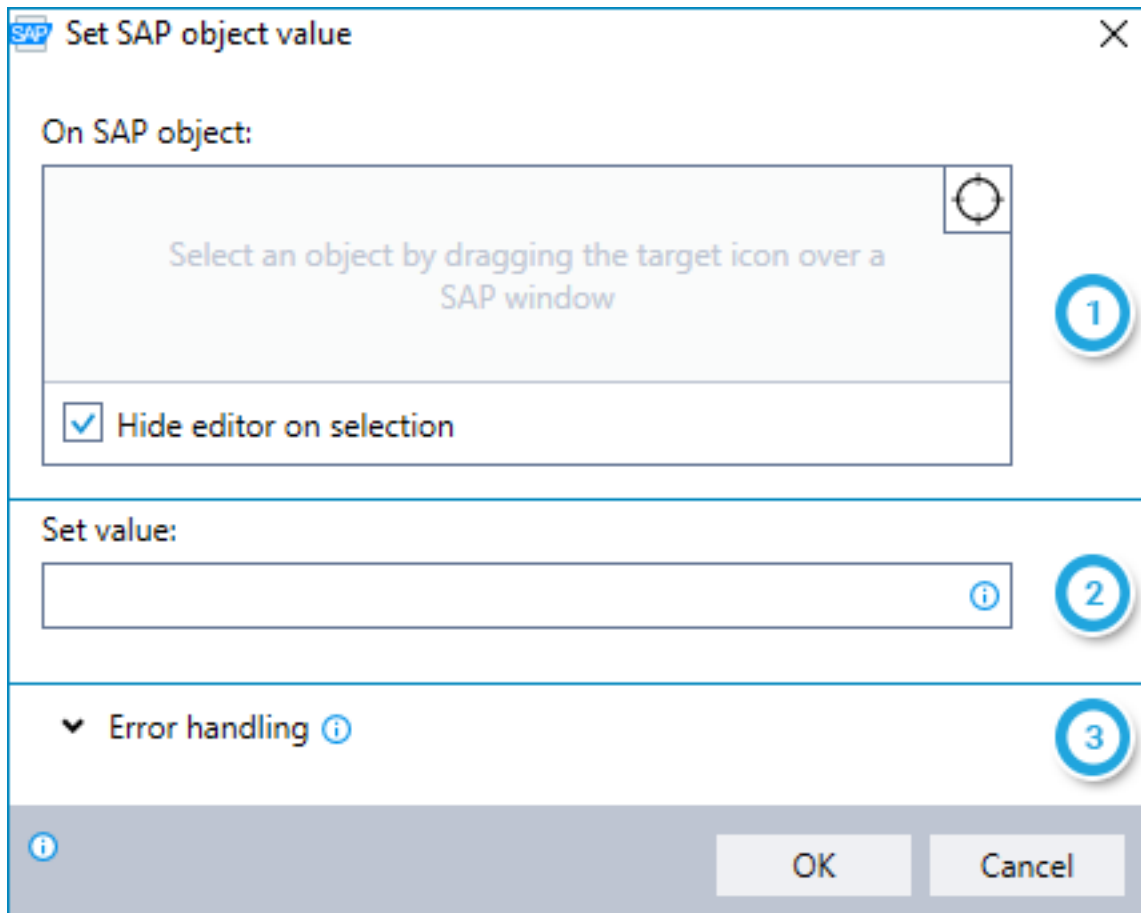
- 1 Choose whether to retrieve the object location either by rectangle or by center point
- 2 Enter the names of the variables into which you would like to place the location information
- 3 Select the object whose location you would like to retrieve by dragging the  icon onto the object in an SAP window
 - 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.

The supported SAP version is 7.4 and later.

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
 -  Display additional information about the selected object
 -  Test placing a value into the selected object
- 2 Enter the value you would like to place (can be free text or copied from values stored in variables)



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	353
Get UI Object Value	355
Get UI Object Location	357
Set UI Object Value	360
Extract UI table text	362
Click on UI object	364
Does UI Object Exist	366
Click on UI object	368



NOTE

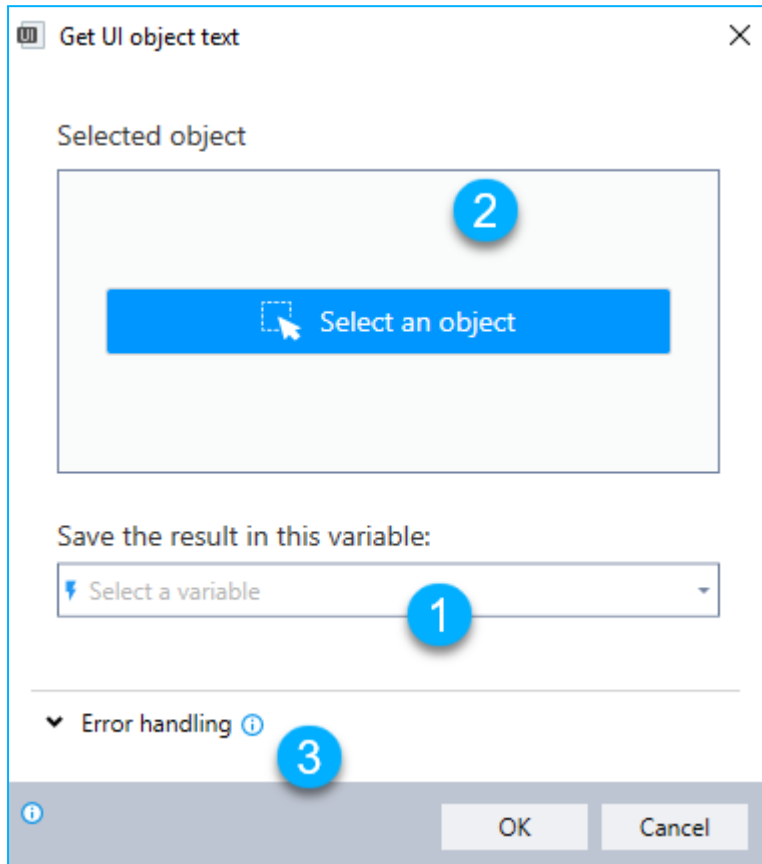
Use in recorded steps only

Use **UI AUTOMATION COMMANDS** within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

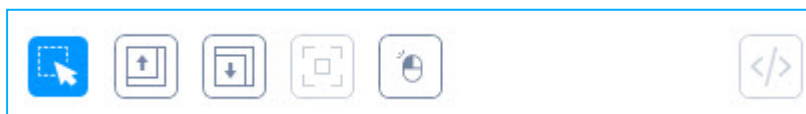
Get UI Object Text

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



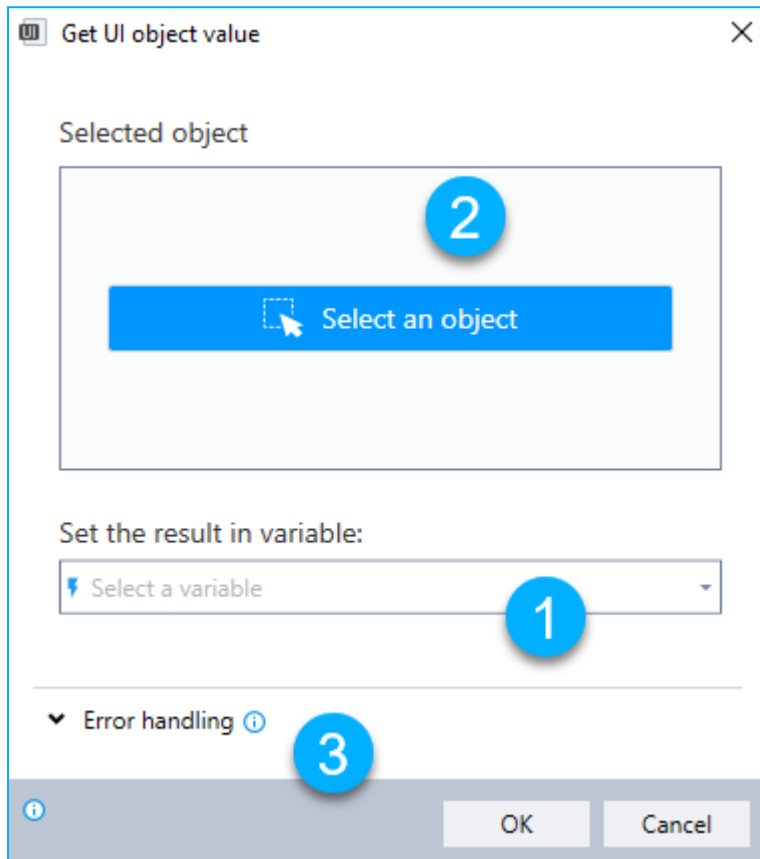
1. Enter the name of the variable into which you would like to place the text of the selected object
2. Select the object whose text you would like to copy.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



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.

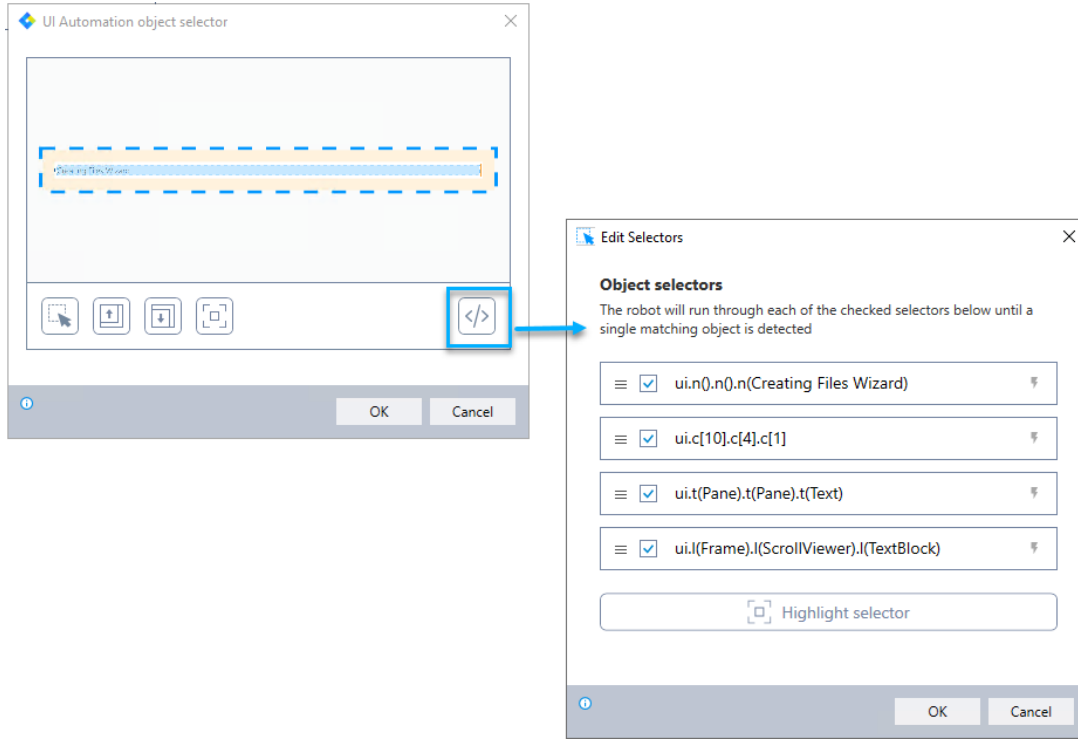
Using the GET UI OBJECT VALUE command



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.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



An example with Tree View selected:



See [UI Automation Selector Syntax](#)

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

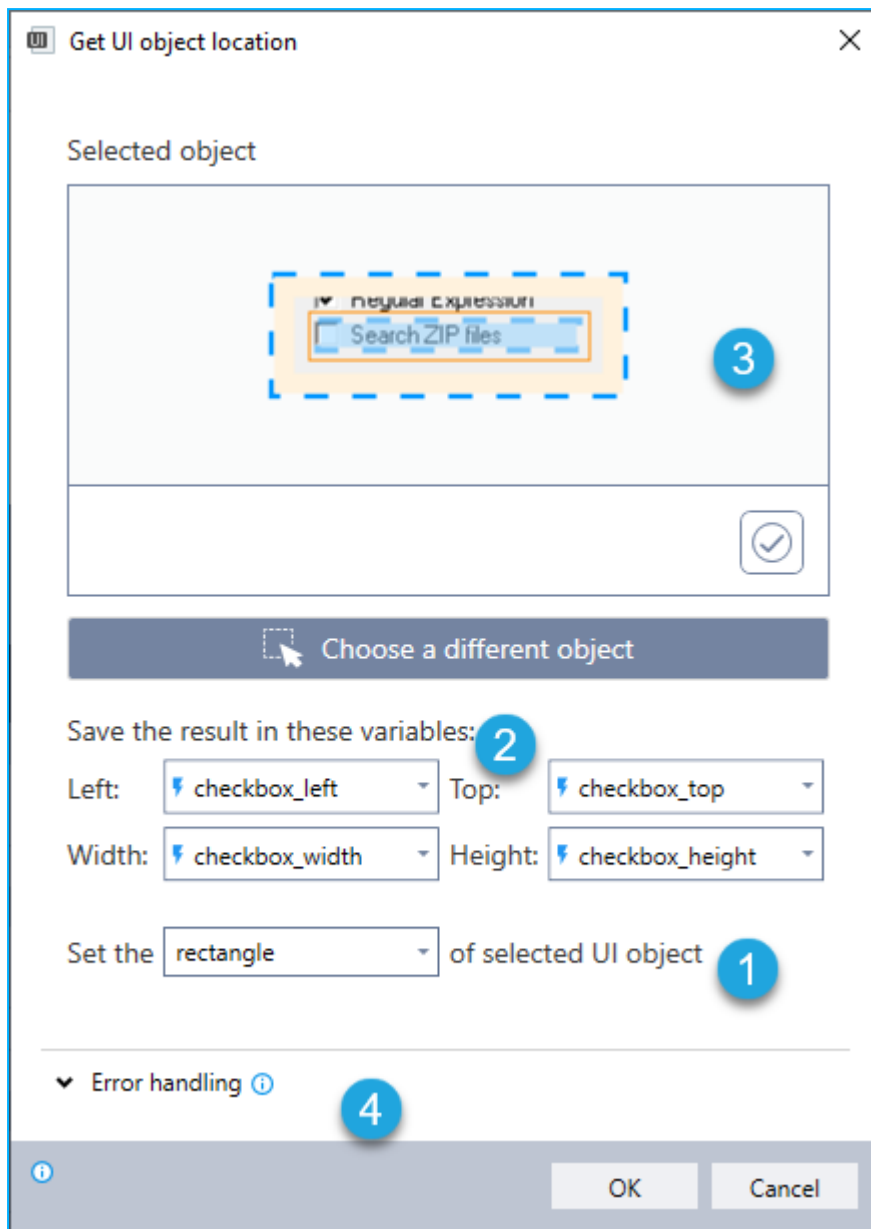


TIP

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.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection

- Tree view



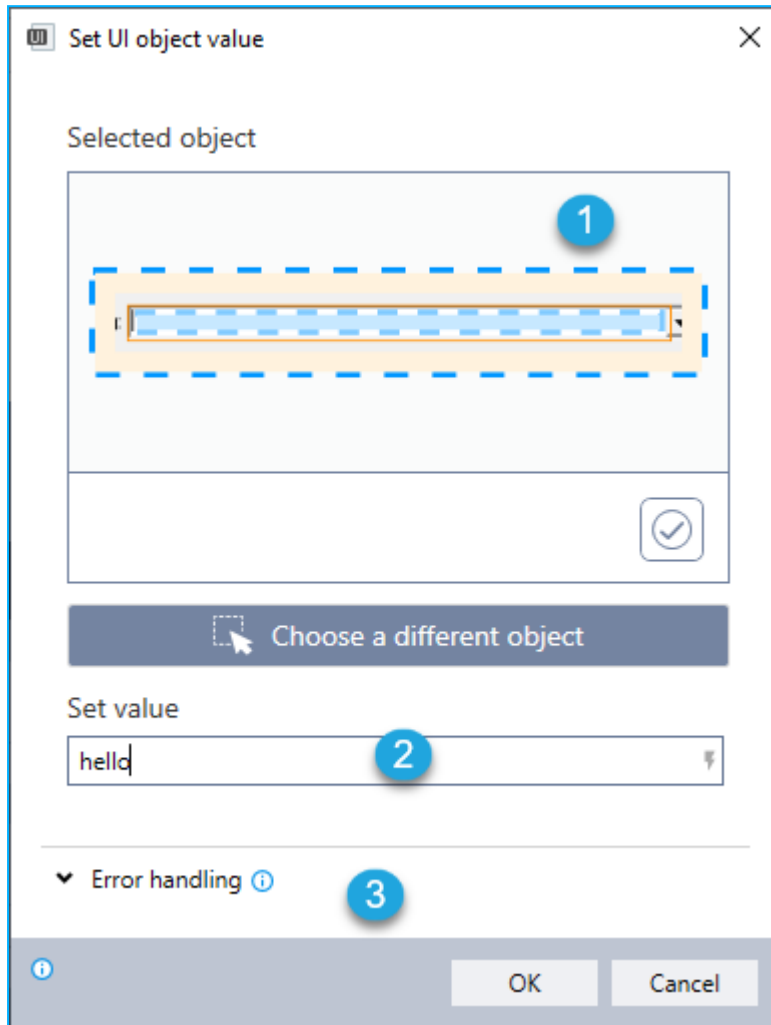
4. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

See [UI Automation Selector Syntax](#)

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 whose value you would like to set.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view

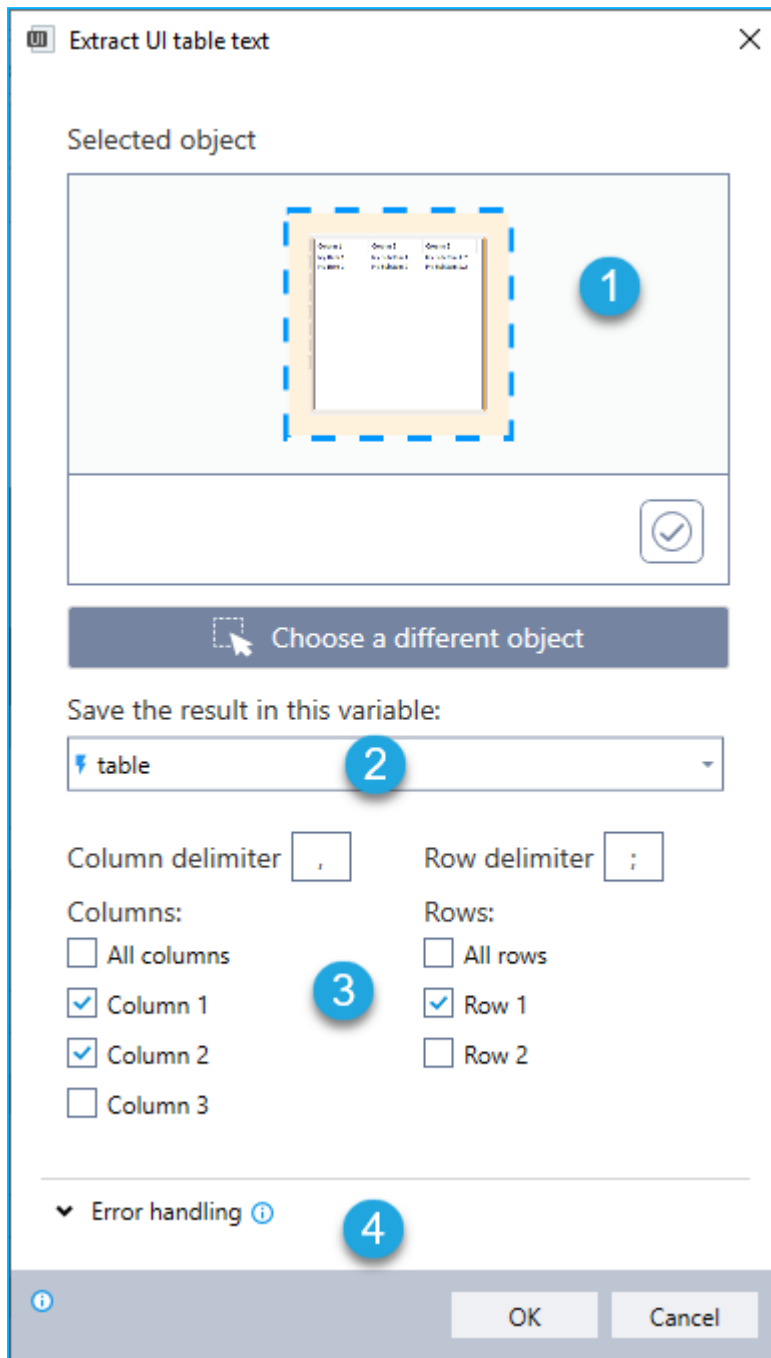


2. Enter the value you would like to place (can be free text or copied from values stored in variables).
3. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

See [UI Automation Selector Syntax](#)

Extract UI table text

Extract the text fields from the Java table into a variable. You can choose the rows and columns you want to extract.



1. Select the Java table whose text fields you want to extract.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection

- Highlight selected object
- Use right click for selection
- Tree view

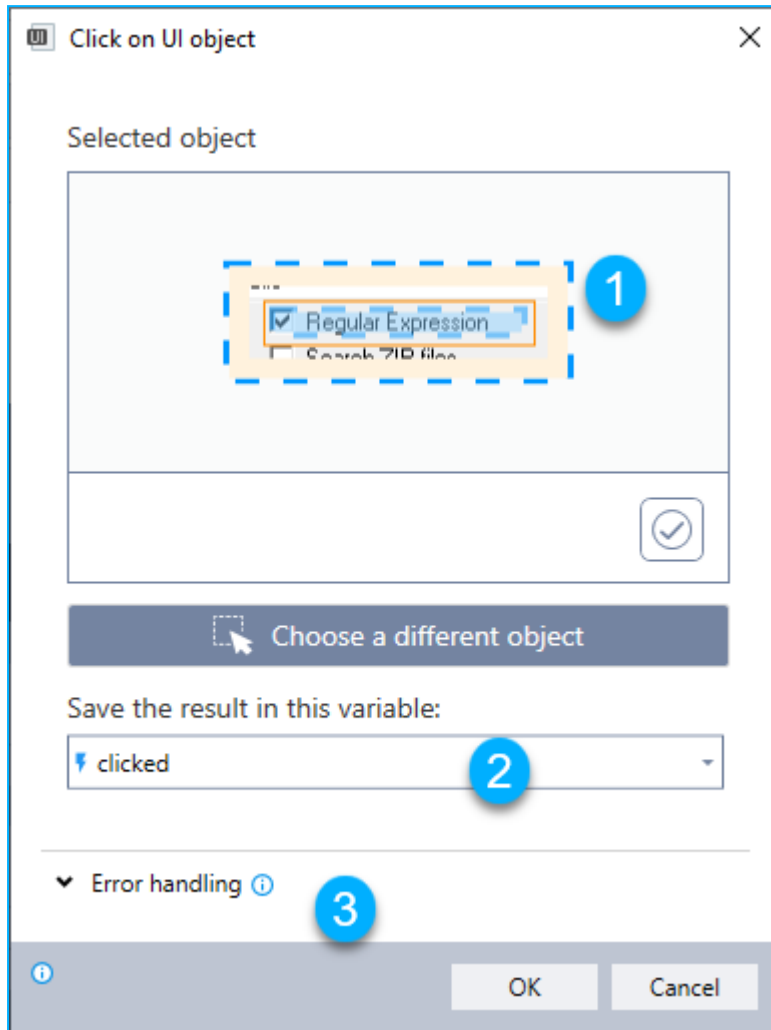


2. Enter the name of the variable into which you want to save the extracted text.
3. Set the delimiters of the relevant column and row from which you want to extract the text.
4. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

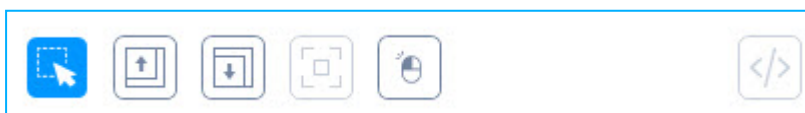
See [UI Automation Selector Syntax](#)

Click on UI object

Click the mouse on an object in the currently active UI.



1. Select the UI object you want to click.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



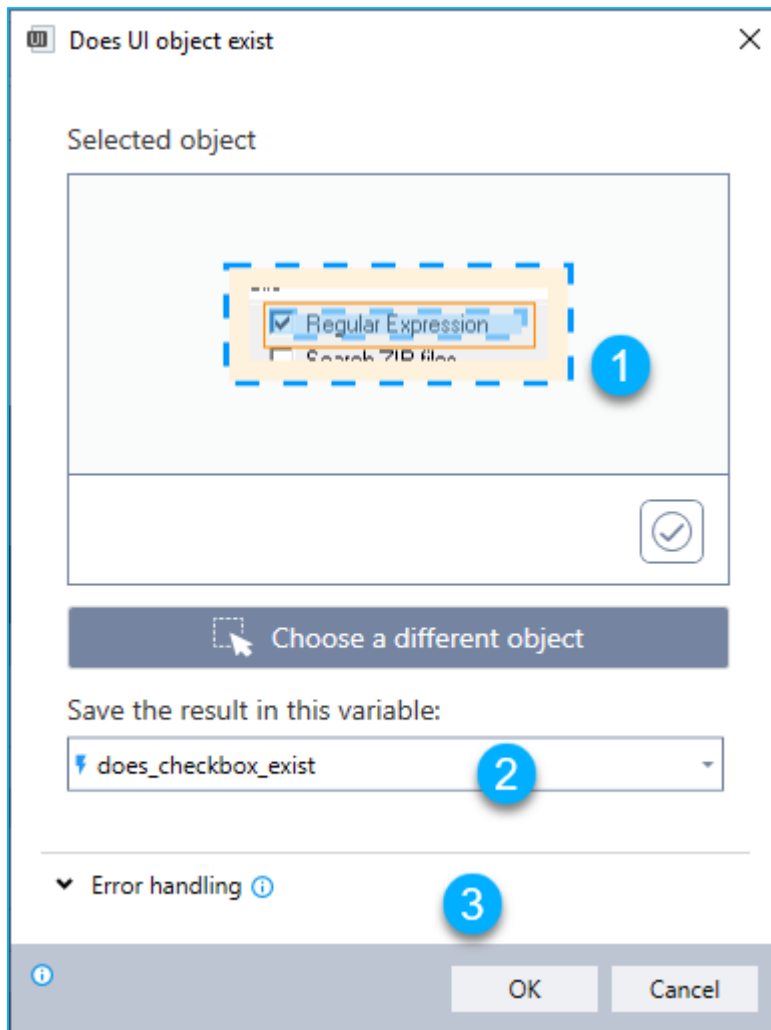
2. Enter the name of the variable into which you want to save the result.

- 3.** Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

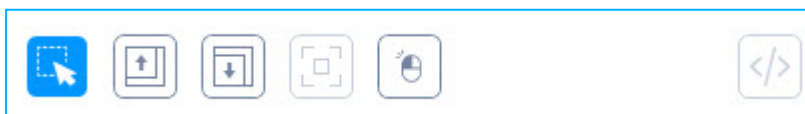
See [UI Automation Selector Syntax](#)

Does UI Object Exist

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



1. Select the UI object whose existence you want to check.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view

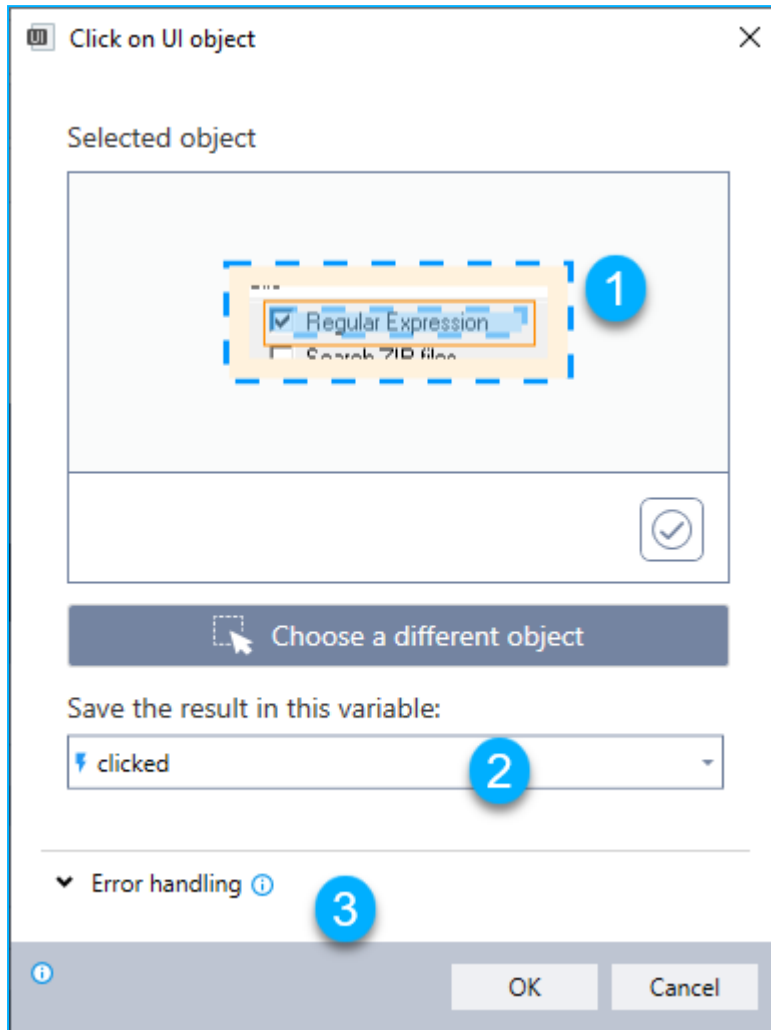


2. Enter the name of the variable into which you want to save the result of the query (TRUE/FALSE).
3. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

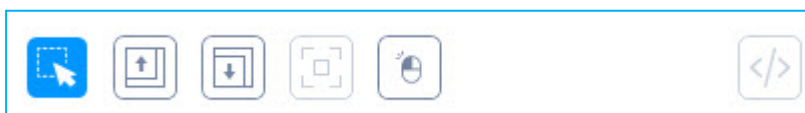
See [UI Automation Selector Syntax](#)

Click on UI object

Click the mouse on an object in the currently active UI.



1. Select the UI object you want to click.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



2. Enter the name of the variable into which you want to save the result.

- 3.** Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

See [UI Automation Selector Syntax](#)

CHAPTER 21: HTML Commands

In this chapter:

HTML Object Selector	372
Get HTML Table	377
Get HTML Object Text	379
Get HTML Object Value	382
Get HTML Object	384
Set HTML Object Value	386
Does HTML Object Exist	388
Click on HTML Object	390
Scroll to HTML Object	392
Extract from HTML Table/List	394



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

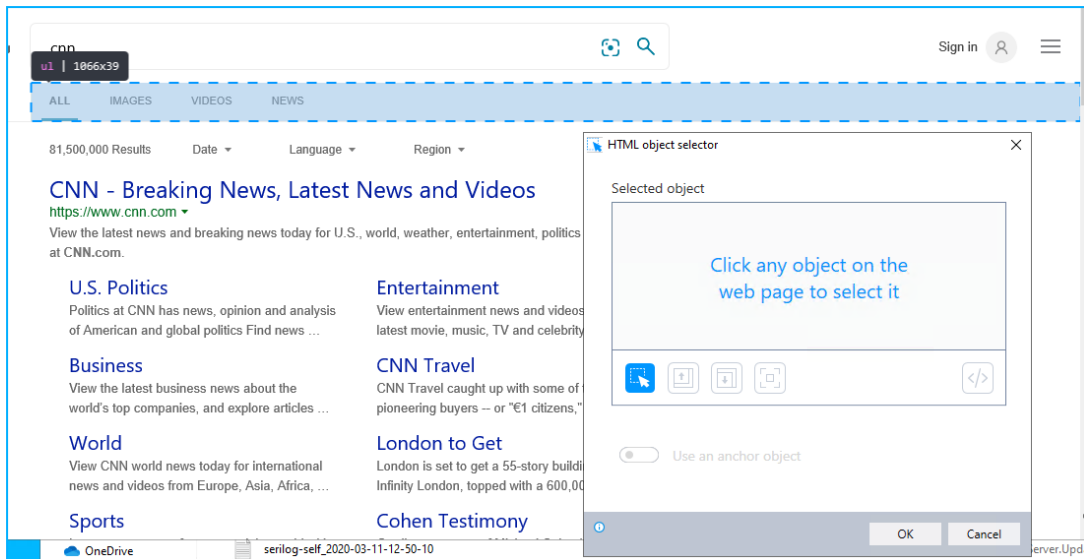


TIP

It's easiest to open your web browser and navigate to the desired page prior to using any of 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 **Select an object** to invoke the **HTML OBJECT SELECTOR**.
2. The **HTML OBJECT 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 **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.

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

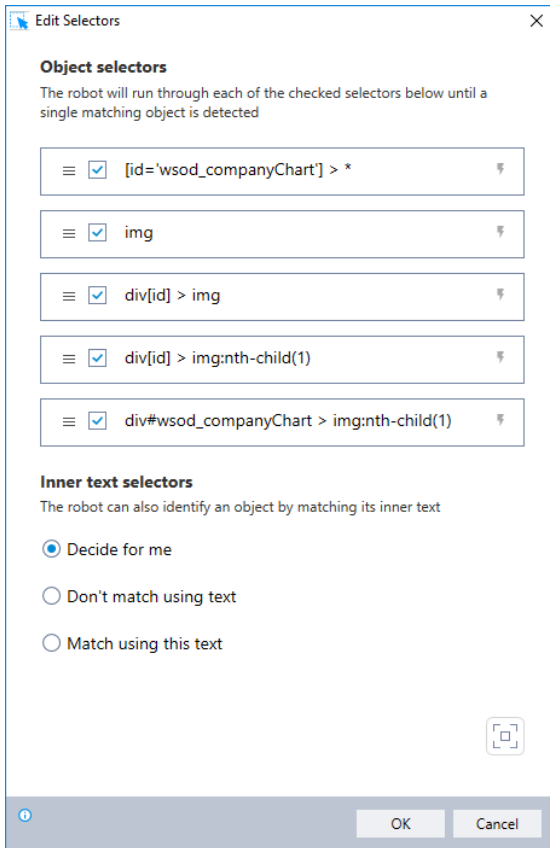
	Expands your selection to the next higher object (the "parent object") in the HTML hierarchy, if one exists
	Narrows your selection to the next lower object (the "child object") in the HTML hierarchy, if it one exists
	Briefly highlights the selected object in the browser
Edit Selectors	Opens a dialog 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


6. 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

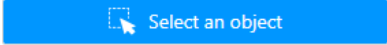
The  button invokes a dialog for directly editing two types of selectors used by the robot to identify the relevant object on the webpage:

- [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:

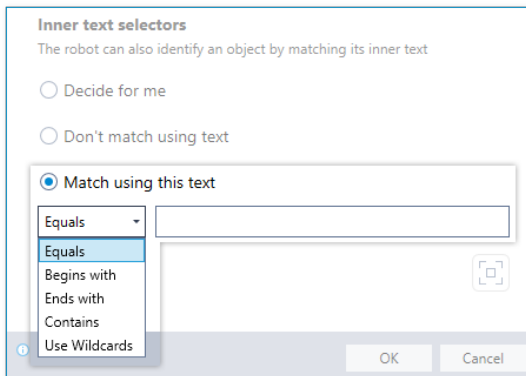
- **Select/deselect selectors:** tick/untick the checkbox of any selector to choose whether it should or should not be used in identifying the relevant object
- **Change the order in which the selectors will be processed:** reorder the selectors by dragging them into the desired order using the  icon 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



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

Use Wildcards

OK Cancel

Using an anchor object



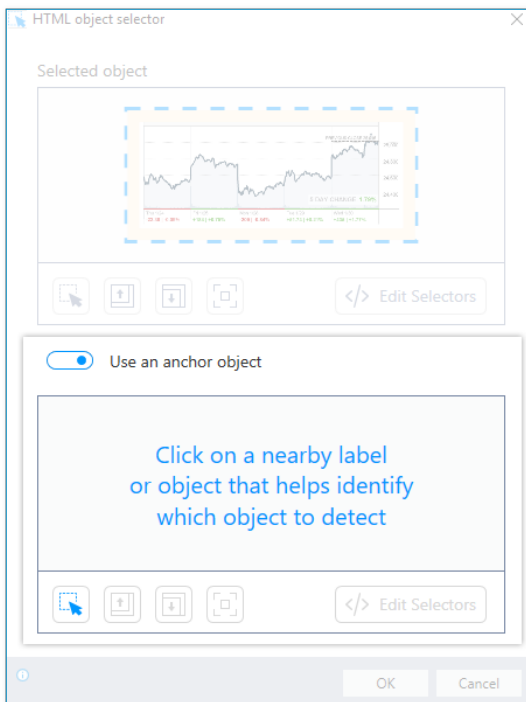
NOTE

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.

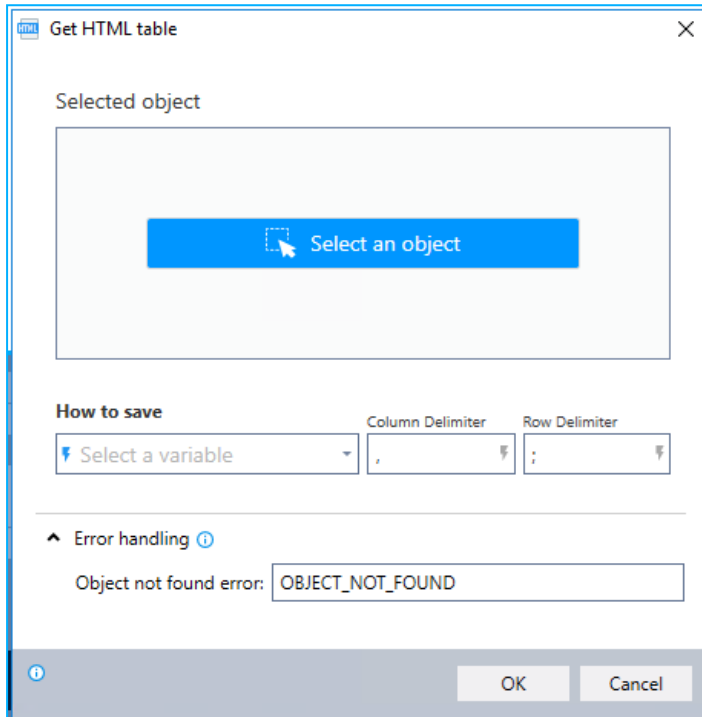


Get HTML Table

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

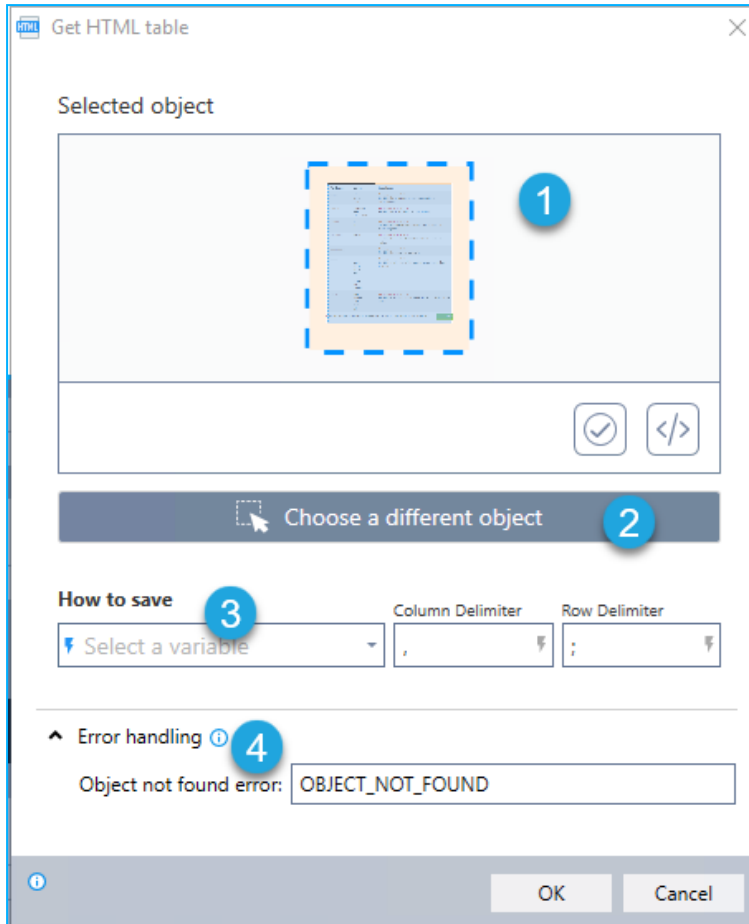
- a. Click **Select an object** to select the table whose text you would like to copy.



NOTE

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

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



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](#).



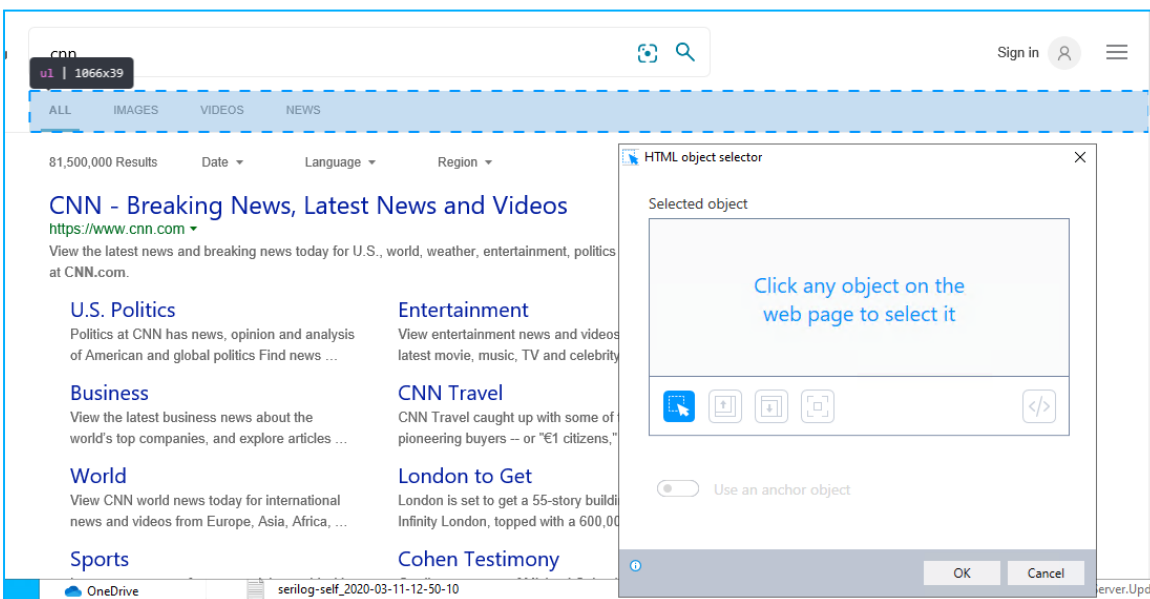
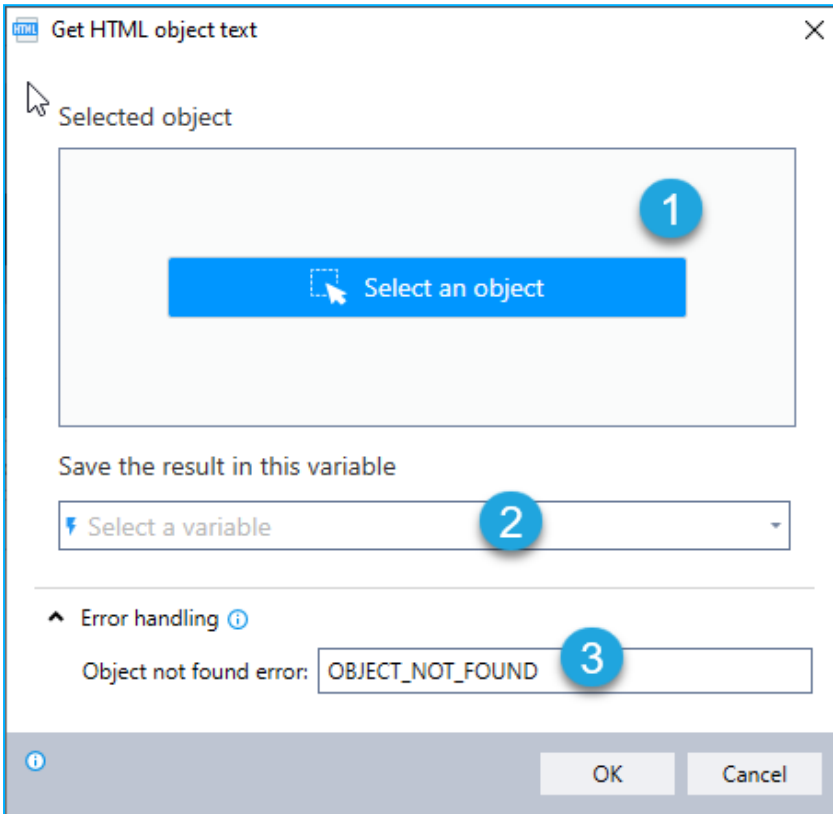
NOTE

This command has a built-in timeout monitoring of 5 seconds. [Read more](#) about Timeout Monitoring.

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



Edit Selectors

Object selectors
The robot will run through each of the checked selectors below until a single matching object is detected

- A
- a
- ol[id] > li > h2 > a
- ol[id] > li:nth-child(1) > h2:nth-child(1) > a:nth-chi
- ol#b_results > li:nth-child(1) > h2:nth-child(1) > a:

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

OK Cancel

1. Click **Select an object** to select the object whose text you would like to copy. Click again to restart the selection process if you need to select a different object.
2. Enter the name of the variable into which you would like to place the text of the selected object.
3. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).



NOTE

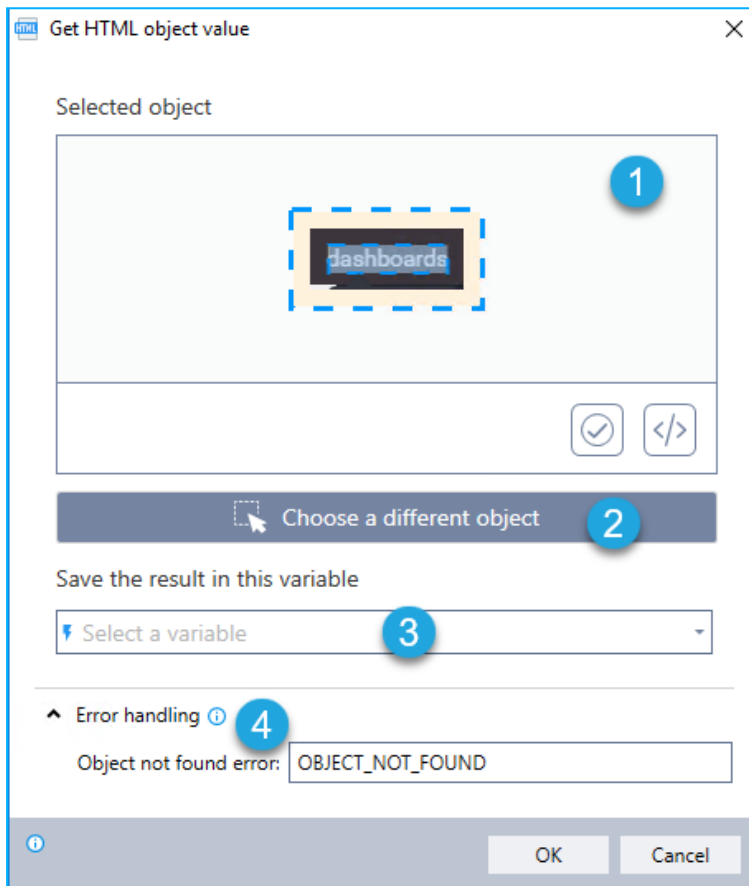
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

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

- a. Click **Select an object** to select the object whose value you would like to retrieve.
- b. 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](#).



NOTE

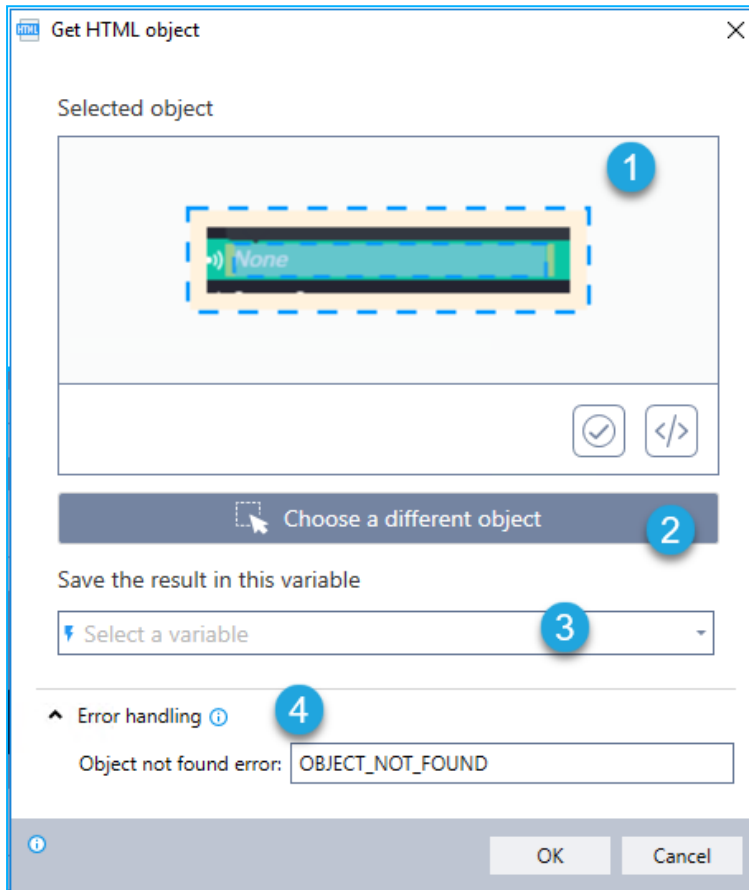
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

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

- a. Click the **Select an object** to select an object whose code you would like to retrieve.
- b. 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](#).



NOTE

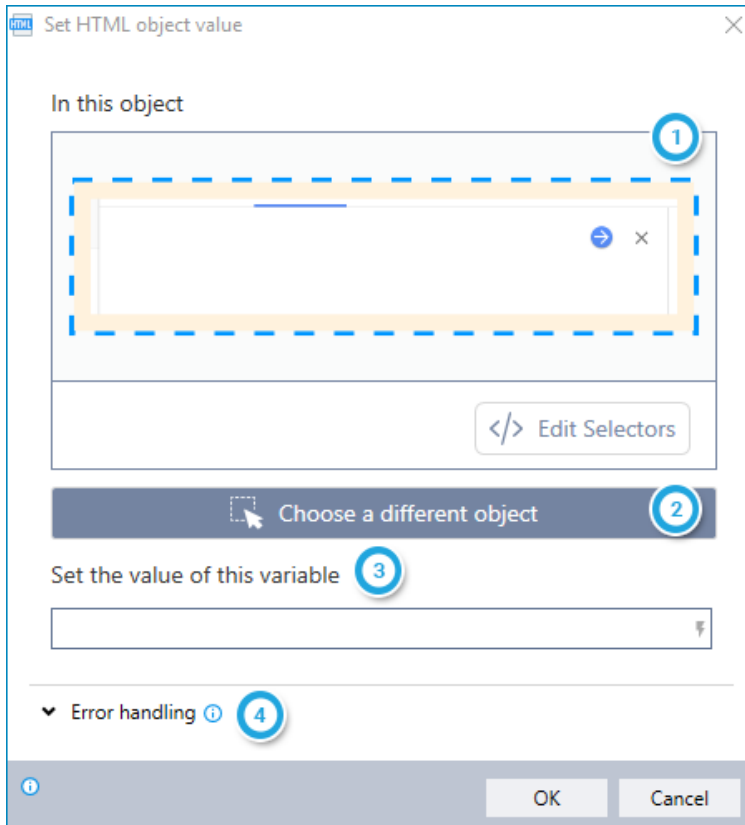
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.





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](#).



NOTE

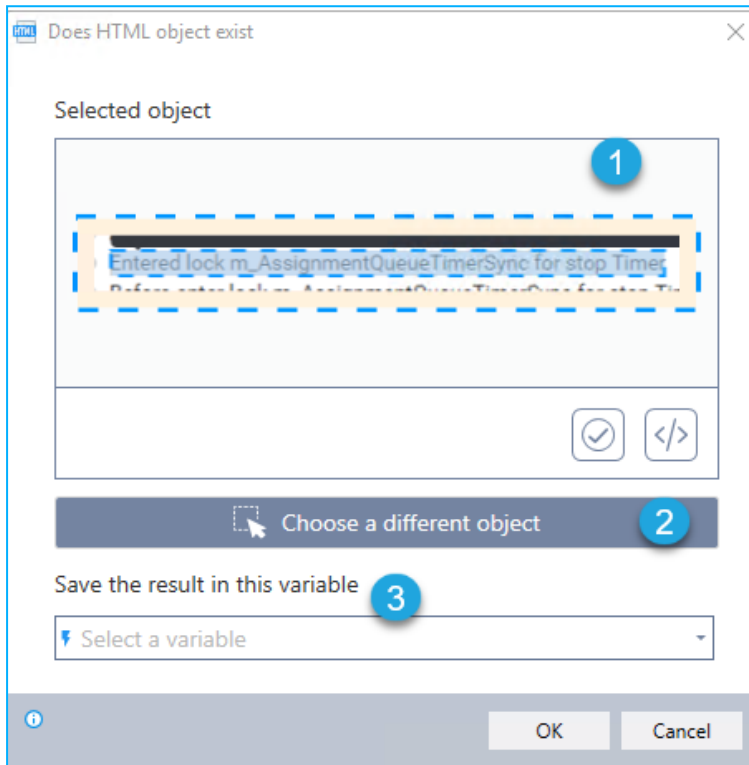
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

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

- a. Click the **Select an object** to select an object whose existence you would like to check.
- b. 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.



NOTE

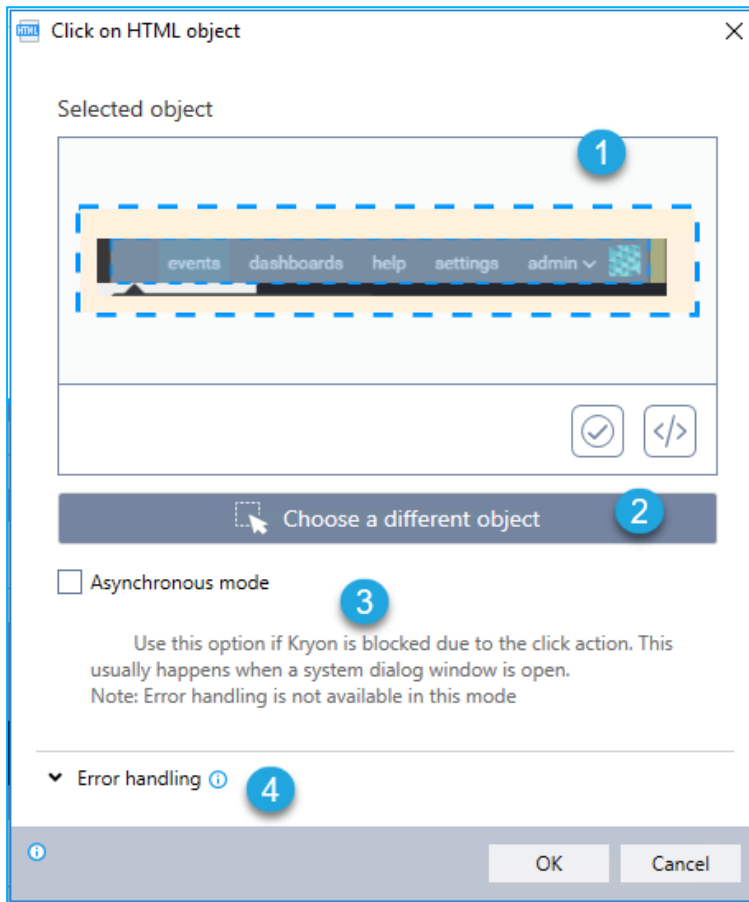
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

Click on HTML Object

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

Using the **CLICK ON HTML OBJECT** command

- a. Click the **Select an object** to invoke the **HTML OBJECT SELECTOR**.
- b. 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](#).



NOTE

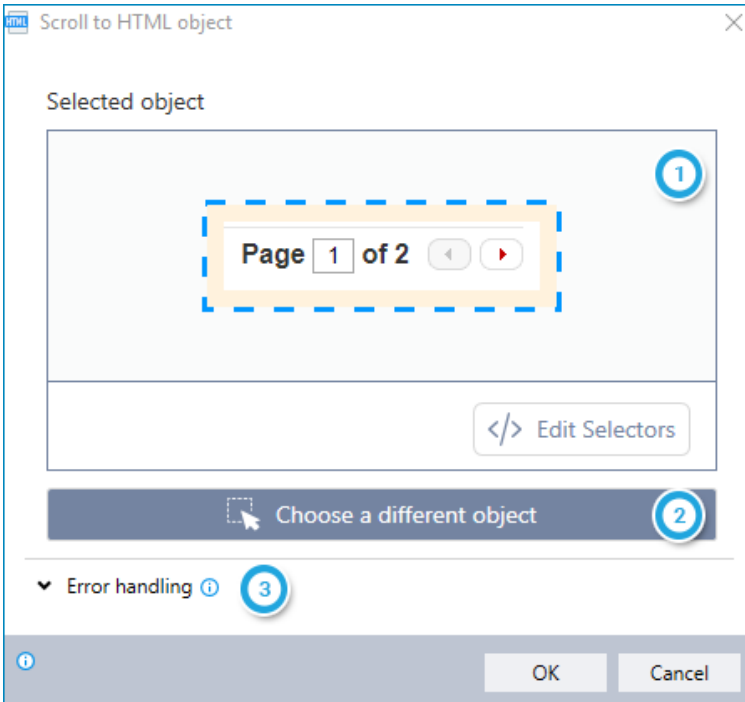
This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.




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](#).



NOTE

This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

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!)

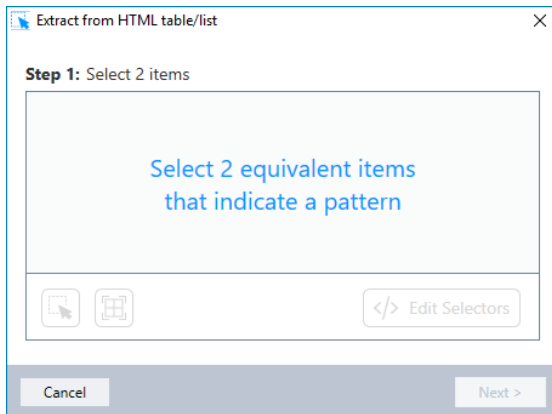
In this case:

- 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**

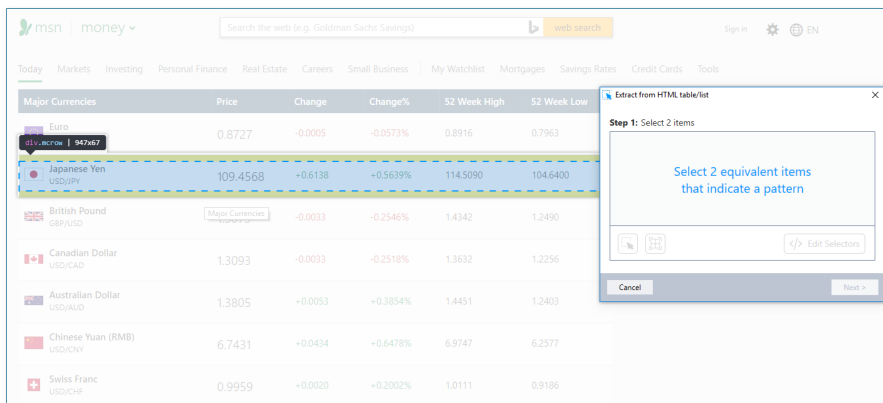
Selecting items

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 **Select item** 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.

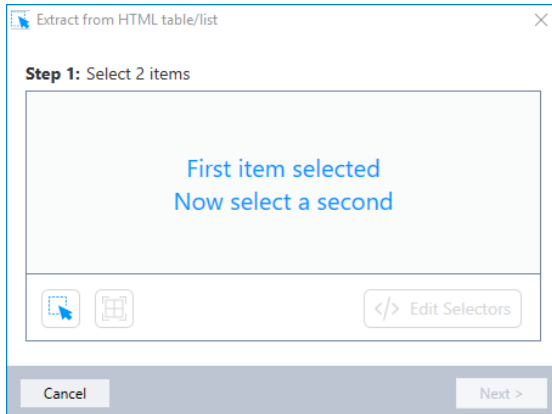




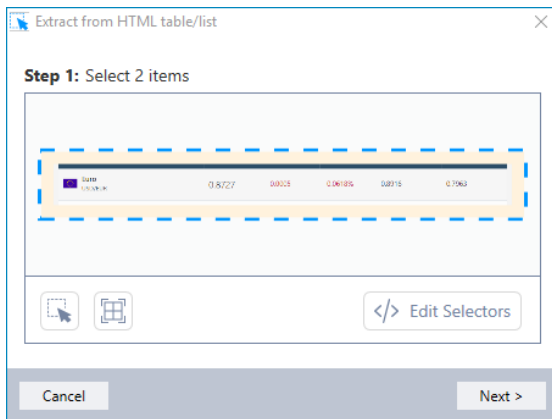
TIP

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.


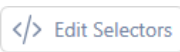
4. Click on the object that represents an item in the table you want to work with.
5. You will be prompted to select a second (equivalent) item.



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



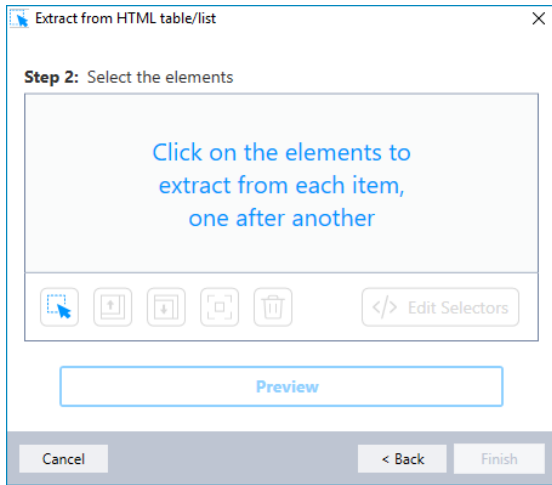
The following additional options will be enabled, allowing you to work the your item selection:

	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

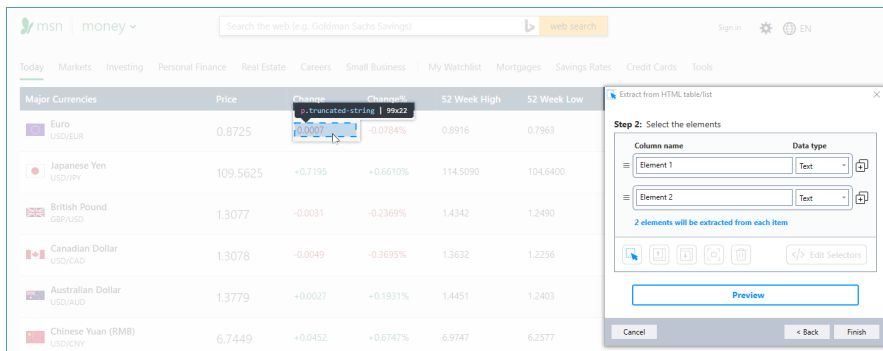
7. When you are satisfied with your selection, click **Next** to move on to selecting elements

Selecting elements





1. 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.




2. 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:



3. 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.



The screenshot shows a dialog box titled "Extract from HTML table/list" with a close button (X) in the top right corner. The main heading is "Step 2: Select the elements". Below this is a table with two columns: "Column name" and "Data type". There are four rows of elements:


Column name	Data type
Currency	Text
Price	Text
Element 3	Text
Element 4	Text

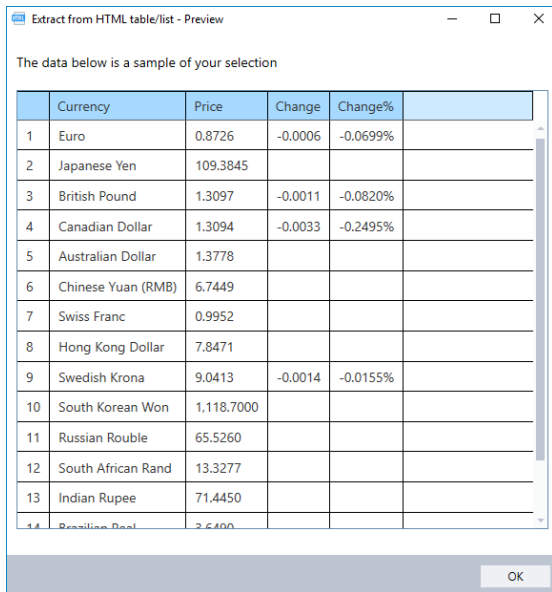
Each row has a menu icon (three horizontal lines) on the left and a plus icon in a square on the right. Below the table, it says "4 elements will be extracted from each item". At the bottom of the dialog, there are icons for moving elements (up, down, left, right) and a trash icon, followed by a button labeled "</> Edit Selectors". A large blue "Preview" button is centered below these icons. At the very bottom, there are "Cancel", "< Back", and "Finish" buttons.

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

4. Click on the  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

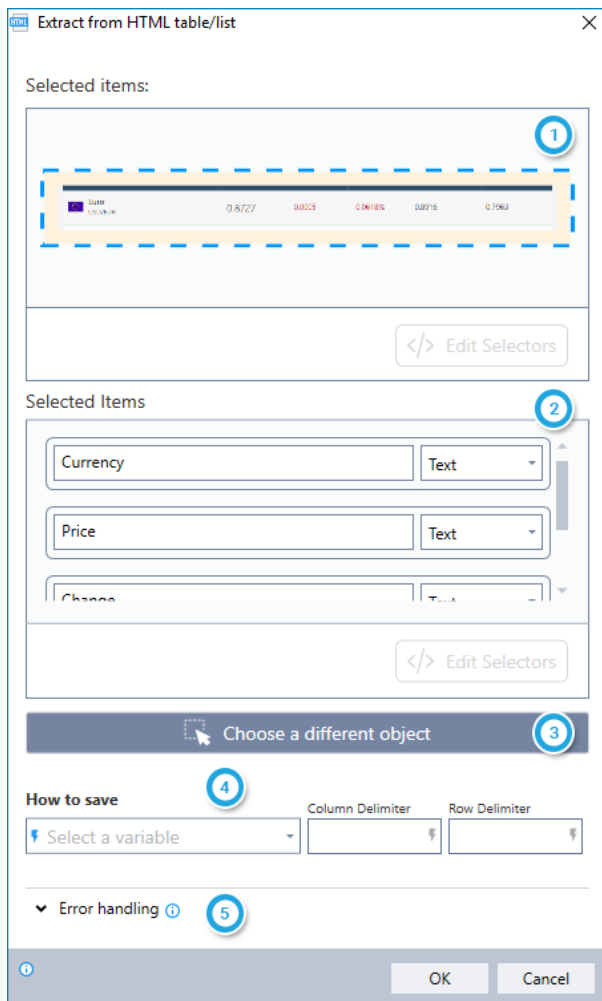
	Currency	Price	Change	Change%	
1	Euro	0.8726	-0.0006	-0.0699%	
2	Japanese Yen	109.3845			
3	British Pound	1.3097	-0.0011	-0.0820%	
4	Canadian Dollar	1.3094	-0.0033	-0.2495%	
5	Australian Dollar	1.3778			
6	Chinese Yuan (RMB)	6.7449			
7	Swiss Franc	0.9952			
8	Hong Kong Dollar	7.8471			
9	Swedish Krona	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	2.6400			

OK

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

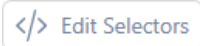
Finalize command options

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

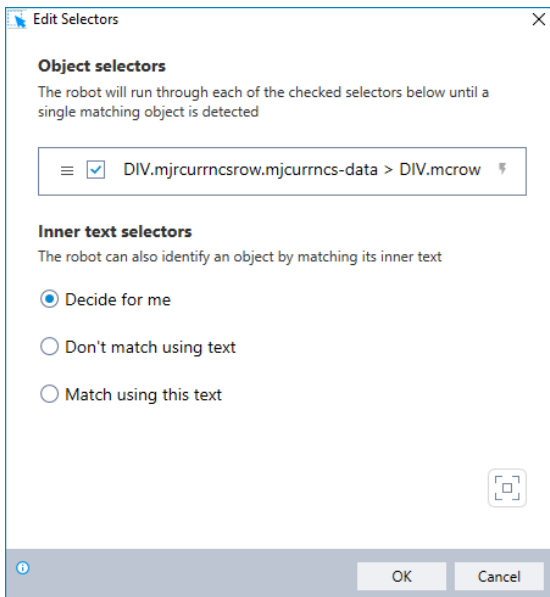



- 1 Preview a thumbnail image of a typical selected item
- 2 Review the list of elements that will be extracted from each item
- 3 Click to restart the selection process (if necessary)
- 4 Enter the name of the variable into which you would like to place the extracted data; **and** the delimiters to 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



NOTE

This command has a built-in timeout monitoring of 5 seconds.
[Read more](#) about Timeout Monitoring.

CHAPTER 22: .NET Automation Commands

In this chapter:

Get .NET Object Text	406
Get .NET Object Value	407
Get .NET Object Location	408
Set .NET Object Value	410



NOTE

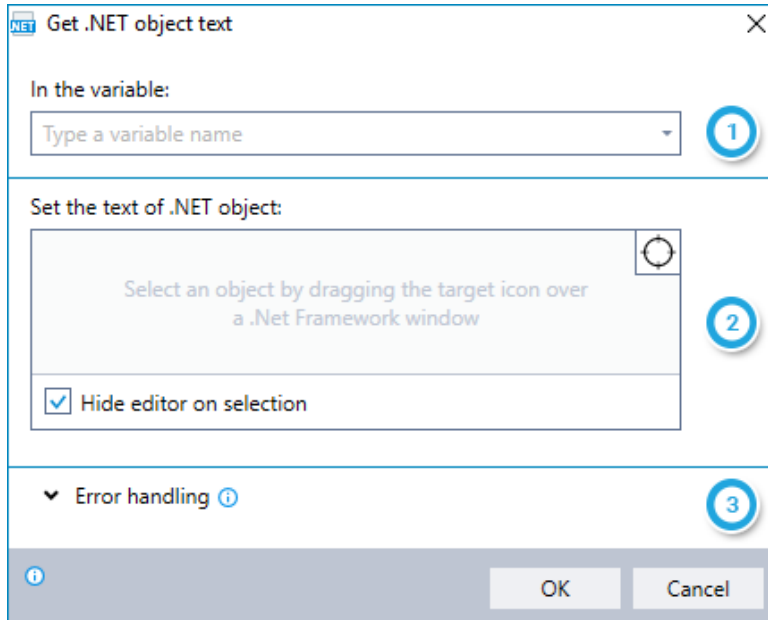
Use in recorded steps only





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

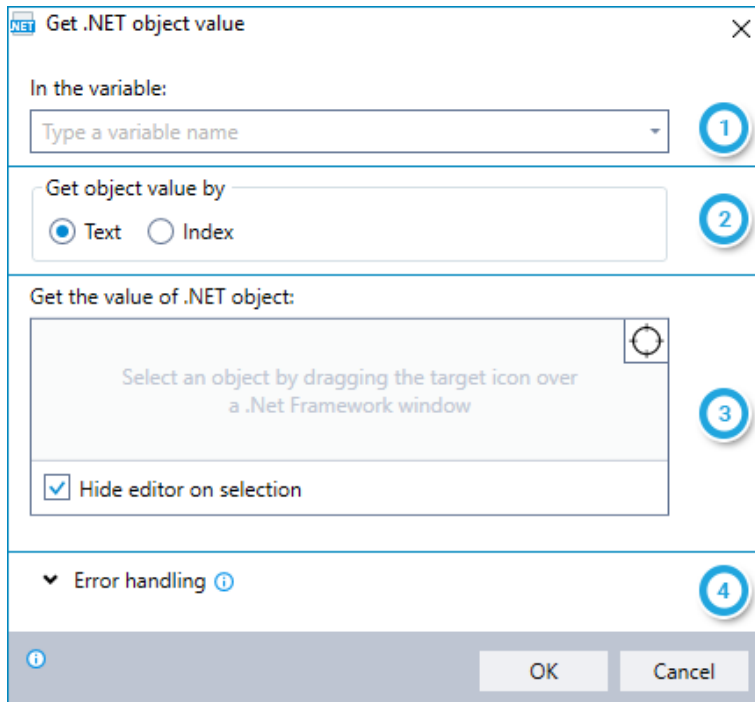




- 1 Enter the name of the variable into which you would like to place the text of the selected object
- 2 Select the object whose text you would like to copy by dragging the  icon onto the object in a .NET Framework window
 - Indicate whether you would like to hide Kryon Studio while you are selecting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

3

Select the object whose location you would like to retrieve by dragging the  icon onto the object in a .NET Framework window

- Indicate whether you would like to hide 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 location of the selected object

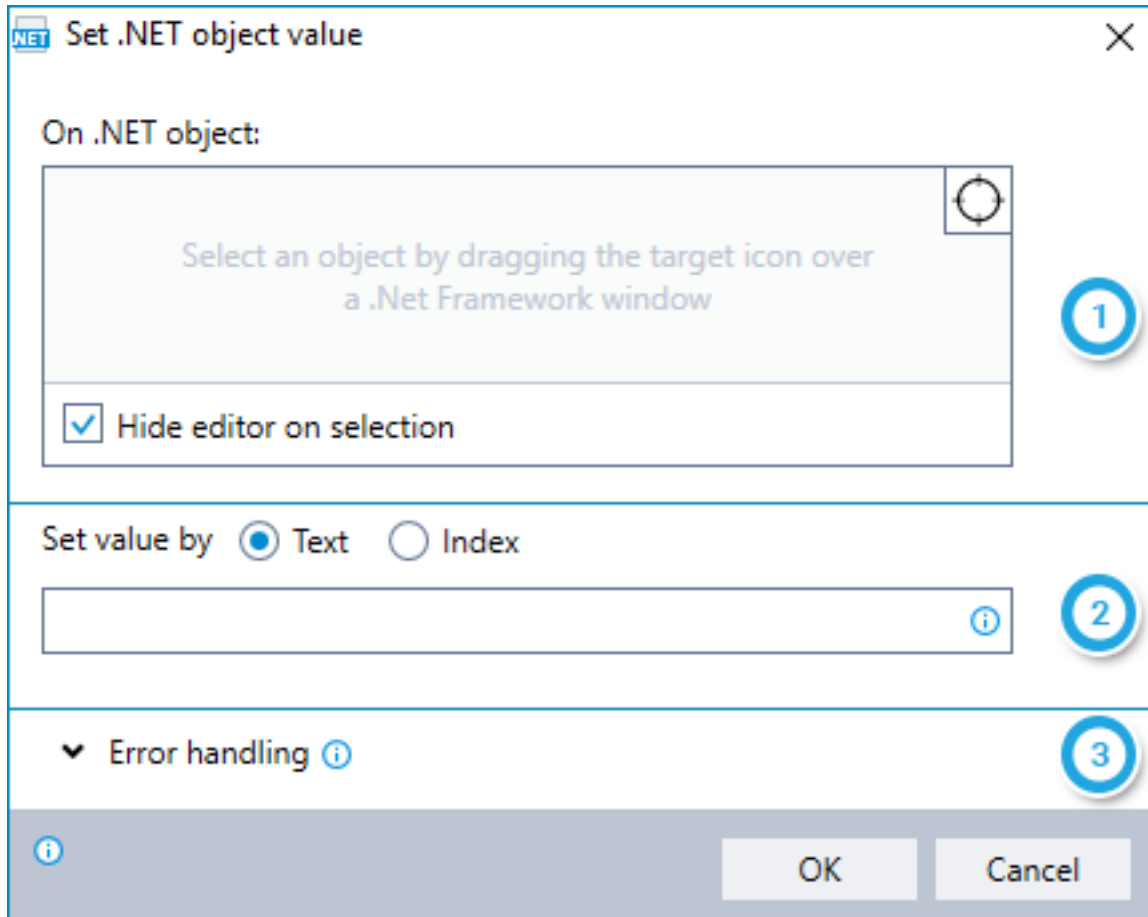
4



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

To be able to use Java Advanced commands, you need to have:

- Java Bridge installed
See [Installing Java Bridge](#) to make sure you have it configured right.
- VC++ 2010 installed

If not installed, you might encounter the following error when executing `jabswitch.exe`:

"The code execution cannot proceed because MSVCR100.dll was not found."

Note: Use JAVA AUTOMATION COMMANDS within recorded steps only. They are not supported for steps that are comprised solely of Advanced Commands.

Note: When using the selector in the commands, highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

Support

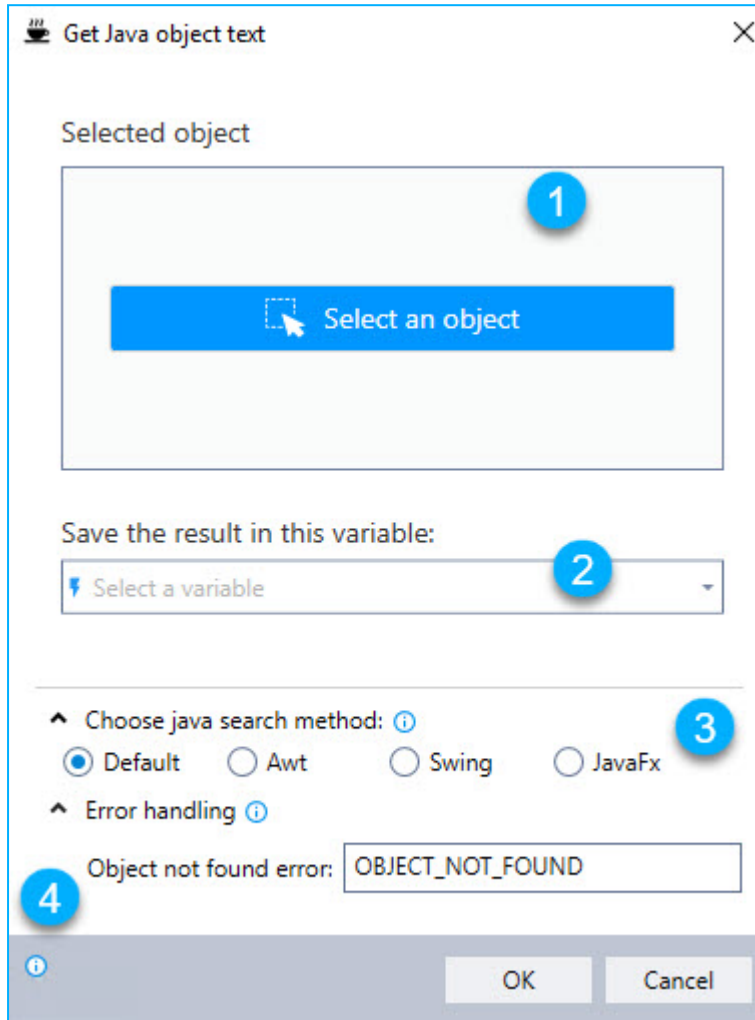
Supported Java runtime version	Java v1.2 and later Note: The advanced command "Click on Java Object" isn't supported in Java 1.2. Workaround: 1. Use the "Get Java Location" command to get the location coordinates. 2. Add the "Click mouse" command and insert the coordinates in it.
Supported Java GUI framework	Swing JavaFX Applet (AWT)
Supported Java controls	Label Buttons Checkboxes Radio buttons Combo boxes Text boxes

	Text areas
	Panel
	Headers
	Tabs
	Table
	Tree
	Menu item

Get Java Object Text

Retrieve the text of an object in the active Java application and place it into a new or existing variable.

Using the GET JAVA OBJECT TEXT command



1. Select the Java object whose text you would like to retrieve.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



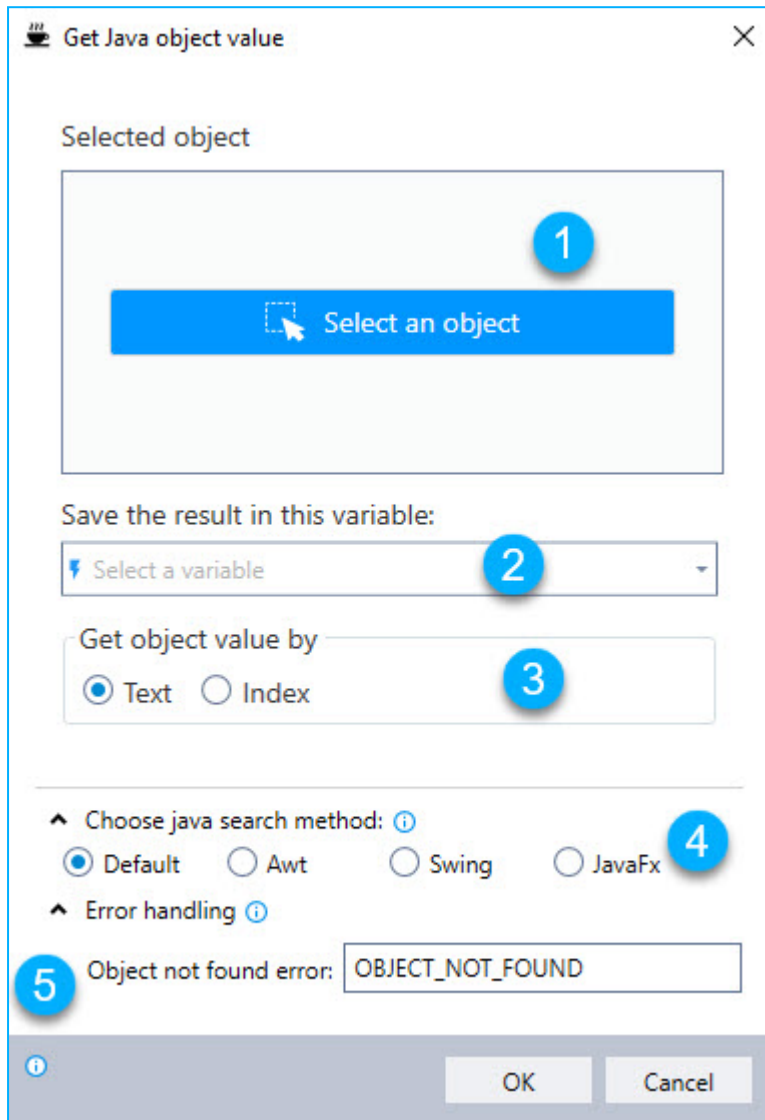
Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

2. Enter the name of the variable into which you would like to place 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 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 JAVA OBJECT VALUE command



1. Select the Java object whose value you would like to retrieve. After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection

- Tree view



Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

2. Enter the name of the variable into which you would like to place the value of the selected object.
3. Choose whether to retrieve the object value by text or by index.
4. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
5. 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 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

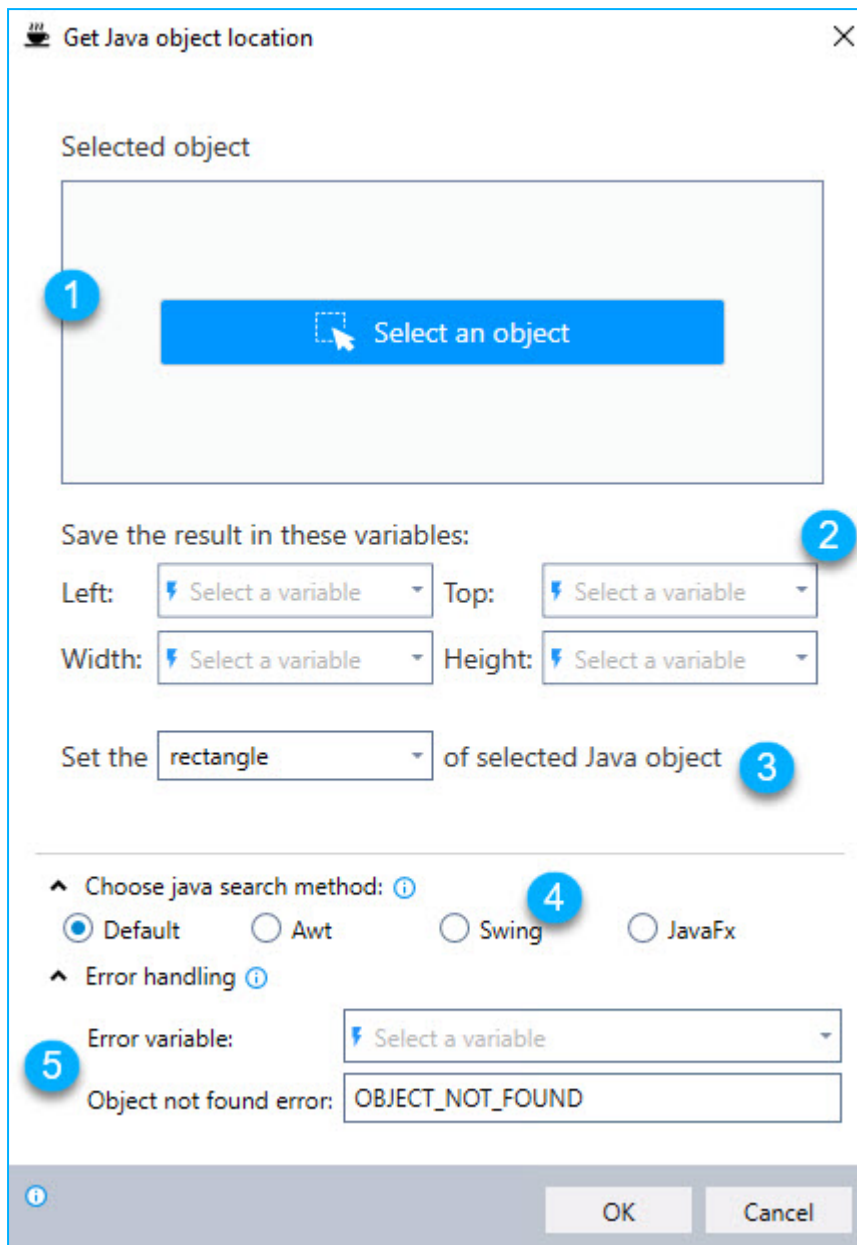


TIP

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 JAVA OBJECT LOCATION command



1. Choose whether to retrieve the object location either by rectangle or by center point.
2. Enter the name of the variable into which you would like to place the location of the selected object.
3. Select the Java object whose location you would like to retrieve.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection

- Tree view



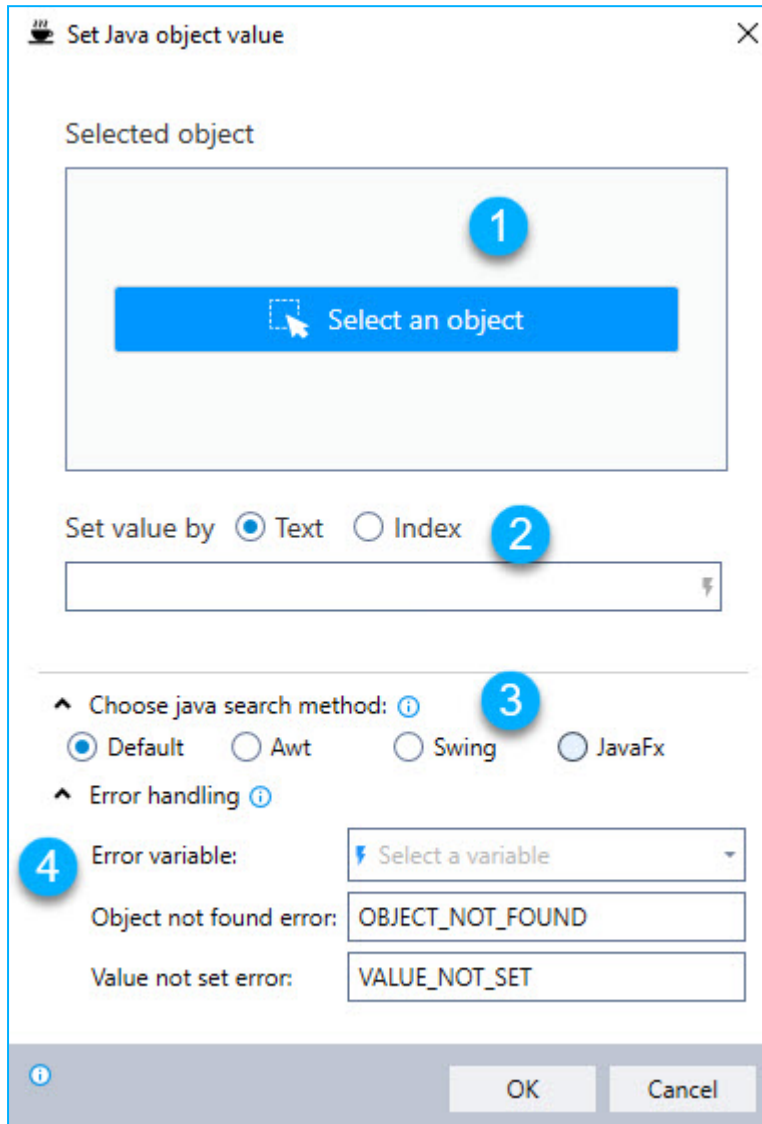
Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

4. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
5. 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 application .

Using the SET JAVA OBJECT VALUE command



1. Select the Java object whose value you would like to set.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view

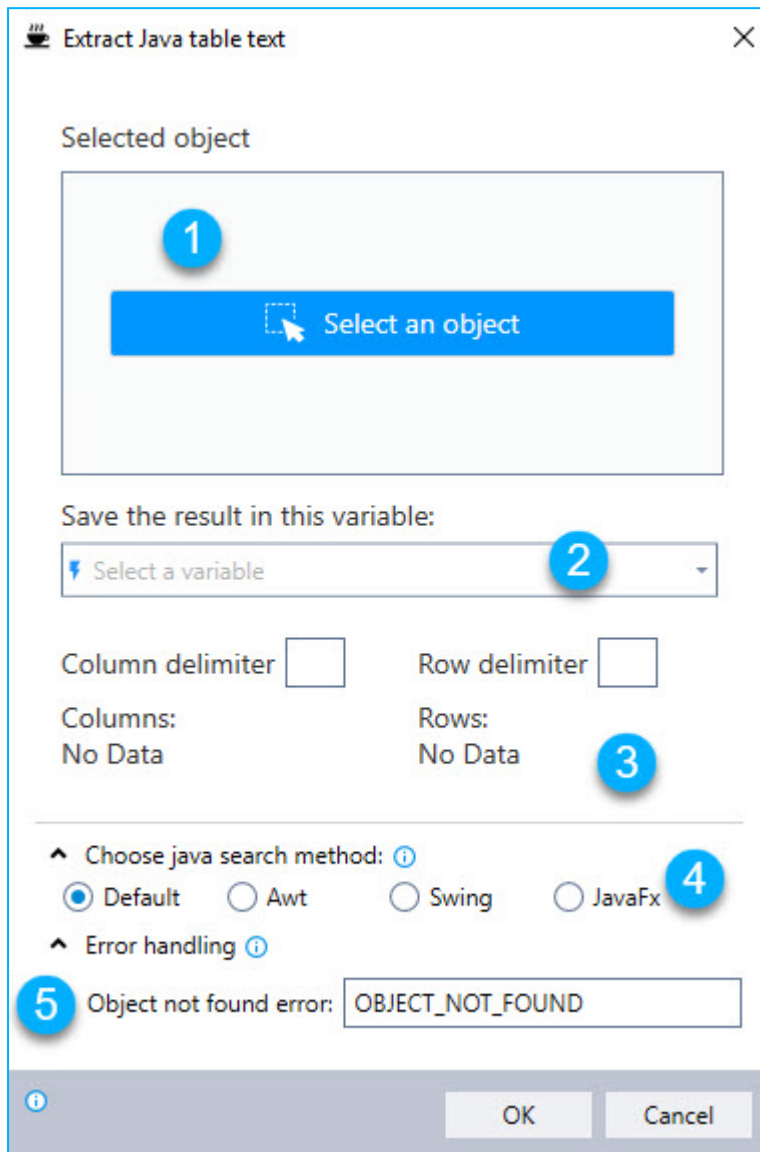


Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

2. Choose whether to place the value by text or index; **and** Enter the value you want to place (can be free text or copied from values stored in variables).
3. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
4. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

Extract Java Table Text

Extract the text fields from the Java table into a variable. You can choose the rows and columns you want to extract.



1. Select the Java table whose text fields you want to extract. After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view

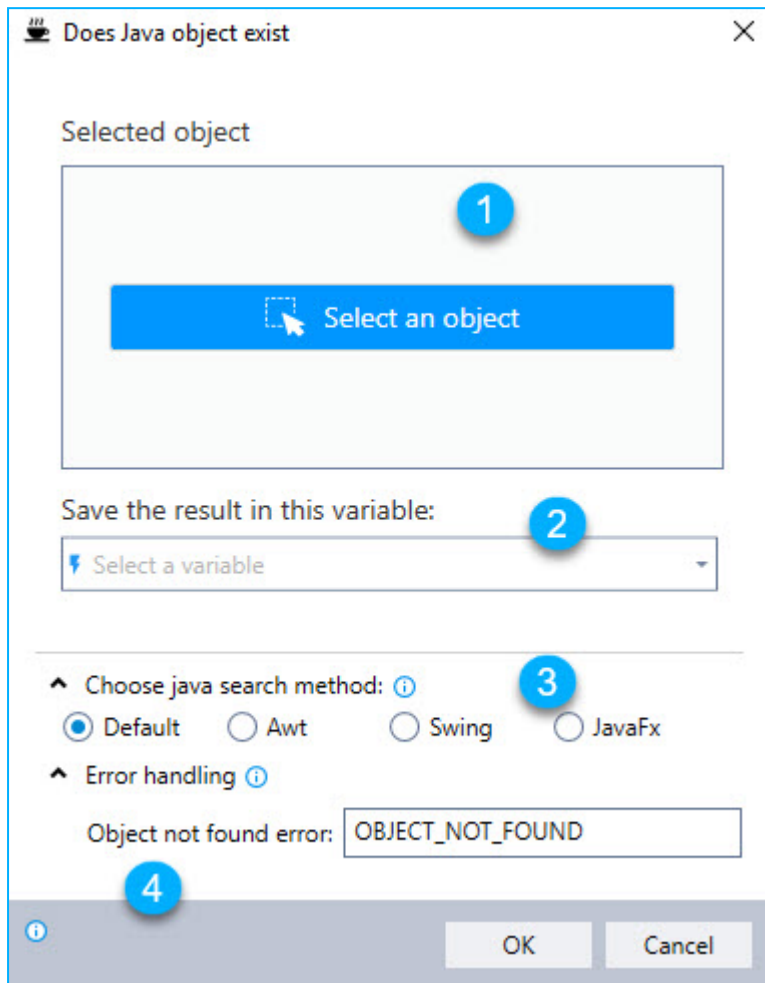


Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

2. Enter the name of the variable into which you want to save the extracted text.
3. Set the delimiters of the relevant column and row from which you want to extract the text.
4. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
5. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

Does Java Object Exist

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



1. Select the Java object whose existence you want to check.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object
 - Use right click for selection
 - Tree view



Note: Highlighting items with blue while hovering them is set to be off by default. To

change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to `"true"` in the `appsettings.config` file of the Studio.

2. Enter the name of the variable into which you want to save the result of the query (TRUE/FALSE).
3. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
4. Instruct the wizard how to handle any errors encountered. Read more about [Error Handling](#).

Click on Java Object

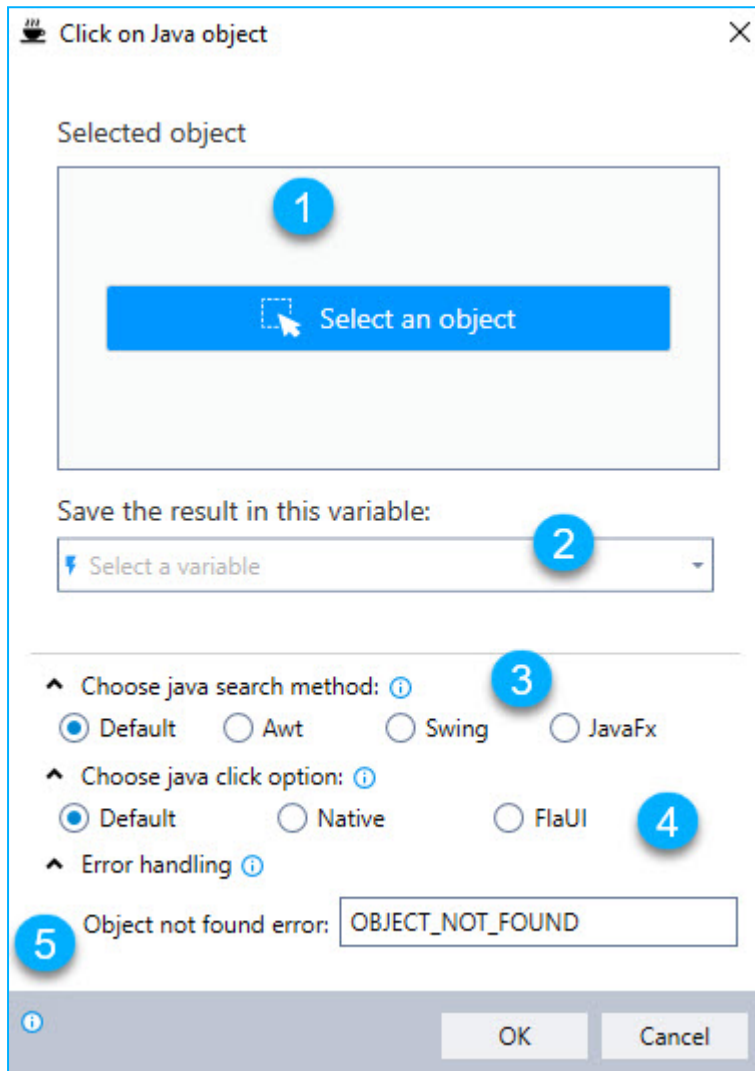
Click on an object in the currently active Java application. This enables you to imitate a mouse click on a specific component in the Java application.

- **Note:**

The advanced command "Click on Java Object" isn't supported in Java 1.2.

Workaround:

1. Use the "Get Java Location" command to get the location coordinates.
2. Add the "Click mouse" command and insert the coordinates in it.



1. Select the Java object you want to click.
After selecting the object, the following buttons become available (when relevant):
 - Click an object
 - Expand selection
 - Narrow selection
 - Highlight selected object

- Use right click for selection
- Tree view



Note: Highlighting items with blue while hovering them is set to be off by default. To change that, change the key value `<add key="FeatureFlag_JavaHighlight" value="false" />` to "true" in the `appsettings.config` file of the Studio.

2. Enter the name of the variable into which you want to save the result.
3. Select the Java search method by platform (Default, Awt, Swing, or JavaFx).
4. Select the Java click option (Default, Native, or FlaUI)- enabling capturing elements using right click.
5. 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	430
Get Global Variable	431
Delete Global Variable	432

Set Global Variable

Place a value into a variable that is available for use in other wizards and sensors (a "global variable"), by either:

- Creating a new global variable and setting its value; or
- Setting the value of an global existing variable

Using the SET GLOBAL VARIABLE command

- 1 Enter the name of the global variable (new or existing) to which you want to assign a value
 - If you want to create a new global variable, type the name of the new variable
 - If the global variable already exists, choose its name from the drop-down list
- 2 Set the value of the global variable you have specified
 - You can include free text and/or values copied from different variables
 - <Enter> <Space> and/or <Tab> can be used



TIP

Turn something standard into something global...

Very often, the value set into a global variable is actually the value of one of the current wizard's "standard" variables. Just type the standard variable's name between dollar signs (e.g., \$MyVar\$) in step **2** to make this happen.

Get Global Variable

Retrieve the value of a global variable (previously created in a different wizard using the **SET GLOBAL VARIABLE** command) and place it into a standard variable in the current wizard.



NOTE

In order to use a global variable in the current wizard, its value must first be placed into a standard variable.

Using the GET GLOBAL VARIABLE command

The screenshot shows a dialog box titled "Get global variable" with a close button in the top right corner. The dialog is divided into two main sections. The first section, "In the variable:", features a dropdown menu with the placeholder text "Type a variable name" and a blue circular icon with the number "1" to its right. The second section, "Set the global variable value:", also features a dropdown menu with the placeholder text "Type a variable name" and a blue circular icon with the number "2" to its right. 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

The screenshot shows a dialog box titled "Delete global variable" with a close button in the top right corner. Below the title bar, the text "Delete the global variable:" is displayed above a text input field containing the placeholder "Type a variable name". To the right of the input field is a blue circular icon with the number "1". At the bottom left of the dialog is an information icon, and at the bottom right are "OK" and "Cancel" buttons.



Enter the name of the global variable you would like to delete

CHAPTER 25: Scripting Commands

In this chapter:

Note	434
Notes for Mobile/Web Access	435
View Variable List	436
Show Debug Message	438

Note

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



TIP

Just do it.

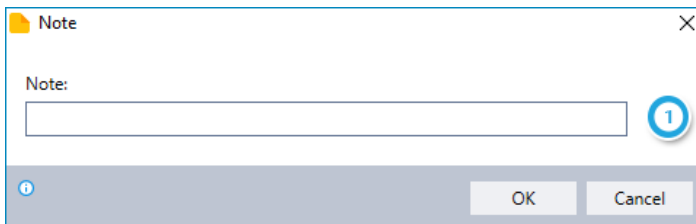
Anyone who has ever developed code knows how important it is to document it internally. But all too often, in the rush of getting things done, this crucial practice is overlooked. Don't let it happen to you!

Enter notes to document various sections of the wizard and its logical flow. You'll be surprised how many hours and headaches it will save when you (or someone else) is revising, updating, or debugging.

A few examples:

- *Fallback for when the date is empty*
- *This is the end point of a loop*
- *Ensures that the wizard will continue even if the file can't be found*

Using the NOTE command



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

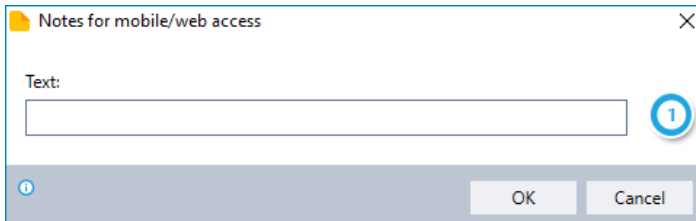
- The note will appear only in the Advanced Commands Editor, so end users will never see it when the wizard/sensor is run

Notes for Mobile/Web Access

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

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

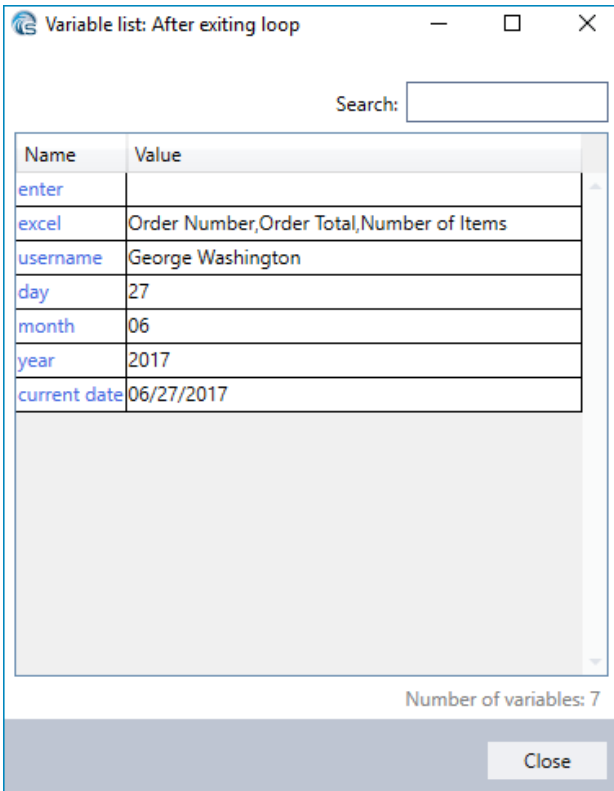
Using the NOTES FOR MOBILE/WEB ACCESS command



- 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:

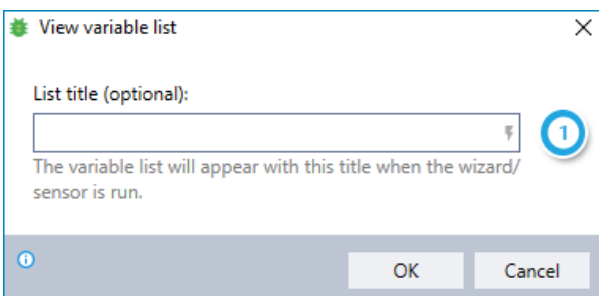


CAUTION

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



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

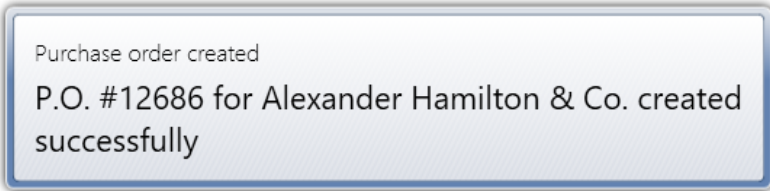


TIP

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

Show Debug Message

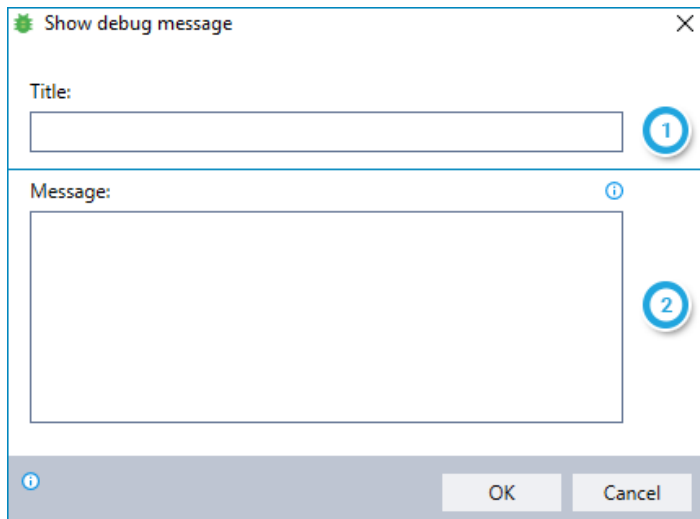
Display a text message to the Kryon Studio user when the wizard is run in debug mode. Here's a sample:



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



1

Enter the title of the message

2

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\$)

APPENDICES

APPENDIX A: Timeout Monitoring Support for Advanced Commands

By default, the Advanced Commands are designed to run until process completion. However, if the process is interrupted, the execution of the command might enter an endless loop state. Process interruptions can be caused by various reasons, such as browser crashes, application 'Not Responding', and blocking by another application.

You can prevent such endless loop states and gain a better control over the process by enabling the Timeout Monitoring setting for the supporting advanced commands.

Monitoring Types

- Built-in (hard-coded; fixed)
- Manually enabled through the settings of the advanced command in Kryon Studio

See [Availability](#)



NOTE

For advanced commands, you can control and configure *only* the timeout monitoring that can be manually enabled.

Configuration

To enable Timeout Monitoring for an Advanced Command:

1. Open the **Advanced Commands** view > Double click or drag the relevant **Advanced Command** > Expand **Error handling**.
2. Check "Enable timeout monitoring".
3. Set the amount of time in seconds (0-9999) for the process to raise an error and terminate. The default value is 1800 sec (30 min).
4. Set the error text message that raised upon termination.

The default value is `TIMEOUT_ERROR`.

Enable timeout monitoring
 Raise error after: seconds
 Timeout error text:
 [OK] [Cancel]

Timeout Monitoring Availability

Category	Advanced Command	Monitoring Type
Excel Commands	<ul style="list-style-type: none"> • Convert Excel to CSV • Query from Excel • Copy from Excel • Paste to Excel • Delete from Excel • Create new excel file • Excel Worksheet Actions • Excel Column Actions • Excel Row Actions • Run Macro 	Manually enabled
Window Commands	Control Window State	Manually enabled
External Program Commands	Run Javascript on Page	Built-in: <ul style="list-style-type: none"> ■ 5 seconds in FireFox/Chrome ■ 15 seconds in Internet Explorer
HTML Commands	<ul style="list-style-type: none"> • 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 	Built-in: <ul style="list-style-type: none"> ■ 5 seconds in Internet Explorer/Chrome/FireFox

- Extract from HTML table/list

APPENDIX B: Timeout Monitoring for Click-Blocked

By default, if the Click core action cannot be executed (the click is blocked for any reason) after 4 click attempts, the Wizard proceeds to the next core action (usually a fallback). You can change the default click-timeout configuration through the `appSettings.config` file.

[To change the timeout configuration for the Click core action:](#)

Add the Key and Value below to the `appSettings.config` files of both Studio and Robot applications, and define the desired click number:

```
<add key="MaxRepeatStepsTimes" value="<click number>"/>
```

Example: `<add key="MaxRepeatStepsTimes" value="6">`

APPENDIX C: 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 the [Error handling](#) option:

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**

In some advanced commands you have the option to configure Timeout Monitoring.

See [Timeout Monitoring](#).

Error handling fields of Global Function

^ Error handling ⓘ

General error:

Global function not found:

Error variable:

ⓘ

APPENDIX D: Kryon Connector Extension

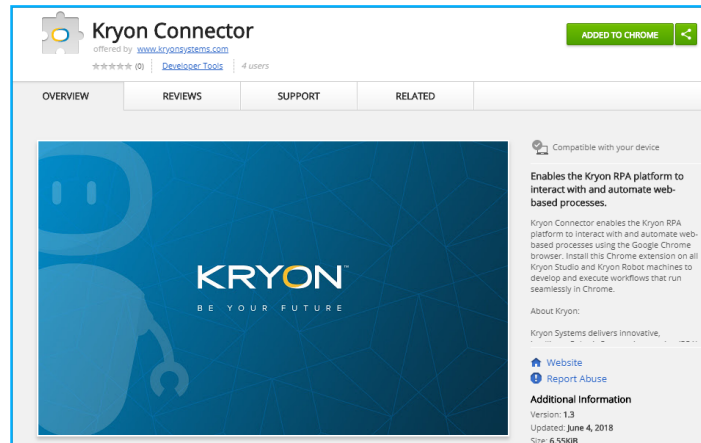
Install the [Kryon Connector](#) Chrome extension to use Chrome when developing wizards with these Advanced Commands:

[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 extension

Remove from Chrome

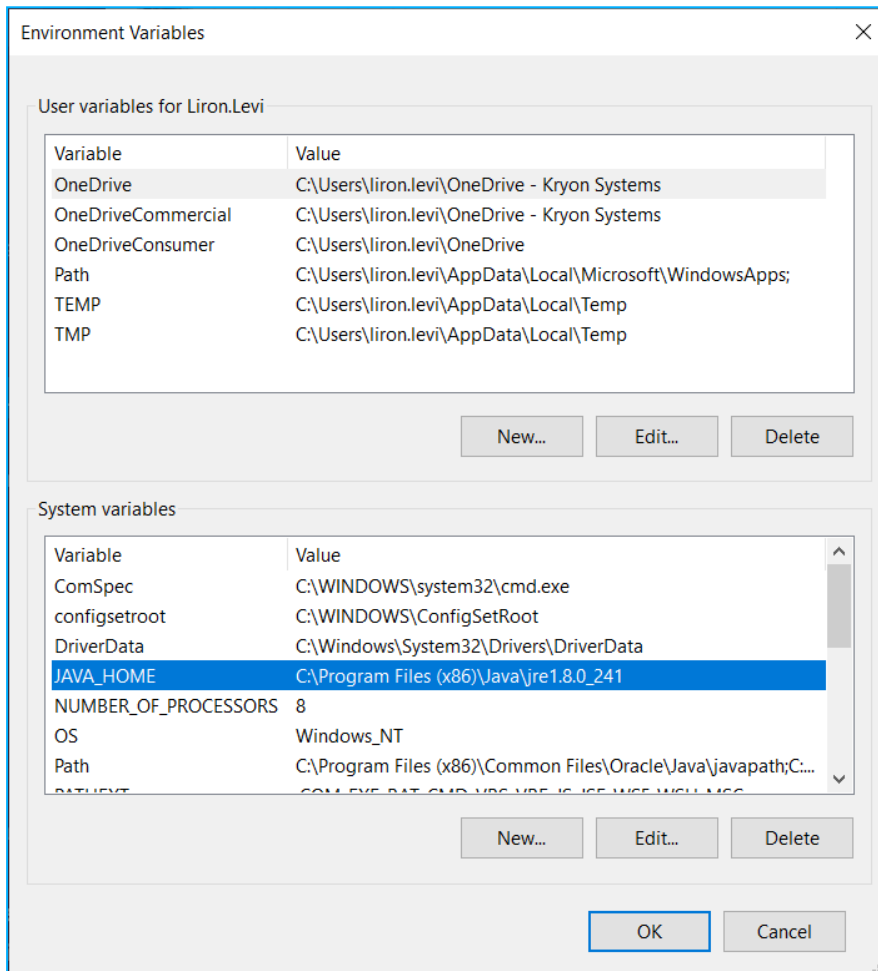
Kryon Connector is also available as a free download from the [Chrome Store](#).

APPENDIX E: Installing Kryon Java Bridge

Before running Studio and Robot installation package, you need to prepare the ground for Java Bridge and run the Java Bridge script.

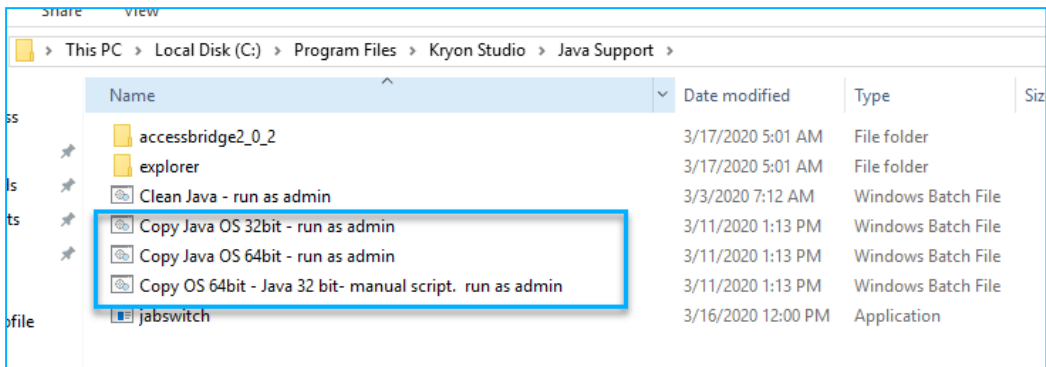
Java, Kryon Java Bridge, and JAVA_HOME are prerequisites to be able to use Java Automation Commands

1. Make sure you have Java installed. If not, [download and install Java](#) according to the Windows architecture (64 or 32 bit).
2. Make sure JAVA_HOME environment variable is configured for the installed Java:



3. **Install the Kryon Java Bridge** by running the Java Support Script (for Studio and Robot) *as admin* according to your architecture:
 - If you have 64bit architecture and you installed Java 64bit, run the script: "Copy Java OS 64bit".
 - If you have 32bit architecture and you installed Java 32bit, run the script: "Copy Java OS 32bit".

- If you have 64bit architecture and you installed Java 32bit, run the scripts: "Copy OS 64bit - Java 32 bit- manual scripts."
Note: Make sure to run the relevant scripts for both Studio and Robot; navigate to both Studio and Robot folders > Java Support folder > run the relevant script.



NOTE

Contact Kryon support team for any additional or customizable Java configurations.

[-Take me to the relevant step in the RPA Installation Guide-](#)

APPENDIX F: Java Supported Controls and Accessibility

Supported Controls Accessibility

Control / Java GUI Framework	Swing	JavaFX	Applet (AWT)
Label	Read	Read	Read
Buttons	Read	Read	Read
Checkboxes	Read, Write	Read, Write	Read, Write
Radio buttons	Read, Write only true (1)	Read, Write only true (1)	Read, Write only true (1)
Combo boxes	Read, Write (not always) (2)	Read, can't write (3)	Can't read, Write (4)
List boxes	Read, Write	Read, Write	Can't read, Write only by index (5)
Text boxes	Read, Write	Read, Write	Read, Write
Text areas	Read, Write	Read, Write	Read, Write
Panel headers	Read	Read	Non-existing control in platform
Tabs	Read	Read	Non-existing control in platform
Canvas (6)	-	-	Non-existing control in platform
Table	Read	Read, Write on cell	Non-existing control in platform
Tree	Read, Expand, Collapse by Index, Text	Read	Non-existing control in platform
Menu Item	Read	Read	Non-existing control in platform
Spinner	Non-existing control in platform	Read, can't write (7)	Non-existing control in platform

(1)Radio buttons can be checked by our tool if they are already unchecked, but cannot be unchecked. To uncheck radio button, you should check another one.

(2) Combo boxes can be set only in Swing framework, but it does not work for all controls. Java app may throw Exception to invalid combo box state. It is related to the internal behavior of Java Access Bridge, which sets selection index to -1 before setting value. In such cases double setting action might be needed, first one will set dropdown to empty value and second will set correct index.

(3) In JavaFX we can't set Combo box too, but it's value can be read.

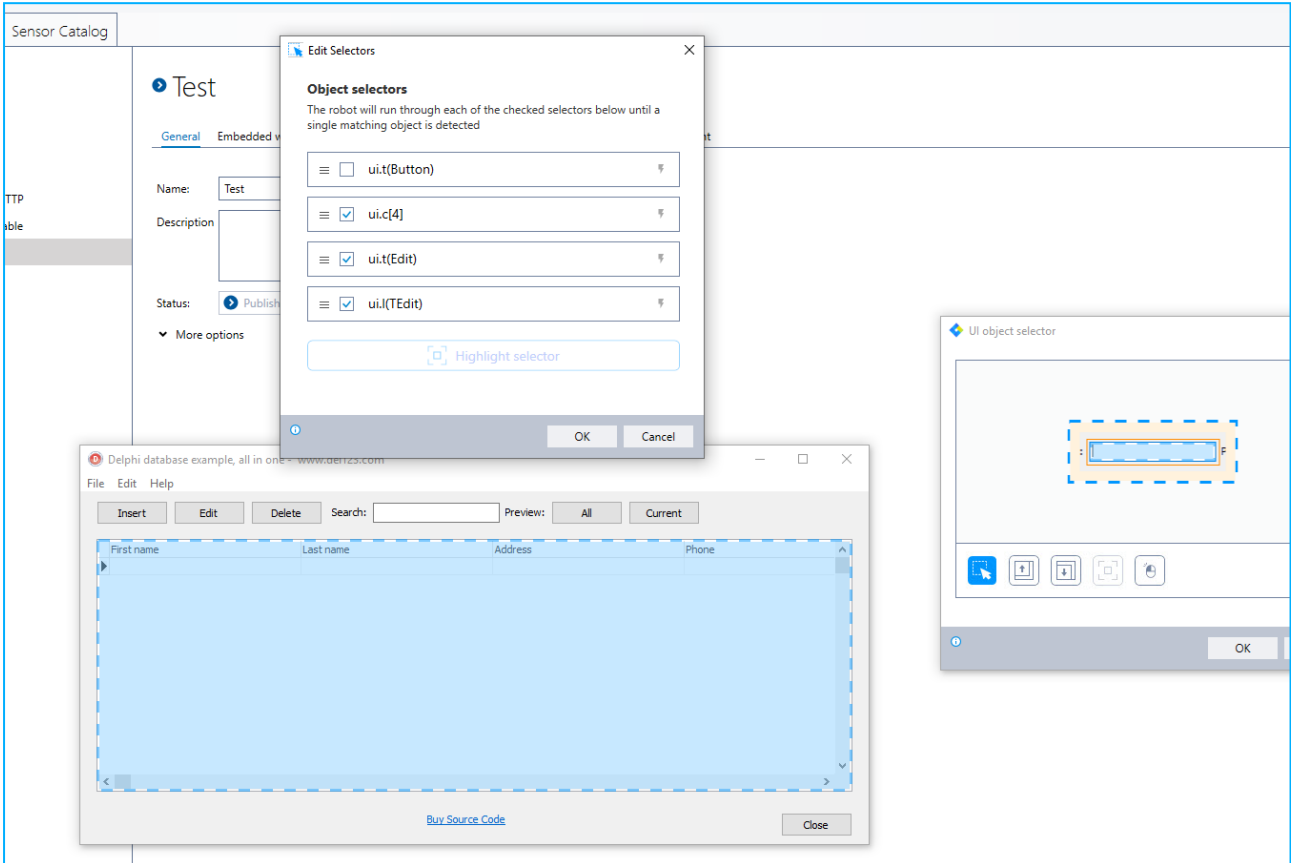
(4) In AWT there is a problem of reading combo box value, it displays random unreadable characters instead. That's why we can't set combo box by value in AWT, but setting by index works.

(5) List box in AWT can't be read and set by value, but can be set by index.

(6) JAB/FlaUI can't interact with Canvas at all, because it shows only Canvas Pane, but not internal objects (texts, labels, etc.).

(7) In JavaFX there is Spinner control, which can be read by our tool but can't be set.

APPENDIX G: Editing the UI Automation Selector Syntax



TO UNIQUELY IDENTIFY A CONTROL IN A PROGRAM (UI)

Note: The query must always start with [ui.]

Syntax	Purpose	Example	Behavior and logic
c [index]	Identifies control by index in parent	ui.c[3].c[5].c[0]	Selects window (ui.) > forth child (c[3]), sixth child (c[5]), and first child (c[0])
n (Name)	Identifies control by name	ui.n (Delete).c[5]	Selects window (ui.) > element with name "Delete" > element's sixth child
t (Type)	Identifies control by type	ui.t (Pane).n (BuyMe)	Selects window (ui.) > element with type "Pane" > control with name "BuyMe"
l (Classname)	Identifies control by className	ui.l (TDBPane)	Selects window (ui.) > element with className "TDBPane"

			Note: Different platforms have different ClassName. For example, in delphi button it is called TButton, while in QT it's called QPushButton.
--	--	--	---

The selection performs a full tree search and it also knows how to go back in a middle of a query and select another path to traverse. For example, a tree might have these elements:

Window

 Checkbox

 Pane

 Button

The query `.ui.t(Pane).t(Button)` works. The internal control finder returns back to search another 'Pane' once it sees the "Checkbox".