Lexema-RPA

LEXEMA-RPA USER GUIDE (EN)
VER 0.2

DATA PROCESSING modules

EXCEL module

The Excel module is the most extensive module in the program. This module is designed to process Excel files in the background, unnoticed by the computer user. The module will sequentially execute the list of commands given to it.

The Excel module works with files in the following way - as soon as it starts working with some file, it opens it (in background mode, unnoticeable to user), and keeps it open until the end of robot's work. This was done to speed up the Excel module - if there were several modules using the same file, for example, it would have to open and save the file as many times as it is used, which would have an impact on the program's performance. Therefore, a file is opened the first time it is used in Excel and saved only once, at the very end. If a file needs to be opened for reading only and there is no need to save it, the module has the appropriate setting. Otherwise, if you need to save the file during the robot's work because you need to use it outside of it (send it by mail, add it to the archive and so on), you should save it using the "Save file" action, after which it will be saved and closed and become available for third-party applications and computer users.

Module interface

The module window consists of several sections: "General Settings" (upper part of the window), "Add command", "List of commands" and the window with a preview of the file.

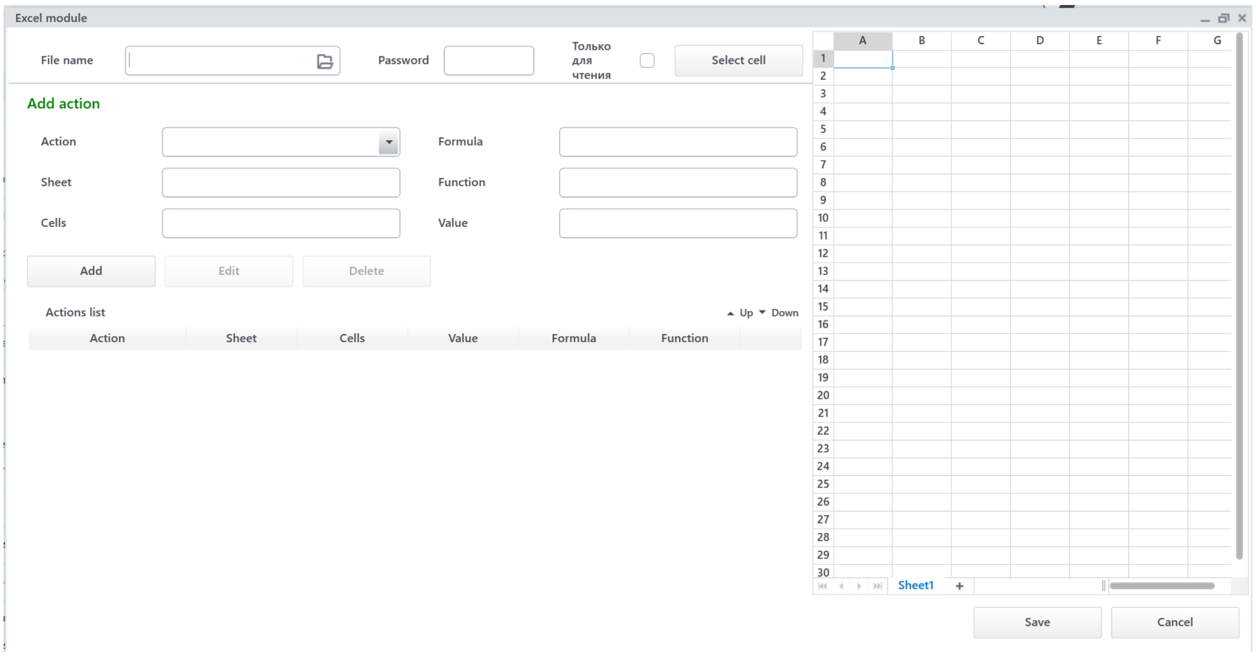


Figure 1. Excel Module window

The preview window allows you to view all the sheets of the loaded file using tabs with sheet names and a scroll bar. There is also an opportunity to write your own values into the cells and apply different formatting, but these changes will not be saved, i.e. this functionality is only available for "trying in".

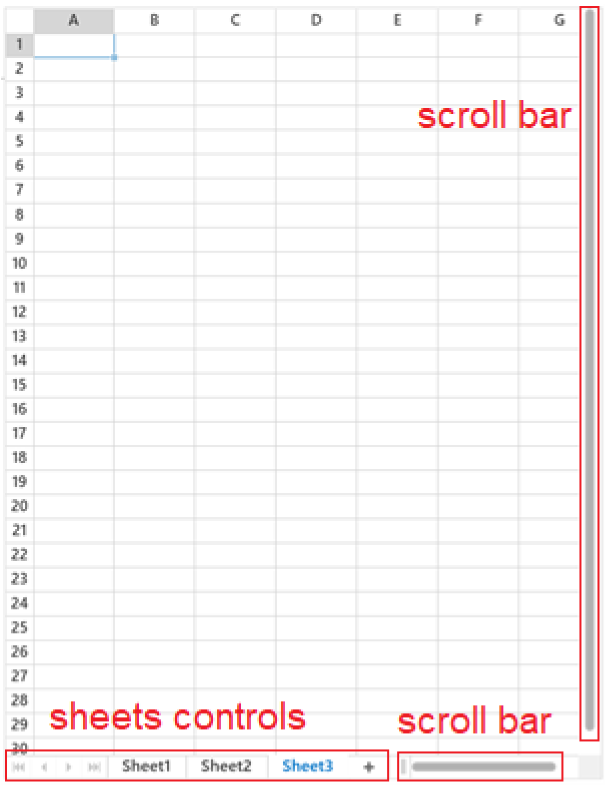


Fig. 2. Preview window controls

General settings section

The "General settings" section consists of the fields "File name", "Password", "Read only" checkbox and "Select cell" button. The field "Password" must be filled only for the locked books.

We recommend that you start working with the Excel module by selecting a work file. After selecting the file by the "Open File" button in the "File Name" field, you should wait for a few seconds and the specified file will be displayed in the preview window (Fig. 3).

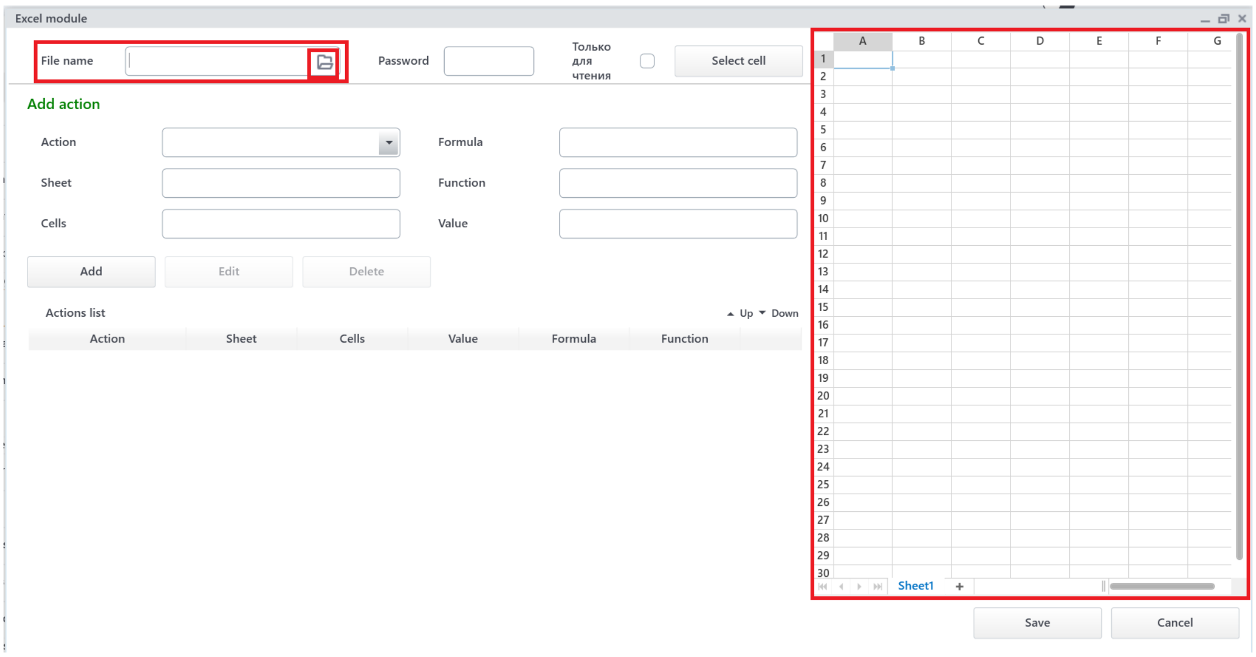


Fig. 3. General settings

Field "Read only" allows you to open a book and work with it, without saving changes made in it. However, the book can be saved by selecting the appropriate action, if necessary. Also, it is not necessary to check the "Read Only" box in each module of work with a particular file, it is enough to set it in the first module.

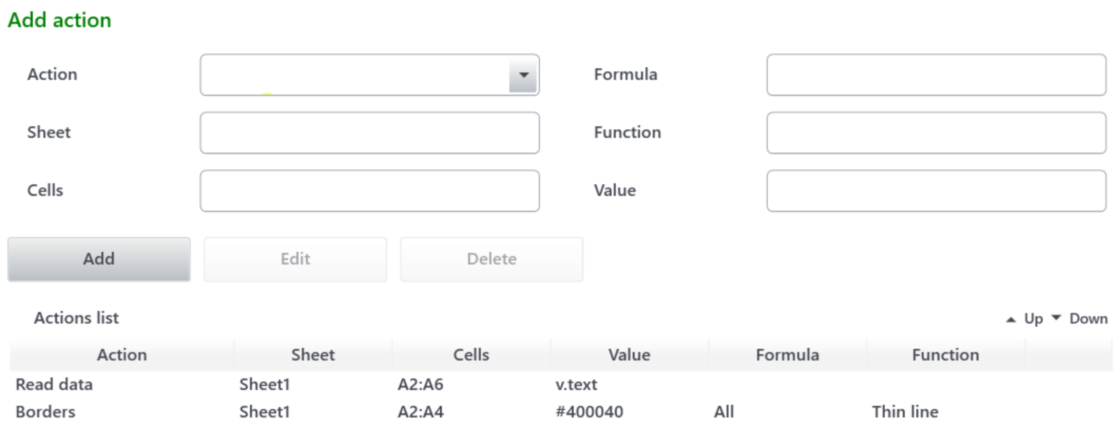
Add Command and Command List section

The "Add command" section is the main "canvas" of this module. It creates actions that simulate the user's work, such as counting a certain range of cells, then transferring it to another sheet, removing duplicates and building a pie chart. In this case, 4 actions have been described. These 4 actions must be sequentially selected and added to the "List of commands" section, a table showing all the actions to be performed by this module. The set of fields in this section is not always the same - it changes depending on the selected action in the first field.

The "Add", "Edit" and "Delete" buttons are intended for working with the "Add command" and "List of commands" sections.

Add Command

To add a command, use the "Add" button. It causes new action to be added to "List of commands" table.

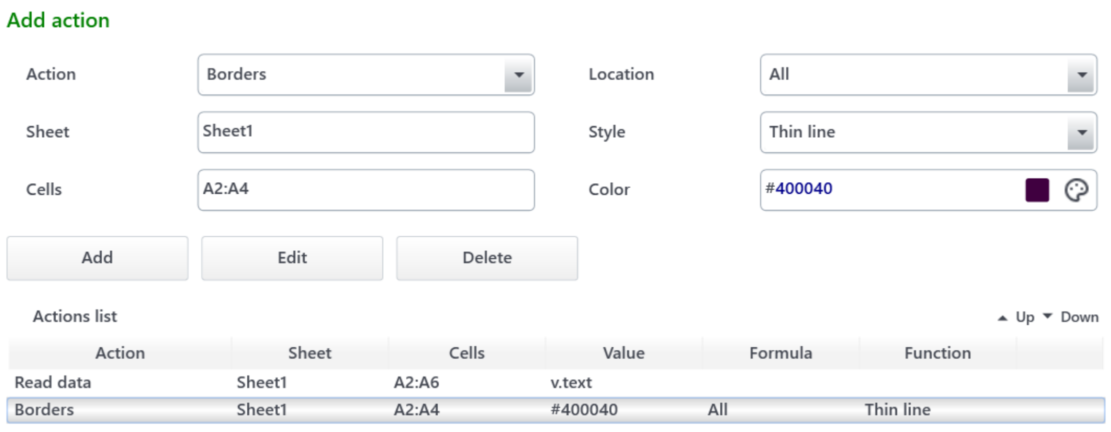


Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A2:A6	v.text		
Borders	Sheet1	A2:A4	#400040	All	Thin line

Fig. 4. Add command

If you choose a line in the list of commands, the information on the selected action will be displayed in the "Add command" section.

Besides described functionality, button "Add" serves for copying already existing actions in the list of commands. To do this, select a line in the list of commands and click on the "Add" button. Thus, if you need to execute action several times, it is enough to copy it instead of creating it again.



Action	Borders	Location	All
Sheet	Sheet1	Style	Thin line
Cells	A2:A4	Color	#400040

Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A2:A6	v.text		
Borders	Sheet1	A2:A4	#400040	All	Thin line

Fig. 5. Displaying of command

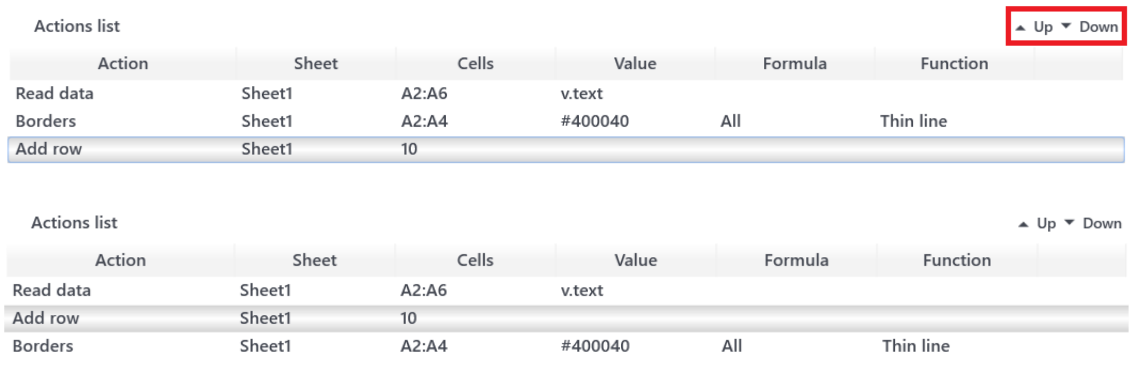
Edit command

To edit a command, you can use the "Edit" button. To change a command, select it in the command list, make necessary changes in the "Add command" section and press the "Edit" button to save the changes.

Changing the order of actions

To change the order of actions there are buttons "Raise" and "Lower". The buttons are located on the top right of the command list.

Changing the order of actions is necessary when you want to add an action, which is not provided at once and was not included in the list of commands. In this case, you can add an action in the standard way, and then move it.



Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A2:A6	v.text		
Borders	Sheet1	A2:A4	#400040	All	Thin line
Add row	Sheet1	10			

Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A2:A6	v.text		
Add row	Sheet1	10			
Borders	Sheet1	A2:A4	#400040	All	Thin line

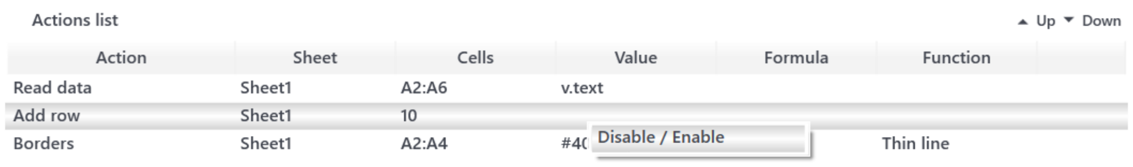
Fig. 6. Changing the order of actions

Delete command

The button "Delete" deletes the selected line from the list of commands.

Disable command

If you don't want to delete the command because you want to check how the robot works without it and then reset it, you can disable the command by right clicking on its line and disable or enable it.



Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A2:A6	v.text		
Add row	Sheet1	10			
Borders	Sheet1	A2:A4	#4(		Thin line

Fig. 7. Command context menu

Select cell button

The "Select Cell" button works differently for every action, but has the same meaning - it fills the "Cells" or "Range" field (depending on what kind of action you selected) with the name of the selected cell/range in the "Preview" window and enters the name of the sheet with the selected cells into the "Sheet" field. In some actions the button fills only one cell although a range was selected - it means that the range cannot be used in this action.

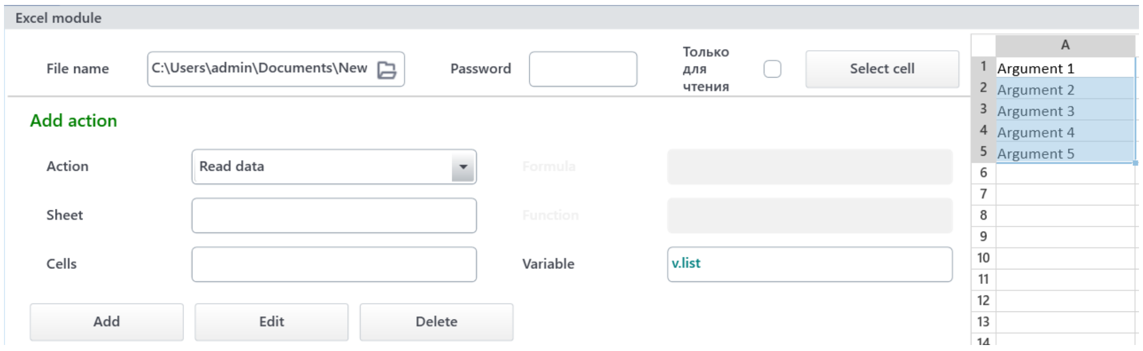


Fig. 8. Selecting the range in the preview window

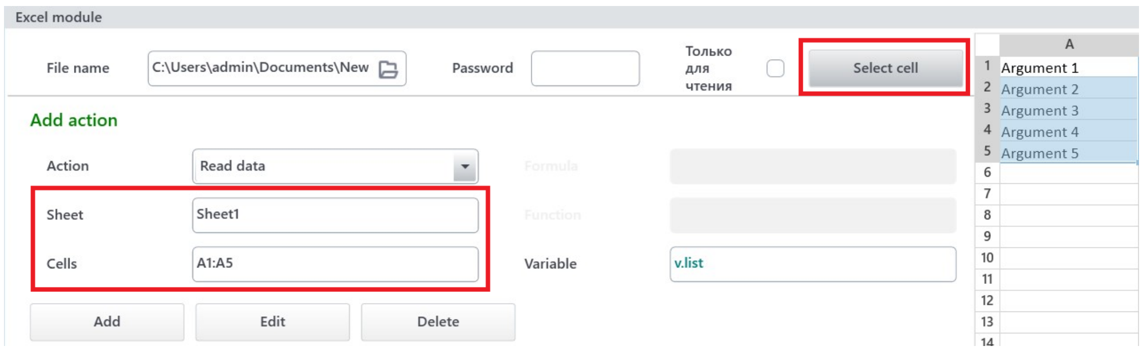
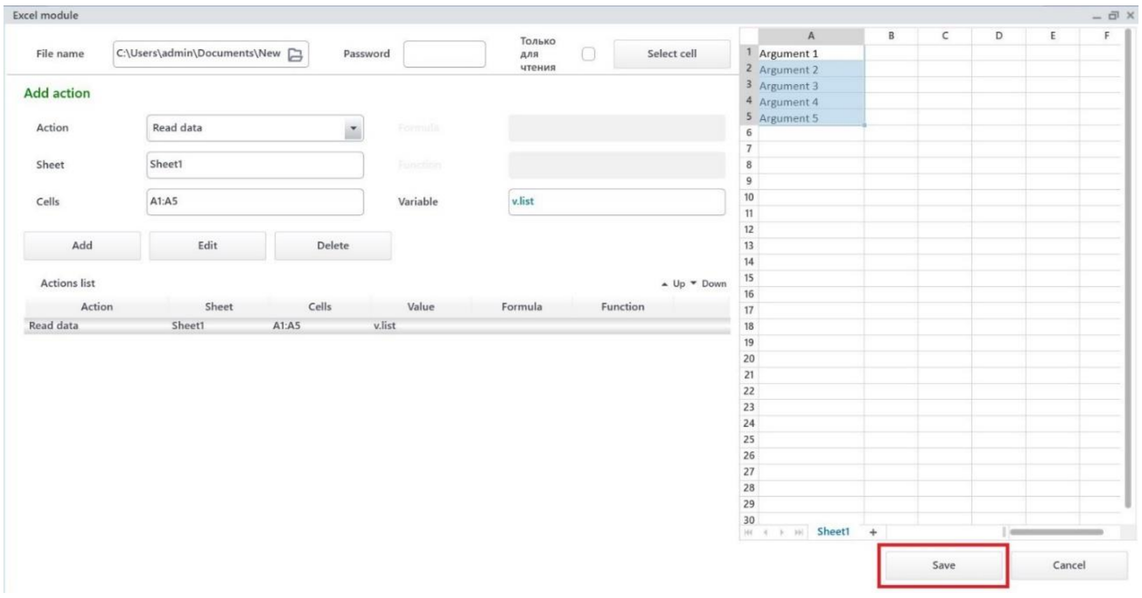


Fig. 9. Select cell

In some actions the button "Select Cell" fills in column numbers, rows, any values. When working with files it is recommended to use it to minimize syntax errors.

Saving a command

To save the command in the Excel module press the "Save" button in the lower right corner of the module.



File name: C:\Users\admin\Documents\New
Password:
Только для чтения: ☐

Add action

Action: Read data
Sheet: Sheet1
Cells: A1:A5
Formula:
Function:
Variable: v.list

Actions list

Action	Sheet	Cells	Value	Formula	Function
Read data	Sheet1	A1:A5	v.list		

	A	B	C	D	E	F
1	Argument 1					
2	Argument 2					
3	Argument 3					
4	Argument 4					
5	Argument 5					

Sheet1

Figure 10. Saving in Excel module

Actions available in the module

Actions are divided into groups for convenient navigation through the drop-down list. If you know the exact name of action, you can start typing it in the "Action" field and the program will try to tell you the action you are looking for.

General rules for filling in the fields

When specifying a sheet to work on, you can specify its sequence number in the book, starting with 0;

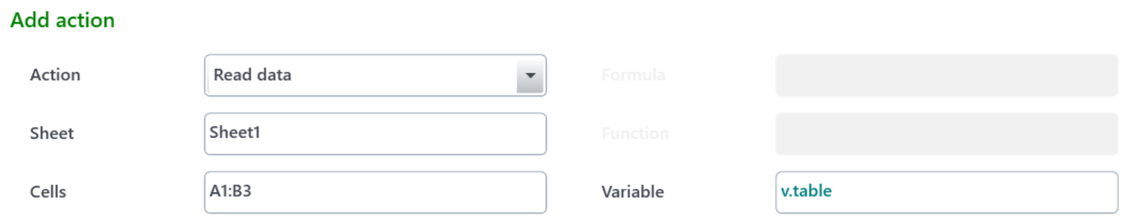
When specifying a range, you must separate the top left cell and the bottom right cell with a colon (A1:C3 includes a range of 9 cells in columns A through C and rows 1 through 3);

Many actions support specifying a cell as "k,c", where k is the column number and c is the row number. This expression can be used in cases where you do not know the name of the column, but have its number. Columns and rows in the expression are counted from 1. Range A1:C3 will be specified as 1,1:3,3. Doesn't work in formulas.

Data Handling group

Read Data Action

This action allows you to read a value from one specific cell, or from a range of cells. If a single cell is specified, a variable of Element type will be created. If a range of cells is specified, a variable of Table type will be created. All values will be rows.

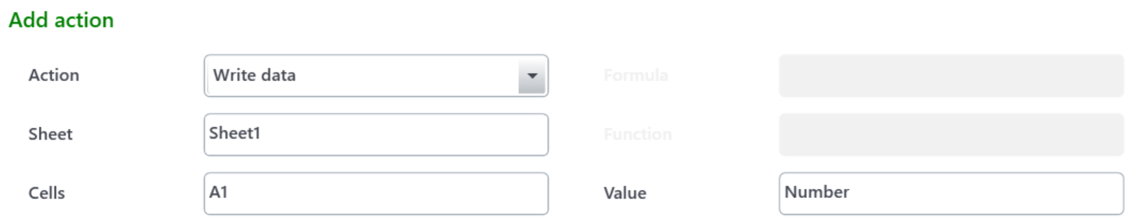

Add action

Action	Read data	Formula	
Sheet	Sheet1	Function	
Cells	A1:B3	Variable	v.table

Fig. 11. Example of Read data action setting

Write data Action

The action lets you write a specified value to one or a range of cells. All values will be written as lines, therefore if you want a cell to have something else you should use "Format cell" action described below.

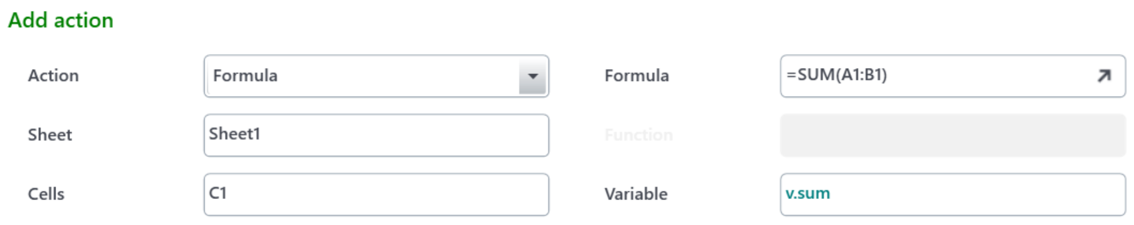

Add action

Action	Write data	Formula	
Sheet	Sheet1	Function	
Cells	A1	Value	Number

Fig. 12. Example of setting the "Write data" action

Formula Action

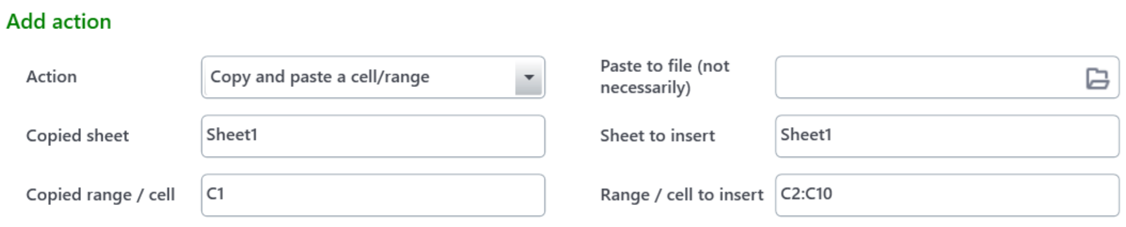
The action lets you write formula to a specified cell (or range) and return its result to a specified variable. You can leave the "Variable" field empty if the result of this formula is not important.


Add action

Action	Formula	Formula	=SUM(A1:B1)
Sheet	Sheet1	Function	
Cells	C1	Variable	v.sum

Figure 13: Example of setting a Formula action

TIP. Suppose you need to write a formula that applies to the current row and "stretch" it to several other rows, as in Excel. Start by creating a Formula action for the first row. Then "stretch" it with the "Copy and Paste Cell/Range" action applied to the customized formula. That is for the above suggested formula "stretching" will look like this:



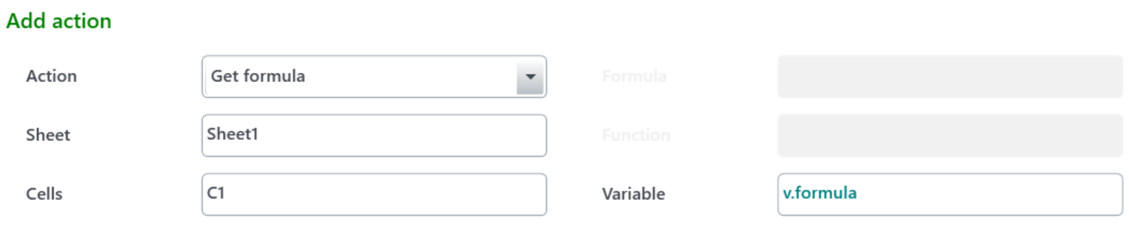
Add action

Action	Copy and paste a cell/range	Paste to file (not necessarily)	
Copied sheet	Sheet1	Sheet to insert	Sheet1
Copied range / cell	C1	Range / cell to insert	C2:C10

Figure 14: Example of setting up the "Copy and Paste Cell/Range" action

Get formula action

The action returns formula from cell to a specified variable.



Add action

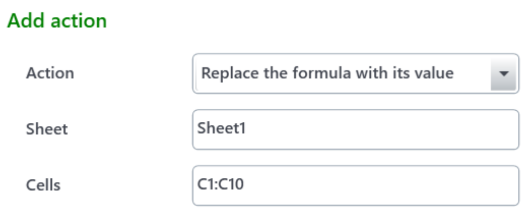
Action	Get formula	Formula	
Sheet	Sheet1	Function	
Cells	C1	Variable	v.formula

figure 15. The Get formula setting example

Fig. 15. Example of setting the "Get formula" action

Replace formula with its value action

This action replaces a formula in a cell or range of cells with its result.



Add action

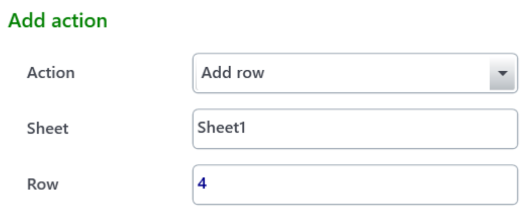
Action	Replace the formula with its value
Sheet	Sheet1
Cells	C1:C10

Figure 16. Example of setting the "Replace a formula with its value" action.

Working with rows Group

Add Row action

Inserts a new line in specified position.



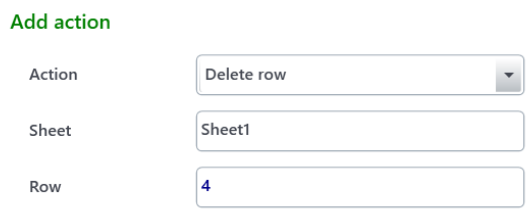
Add action

Action	Add row
Sheet	Sheet1
Row	4

Fig. 17: Example of how to set "Add row" action

Delete row action

This action removes the specified line.



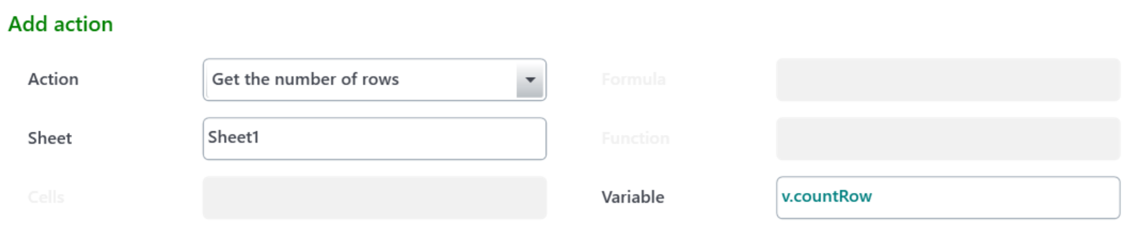
Add action

Action	Delete row
Sheet	Sheet1
Row	4

Fig. 18 Example of how to set "Delete row" action

Get number of rows action

The action returns the number of the last used row in the worksheet into the specified variable. The result will be a number.



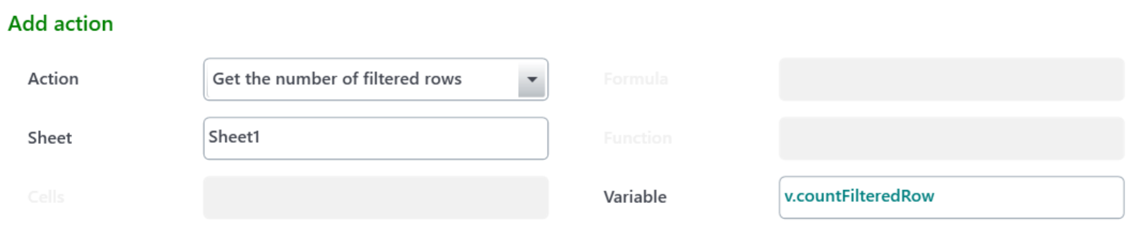
Add action

Action	Get the number of rows	Formula	
Sheet	Sheet1	Function	
Cells		Variable	v.countRow

Fig. 19: Example of customization of "Get number of rows" action

Get number of lines filtered action

The action returns the number of visible rows with applied filter into the specified variable. The result will be a number.



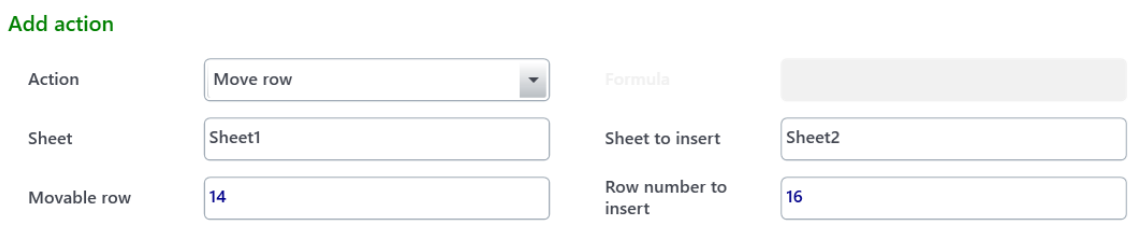
Add action

Action	Get the number of filtered rows	Formula	
Sheet	Sheet1	Function	
Cells		Variable	v.countFilteredRow

Fig. 20. The example of setting the "Get number of strings filtered" action.

Move row Action

The action places the row on the sheet from the "Sheet" field instead of the row number, indicated in the "Line number to insert" field of the "Sheet to insert" sheet.



Add action

Action	Move row	Formula	
Sheet	Sheet1	Sheet to insert	Sheet2
Movable row	14	Row number to insert	16

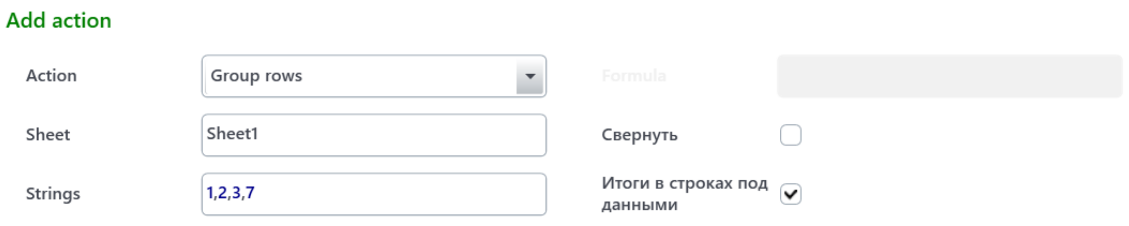
Figure 21: Example of how to set the Move row action

In this example, line 14 of the sheet 1 will be moved to position 16 of the sheet 2.

Group rows action.

This action is used to create groups of rows in Excel. The input is the name of the sheet on which the grouping is taking place, and a list of row numbers. You can list the rows simply separated by a comma or, if the range is not broken, by a colon, for example - "1,2,3,7,8,9" or "1:5", which is equivalent to "1,2,3,4,5".

By default, the groups will be expanded and the group headings (totals) will be above the groups themselves. Use the "Collapse" and "Totals in rows under data" checkboxes to change these settings, respectively. "Totals in rows under data" works similarly to the setting of the same name in Excel, available in the "Data" - "Structure" settings window.

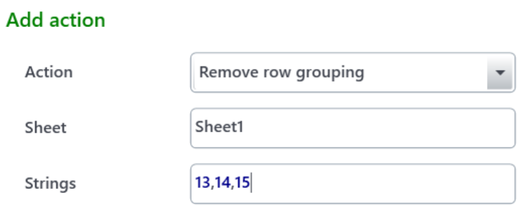

Add action

Action	Group rows	Formula	
Sheet	Sheet1	Свернуть	☐
Strings	1,2,3,7	Итоги в строках под данными	✓

Fig. 22: Example of setting "Group rows" action

Remove row grouping Action

The action is designed to delete the 1st level of the existing row grouping. The input data is similar to the action "Group rows".


Add action

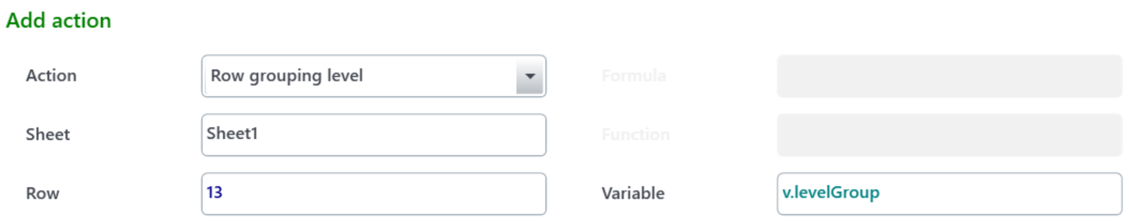
Action	Remove row grouping
Sheet	Sheet1
Strings	13,14,15

Fig. 23. Example of action settings "Delete row grouping"

Row grouping level action

The action allows to get the grouping level of the selected line and to write the result to the defined variable.

If you have selected one row, the result will be a number, otherwise it will be an array of numbers, where the index of the array will correspond to the index of the specified row.


Add action

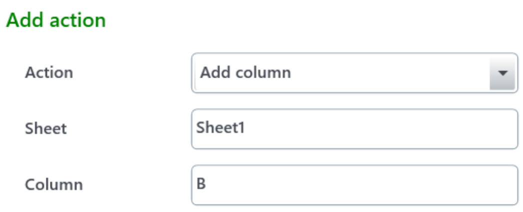
Action	Row grouping level	Formula	
Sheet	Sheet1	Function	
Row	13	Variable	v.levelGroup

Fig. 24. The example of action setting "Rows grouping level".

Working with Columns group of actions

Add column Action

The action adds a new column to the specified location. The column can be specified as a number (from 1) or a name.

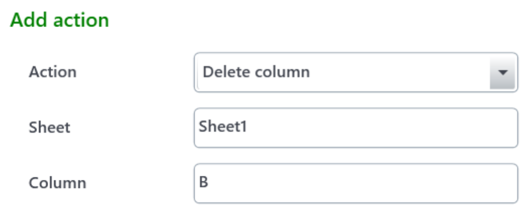

Add action

Action	Add column
Sheet	Sheet1
Column	B

Fig. 25. example of how to set the "Add column" action

Delete column Action

The action deletes the specified column. The column can be specified as a number (from 1) or a name.



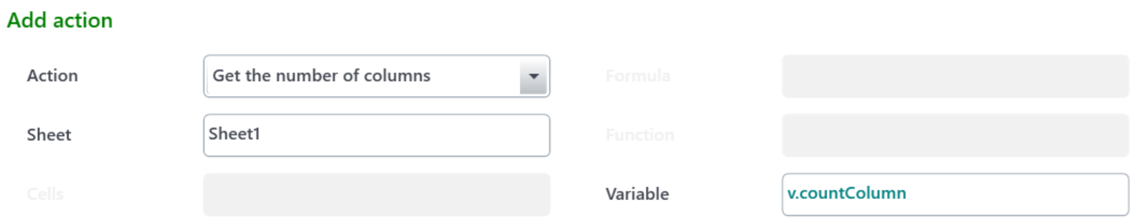
Add action

Action	Delete column
Sheet	Sheet1
Column	B

Fig. 26. Example of setting the "Delete column" action

Get number of columns action

The action returns the number of the last used column. The result will be a number.



Add action

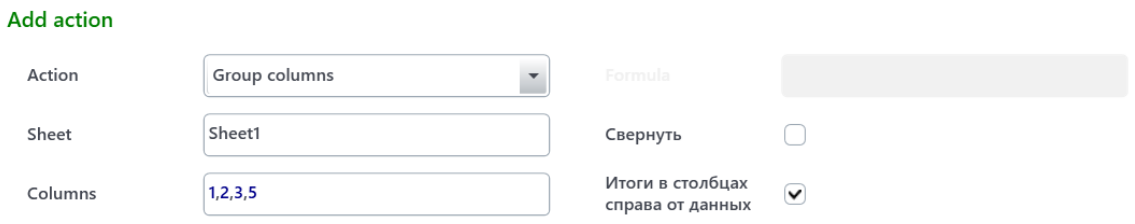
Action	Get the number of columns	Formula	
Sheet	Sheet1	Function	
Cells		Variable	v.countColumn

Fig. 27. Setting example of "Get number of columns" action

Group Columns Action

The action joins the specified columns into the group. The input data are a sheet and a list of columns, separated by commas or as a range, separated by colons ("A,B,C,D,E" or "A:E"). As a column, you can specify both its name and its sequence number, starting with one.

By default, the groups will be expanded and the group headings (totals) will be on the left of the groups. You can use the "Collapse" and "Totals in columns to the right of the data" checkboxes to change these settings, respectively. "Totals in columns to the right of data" works similarly to the setting of the same name in Excel, available in the "Data" - "Structure" settings window.



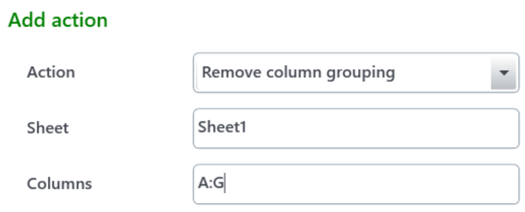
Add action

Action	Group columns	Formula	
Sheet	Sheet1	Свернуть	☐
Columns	1,2,3,5	Итоги в столбцах справа от данных	✓

Figure 28. Example of "Group Columns" action setting

Remove column grouping Action

This action removes one grouping level for the specified columns.



Add action

Action	Remove column grouping
Sheet	Sheet1
Columns	A:G

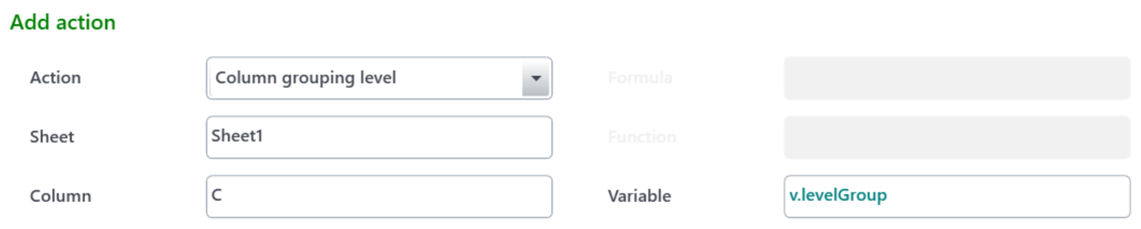
Fig. 28. Example of columns grouping settings

Column grouping level action

The action allows to get the grouping level of the chosen column, columns list separated by comma or columns range and to write the result to the chosen variable.

If one column is selected, the result will be a number, otherwise it will be an array of numbers, where the array index will correspond to the index of the specified column.

The columns can be specified as their name or ordinal number from the 1st column.



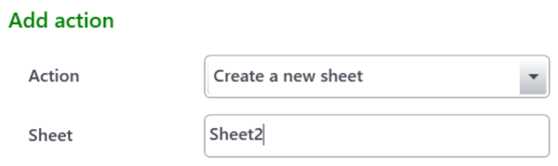
Action	Column grouping level	Formula	
Sheet	Sheet1	Function	
Column	C	Variable	v.levelGroup

Fig. 30. An example of the setting of the "Column grouping level" action

Works with sheets and file Group

Create new sheet Action

The action creates a new sheet with a specified name.

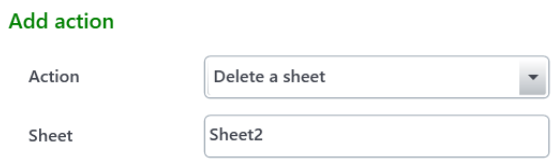


Action	Create a new sheet
Sheet	Sheet2

Fig. 31. An example of how to set the "Create new sheet" action

Delete sheet action

The action deletes the sheet with a specified name.

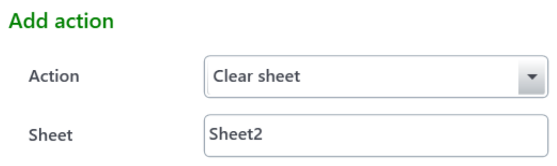


Action	Delete a sheet
Sheet	Sheet2

Fig. 32. Example of action "Delete sheet" settings

Clear sheet action

The action clears the sheet cells from the data. It doesn't clear the formatting!

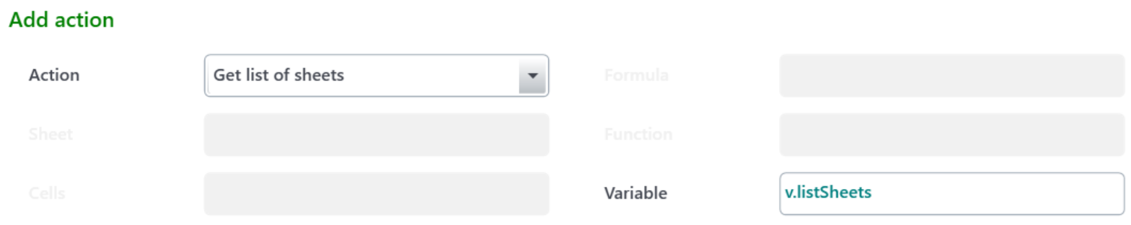


Action	Clear sheet
Sheet	Sheet2

Fig. 33. Example of "Clear sheet" action settings

Get List of Sheets action

Returns a variable list containing the names of all the sheets in the file.



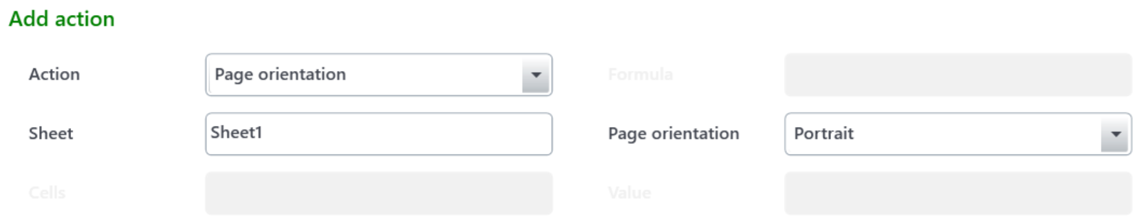
Add action

Action	Get list of sheets	Formula	
Sheet		Function	
Cells		Variable	v.listSheets

Fig. 34. Example of the "List fetching" action settings

Page orientation action

Change the Excel page orientation. The available orientations are portrait and landscape.



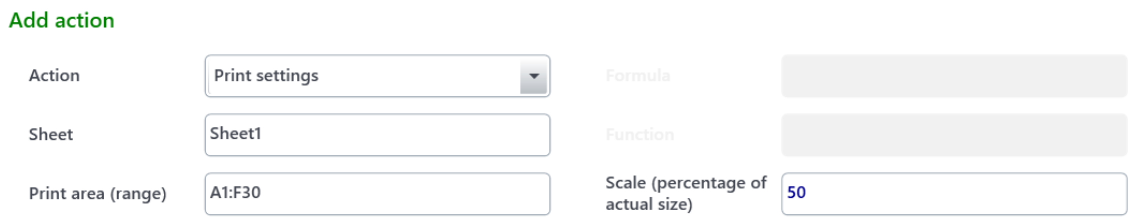
Add action

Action	Page orientation	Formula	
Sheet	Sheet1	Page orientation	Portrait
Cells		Value	

Fig. 35. Example of "Page orientation" action settings

Print Settings action

The action allows you to select the area of the sheet that should be displayed on the page when printing, as well as the scale of that area. It is similar to the Excel setting of the same name.



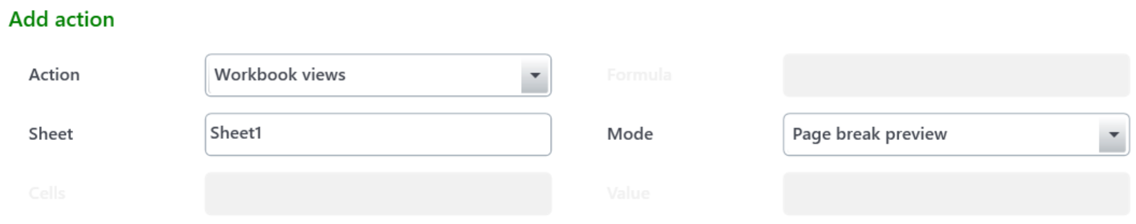
Add action

Action	Print settings	Formula	
Sheet	Sheet1	Function	
Print area (range)	A1:F30	Scale (percentage of actual size)	50

Fig. 36. Example of "Print settings" action setting

Workbook View Mode Action

The action allows you to choose to change the display mode of data in the Excel workbook. Available view modes are Normal, Page and Layout.



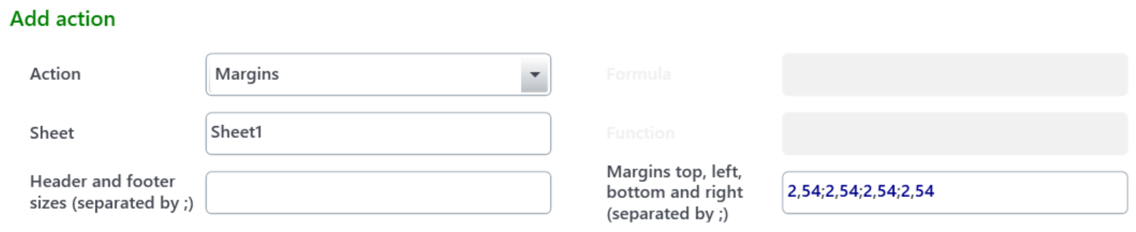
Add action

Action	Workbook views	Formula	
Sheet	Sheet1	Mode	Page break preview
Cells		Value	

Fig. 37. Example of workbook view mode settings

Fields. Action

Sets specified fields on a particular sheet. Three fields are provided for making settings - the "Sheet" field is used to specify the sheet, the "Header and Footer sizes w/o ;" field is used to specify header and footer sizes respectively (in cm), the "Margins at top, left, bottom and right w/o ;" field is used to specify corresponding margins for each side (in cm). All listings are preceded by ";".


Add action

Action	Margins	Formula	
Sheet	Sheet1	Function	
Header and footer sizes (separated by ;)		Margins top, left, bottom and right (separated by ;)	2,54;2,54;2,54;2,54

Fig. 38. Example of the "Fields" action setting

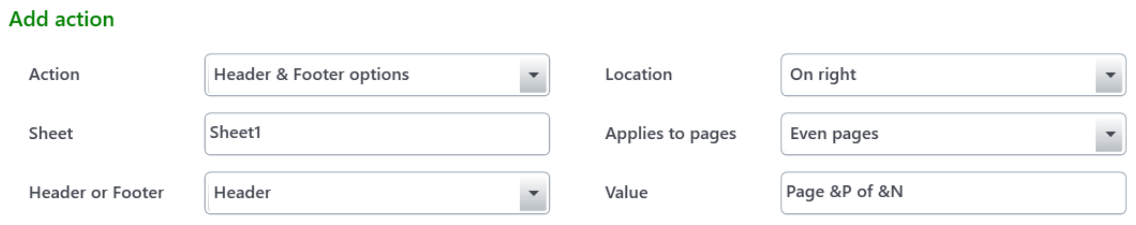
Customize header and footer Action

Use this action to fill page footers on an Excel sheet. All fields are mandatory. In the "Header" field, you select whether the header or footer will be applied to the header or footer. "Location" - one of three text locations within the header - left, right, or center. "Applicable to Pages" - you can make the first header special, different from the others, and also make different headers for even and odd pages, or select "All" and all pages will have the same headers.

Table 1

Value	Symbol
Current page number	&P
Total number of pages	&N
Current date	&D
Current time	&T
Document path	&Z
Document name	&F
Page name	&A
Inserting the "&" symbol	&&
Enabling or disabling bold text formatting	&B
Turns italics on or off	&I
Turns underline text on or off	&U
Turns double-underlining text on or off	&E
Enables or disables subscript formatting	&Y
Enables or disables superscript formatting	&X

In the "Value" field, you enter the line to be inserted in the header in the previously selected location. To insert dynamic data into the line, such as the current page, total number of pages, date and time, and so on, special designators are used. The complete list of symbols is presented in Table 1.

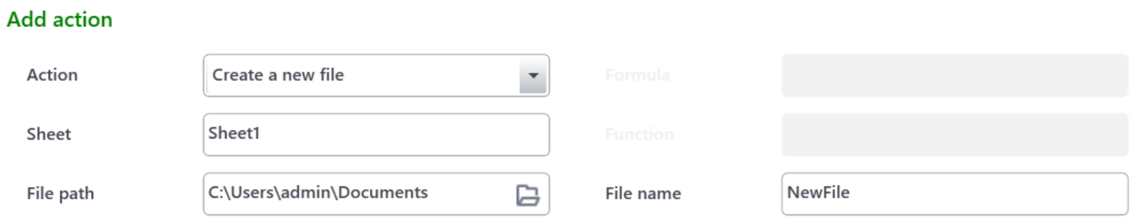

Add action

Action	Header & Footer options	Location	On right
Sheet	Sheet1	Applies to pages	Even pages
Header or Footer	Header	Value	Page &P of &N

Fig. 39. An example of the "Header setting" action

Create new file Action

The action creates a new file in a specified folder with a specified name and extension (if no extension is specified, the .xlsx file is created). If the "Sheet" field is filled in, the first sheet of the file will be named as indicated in the field, otherwise it will be named by default (Sheet 1, Sheet 1, depending on the system language).



Add action

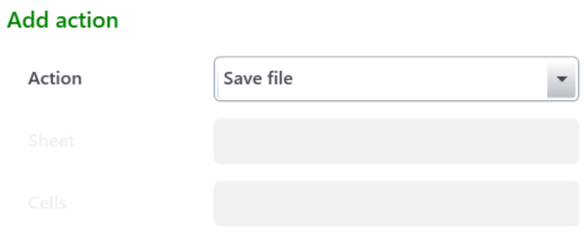
Action	Create a new file	Formula	
Sheet	Sheet1	Function	
File path	C:\Users\admin\Documents	File name	NewFile

Fig. 40. An example of the "Create new file" action settings

If after the file creation it is necessary to do something with it, it is necessary to create the new Excel module, where the path to the created file should be specified in the field "File name" in the settings area.

Save File Action

The action will save the file and close it. There can be no other actions after this action. If the file is to be used repeatedly, one should create a new Excel unit and work with it there.



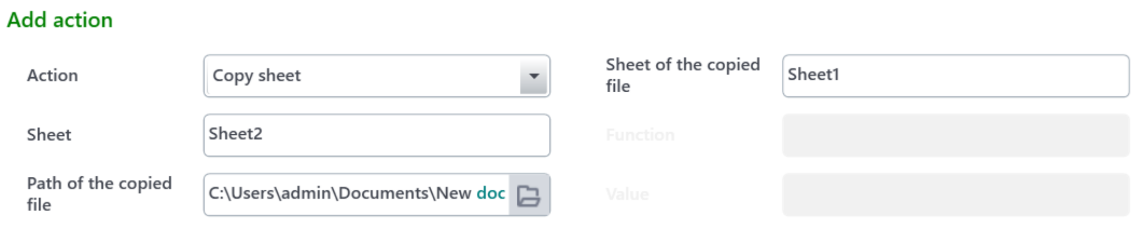
Add action

Action	Save file
Sheet	
Cells	

Fig. 41. Example of "Save file" action settings

Copy sheet Action

This action fills the sheet of the current file by analogy with another sheet from the same or another file. The sheet of the current file is specified in the "Sheet" field, the "Copy File Path" can be left blank if the file is being copied within the same document, the "Copy File Sheet" is the name of the sheet from which the data is copied.



Add action

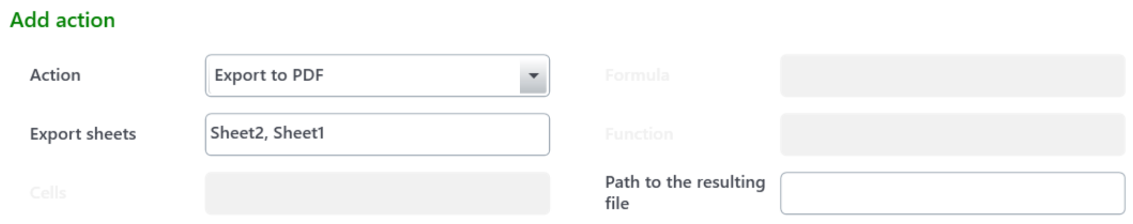
Action	Copy sheet	Sheet of the copied file	Sheet1
Sheet	Sheet2	Function	
Path of the copied file	C:\Users\admin\Documents\New doc	Value	

Fig. 42. Example of "Copy sheet" action settings

Export to PDF Action

The action allows you to save the Excel file in PDF format by standard Windows methods, i.e. the Excel file sheet will be divided into sheets in the PDF document by the standard Excel file settings (default sheet size is A4, with standard margins). If you need to unload only certain sheets of the book - you need to pass them by comma or by variable sheet in the "Sheets to unload" field. If you want to unload all the sheets - just leave the field blank.

If you want to unload a file in a directory other than the main file, you must specify the full path to the destination file in the "Path to resulting file" field. If the field is left empty, the file will be created in the same directory and with the same name as the original Excel file.


Add action

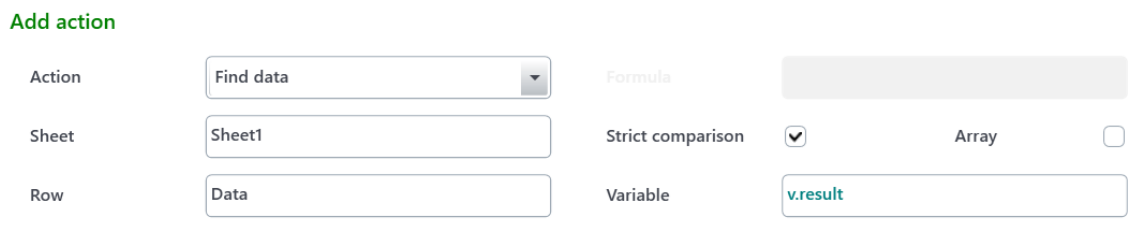
Action	Export to PDF	Formula	
Export sheets	Sheet2, Sheet1	Function	
Cells		Path to the resulting file	

Fig. 43. Example of "Export to PDF" action setting

Search group

Find Data Action

Searches for the specified row in the specified sheet and returns the address of the first found cell if the "Array" checkbox is not checked, and an array of cell addresses if checked. The checkbox "Strict comparison" is checked if the text in the cell must be completely identical with the text entered in the field "String".

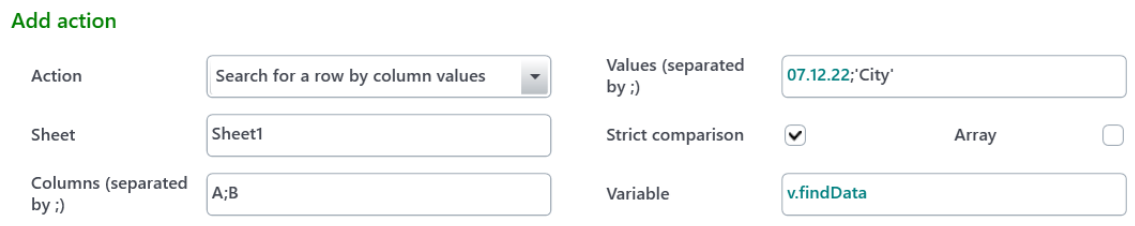

Add action

Action	Find data	Formula			
Sheet	Sheet1	Strict comparison	✓	Array	☐
Row	Data	Variable	v.result		

Fig. 44. The example of action's "Find data" setting

Find row by column values Action

This action searches for and returns the numbers of rows, in the specified columns of which the specified values are written. The Strict comparison checkbox is checked if the text in the cells must be exactly the same as the text entered in the Values field. The checkbox "Array" will return an array of all strings found, the resulting variable will be an array of numbers. If the checkbox "Array" is not checked, then the number of the first found string or -1, if there are no such strings, will be written in the result variable.


Add action

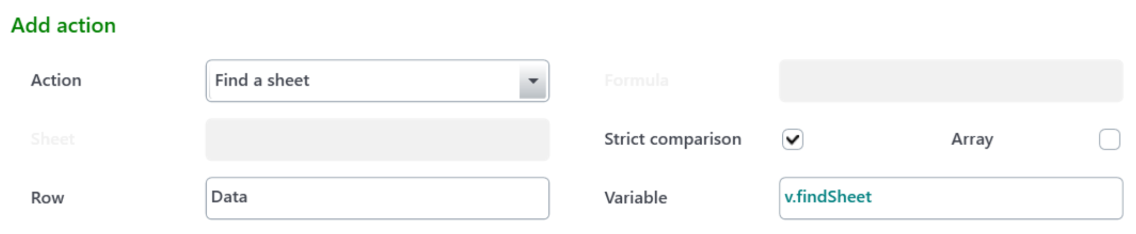
Action	Search for a row by column values	Values (separated by ;)	07.12.22;'City'		
Sheet	Sheet1	Strict comparison	✓	Array	☐
Columns (separated by ;)	A;B	Variable	v.findData		

Fig. 45. Example of setting the "Search row" action

In this case, there will be one line searched, with column A containing 23.03.2020 and column B containing City.

Find Sheet Action

Searches for sheets that have a cell with the value specified in the String field and returns the search result to the specified variable. The checkbox "Strict comparison" is checked if the text in the cell must be exactly the same as the text entered in the field "String". If the Array checkbox is on



Add action

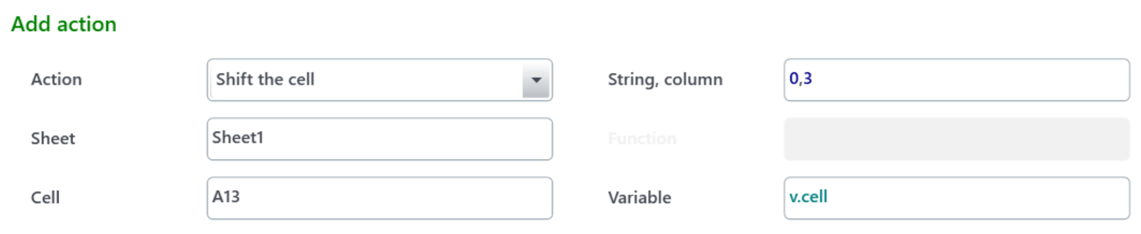
Action	Find a sheet	Formula	
Sheet		Strict comparison	✓
		Array	☐
Row	Data	Variable	v.findSheet

Fig. 46. The example of the "Find sheet" action setting

Other functions group

Shift cell Action

It shifts the cell by specified number of rows and columns and then returns the result to a variable (in this example the result is cell D13).



Add action

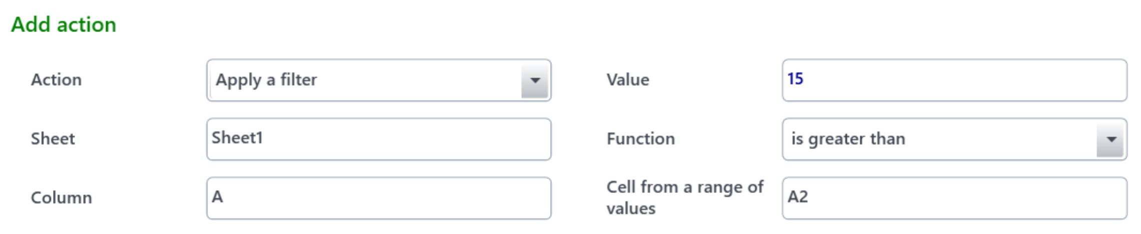
Action	Shift the cell	String, column	0,3
Sheet	Sheet1	Function	
Cell	A13	Variable	v.cell

Fig. 47. Example of " Shift cell" action configuration

Apply Filter Action

This action adds a filter to the table on the specified sheet, the cell from which is specified in the "Value range cell" field. The column you want to apply the filter to is written in the Column field. The column to be filtered must be one. If you want to apply several filters to one table, you should add a different action for each filter within the same Excel module. The function to be compared is selected from the drop-down list in the "Function" field, the value to be compared during filtering is specified in the "Value" field.

If you just want to enable the filter in the table, but without any specific filtering, you should leave the "Function" field empty. If you want to filter by empty values, you must type "null" or "(empty)" in the "Value" field.



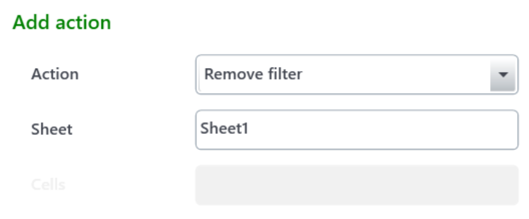
Add action

Action	Apply a filter	Value	15
Sheet	Sheet1	Function	is greater than
Column	A	Cell from a range of values	A2

Fig. 48. Example of the "Apply filter" action settings

Remove filter Action

This action deletes all the filters from the list.



Add action

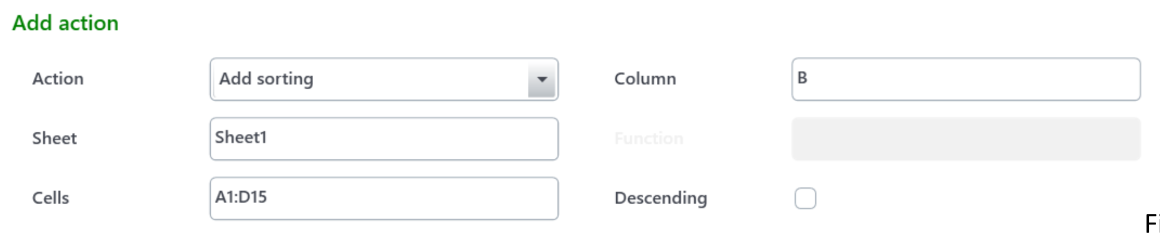
Action	Remove filter
Sheet	Sheet1
Cells	

Fig. 47. Example of the "Remove filter" action settings

Add sorting action

The action adds sorting to the specified range by the selected column. Depending on the checkbox, it is determined whether the column is sorted in descending or ascending order. There must be only one column to sort.

If you need to sort by several columns - add several sorting actions on all necessary columns inside one Excel module and they will all be applied simultaneously.



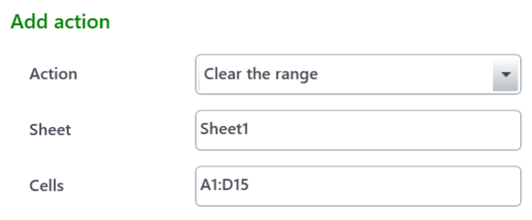
Add action

Action	Add sorting	Column	B
Sheet	Sheet1	Function	
Cells	A1:D15	Descending	☐

Fig. 50. Example of the "Add sorting" action setting

Clear range Action

This action removes all values and formatting of the cells in the specified range.



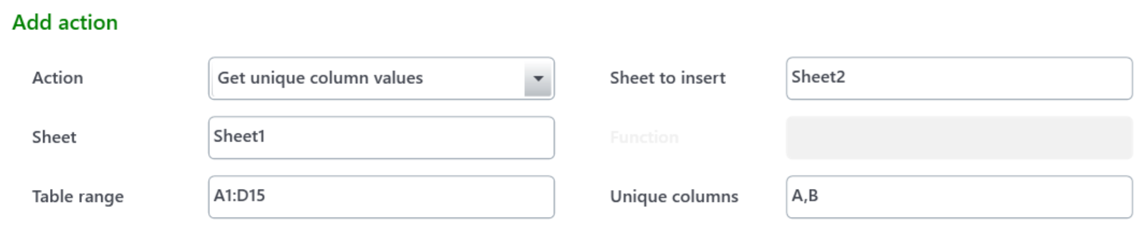
Add action

Action	Clear the range
Sheet	Sheet1
Cells	A1:D15

Fig. 51. Example of range cleaning action settings

Get Unique Column Values action

Similar to the Remove Duplicates action in Excel, this action returns only the unique values of the specified columns from the range. The summary table can be pasted either in the same sheet or in another sheet but in the same range. Depending on the presence of duplicates, the number of rows can be reduced.



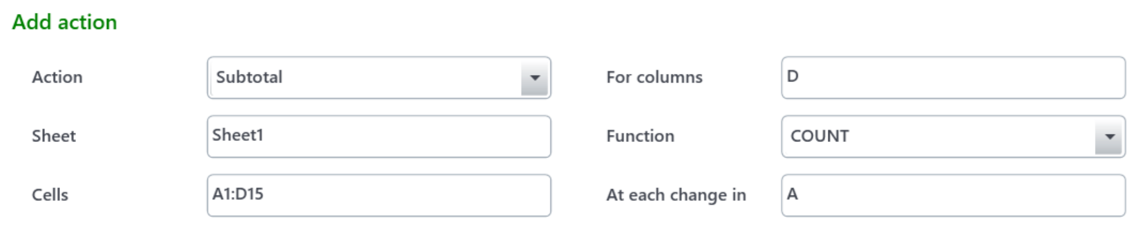
Add action

Action	Get unique column values	Sheet to insert	Sheet2
Sheet	Sheet1	Function	
Table range	A1:D15	Unique columns	A,B

Fig. 52. The example of action settings "Get unique values".

Intermediate totals Action

The action adds the Excel operation "Intermediate Totals" to the specified worksheet to the specified range.



Add action

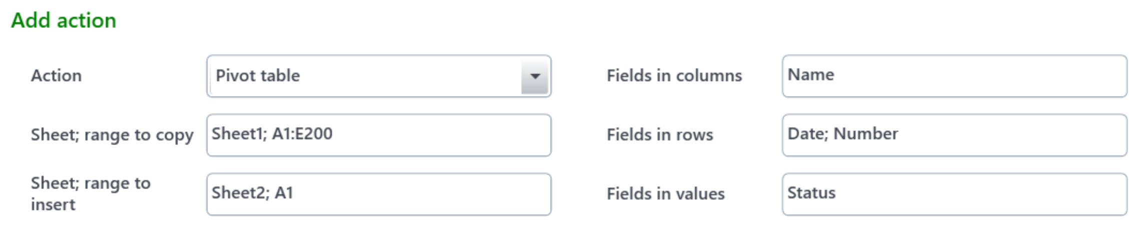
Action	Subtotal	For columns	D
Sheet	Sheet1	Function	COUNT
Cells	A1:D15	At each change in	A

Figure 53. An example of the Intermediate Totals action setting

The field For columns corresponds to the Excel field Add Totals by (it is green in figure 54), the field For each change in a column corresponds to the Excel field but should be filled in with the Excel column name (A, B, ... or 1, 2, ...) (it is red in figure 54), the field Function is filled in with the operation to be applied to the selected columns (it is blue in figure 54).

Summary Table action.

Adds a summary table to the specified sheet and range (Sheet; range to insert field) based on the table data from the Sheet; source range field. All enumerations in this module must be specified with a ";".


Add action

Action	Pivot table	Fields in columns	Name
Sheet; range to copy	Sheet1; A1:E200	Fields in rows	Date; Number
Sheet; range to insert	Sheet2; A1	Fields in values	Status

Fig. 55. Example of setting the "Summary table" action

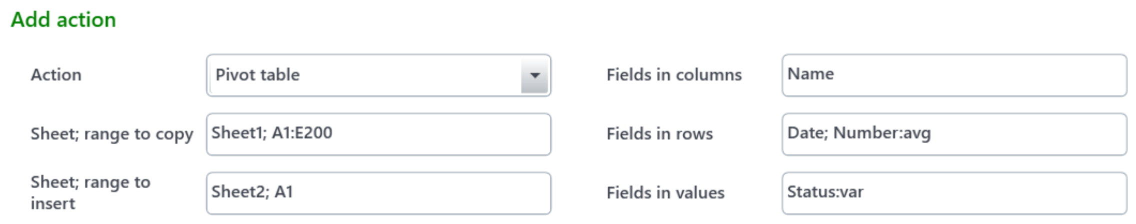
The column names of the source table columns (and not of the Excel table) are entered in the Column Fields, Value Fields, and Row Fields fields, as in the Excel program:

- "Fields in Columns" corresponds to the names of the table columns whose values should be located in the summary table columns (highlighted in green in Figure 56);
- "Fields in rows" correspond to the names of the table columns, the values of which should be located in the rows of the summary table (highlighted in blue in Figure 56);
- "Fields in Values" corresponds to the names of the table columns whose values are to be located in the values of the summary table (highlighted in red in Figure 56);

Excel function	Lexema-RPA Studio **analogue**
Sum	sum, сумма
Count	count, количество
Average	avg, среднее, average
Maximum	max, макс, максимум
Minimum	min, мин, минимум
Product	product, multiply, произведение, умножение
Count of numbers	countnumbers, количествочисел
Standard deviation	stddev, стандартноеотклонение
Unbiased Standard Deviation	stdtevp, несмещенноестандартноеотклонение
Dispersion	var, дисперсия
Unbiased variance	varp, несмещеннаядисперсия

To change the function of counting values in the "Values" fields, you should add the ":" symbol and the function name to the selected field. All names are given in the table below.

An example of using the counting function is shown in the figure below.

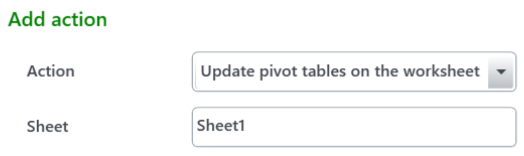

Add action

Action	Pivot table	Fields in columns	Name
Sheet; range to copy	Sheet1; A1:E200	Fields in rows	Date; Number:avg
Sheet; range to insert	Sheet2; A1	Fields in values	Status:var

Figure 57. An example of using the counting functions

Update pivot table on worksheet Action

This action updates the crosstabs, which means that if the data on which they were built has changed, the crosstabs must be updated to recalculate the values in the crosstab. Updates all the tables on a single sheet.



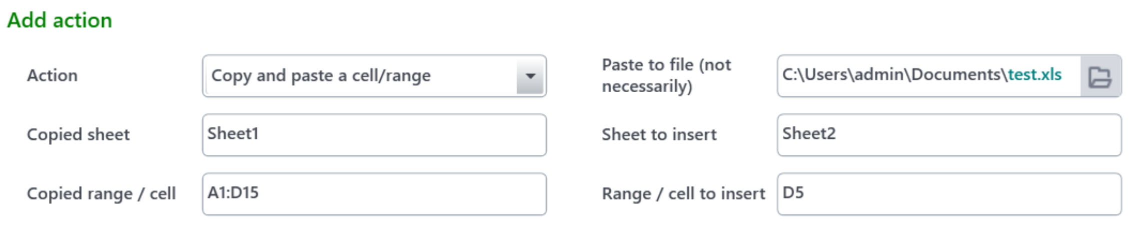
Add action

Action	Update pivot tables on the worksheet
Sheet	Sheet1

Figure 58. Example of setting the Update crosstabs on worksheet action

Copy and Paste Cell/Range Action

This action allows you to copy data and formatting and paste them into the same or into another file. It can be used to "stretch" formulas (see Action Formula).



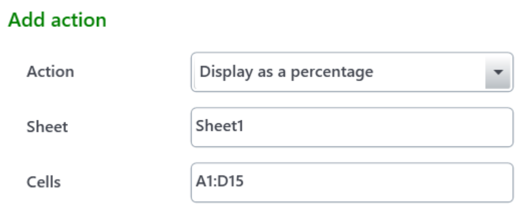
Add action

Action	Copy and paste a cell/range	Paste to file (not necessarily)	C:\Users\admin\Documents\test.xls
Copied sheet	Sheet1	Sheet to insert	Sheet2
Copied range / cell	A1:D15	Range / cell to insert	D5

Fig. 59. Example of "Copy and Paste Cell/Range" action settings

Display as percentage action

The action converts the selected range into percent.



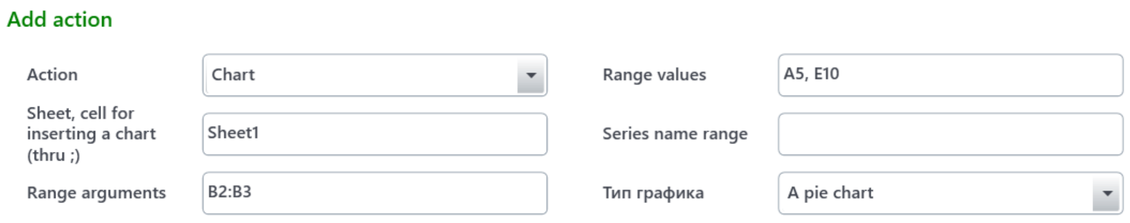
Add action

Action	Display as a percentage
Sheet	Sheet1
Cells	A1:D15

Figure 60. Example of "Show as percent" action settings

Chart Action

The action creates a line chart with one or more rows. The rows should be specified in the "Range of values" field as a one-dimensional range (one column or one row at a time). If there are more than one row, the subsequent rows must be specified in the same form with ";" as shown in Figure 64. The range of arguments must be one and specified similarly to the row. "Cells with row names" are filled with cell names with ";", their number should be equal to the number of rows, the first cell displays the name of the first row, etc. The cell to be inserted is the cell in which the upper left corner of the graph will be located.



Add action

Action	Chart	Range values	A5, E10
Sheet, cell for inserting a chart (thru ;)	Sheet1	Series name range	
Range arguments	B2:B3	Тип графика	A pie chart

Fig. 64. Example of the "Chart" action setting

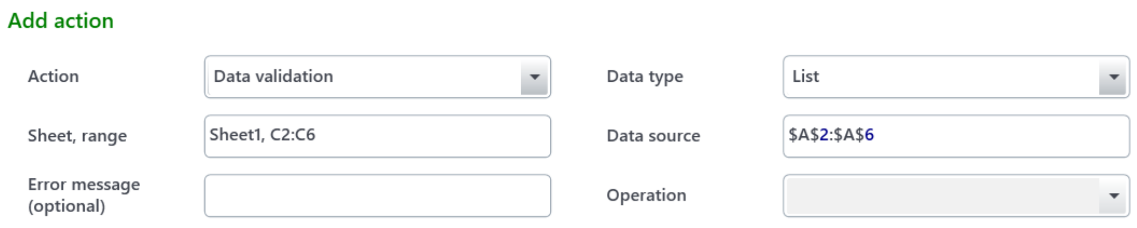
The action supports all available charts that Excel could offer.

Values Checking Action

The action creates a restriction on the types of data that can be entered in the selected cells by selecting the suggested rules. In the "Sheet" field you enter a sheet, in whose cells the restrictions are added, in the "Cells" field you enter the range of cells, on which the validation of values will be performed when entering. In the "Data Type" field you select one of the suggested data types - integer or real number, list, date, text length, time. The "Operation" field consists of the list of available operations for composing the check rule - equal, not equal, greater than, less than or equal, less than or equal, between and outside. When using the data type "List", the filling of the field "Operation" is not required, so it is not available for editing.

The last field changes depending on the data type - for "List" type it will be "Data Source", for all others - "Criteria (w/o ;)". The field "Data Source" is filled with a range of values (within one column or one row), which will be specified in the list (if the list is static, then specify an absolute reference). The "Criteria (w/o ;)" field is filled with the values with respect to which the check rule will be applied.

The "Error message" field is filled with the text of the error that will be displayed if the cell contains a value that doesn't satisfy the rule specified for it. It is an optional field. If it is left empty, the error "The value entered is invalid. The set of values, which can be entered in the cell, is limited.

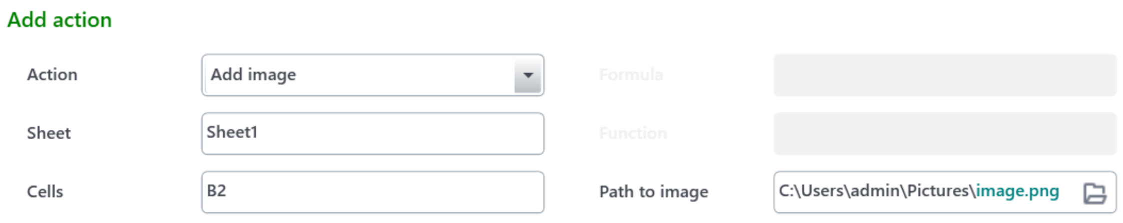

Add action

Action	Data validation	Data type	List
Sheet, range	Sheet1, C2:C6	Data source	$A$2:$A$6
Error message (optional)		Operation	

Fig. 67. An example of the setting of action Value validation

Add image action

The action adds the selected image to the current Excel file. If the image has a transparent background, it will also retain its transparency when added to the Excel file. The "Cells" field specifies the upper left corner of the image location.


Add action

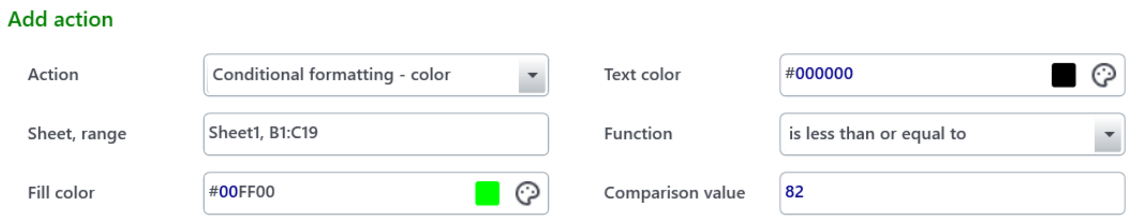
Action	Add image	Formula	
Sheet	Sheet1	Function	
Cells	B2	Path to image	C:\Users\admin\Pictures\image.png

Fig. 72. Example of setting the "Add image" action

Formatting actions group

Conditional Formatting - Color action

The action colors the specified cells on the selected sheet depending on whether they satisfy the specified condition. In the "Sheet, range" field you enter a sheet and range or one cell from this sheet, to which conditional formatting will be applied, separated by commas. The "Fill color" and "Text color" fields contain the colors the cell will be filled with and the text in it will be colored with, if it satisfies the condition. The condition consists of choosing a function and specifying a value for comparison. The available functions are equal, not equal, greater than, less than, greater than or equal to, less than or equal to, formula. If the function "formula" has been chosen, then in the "Value for comparison" field, you should write the formula in Excel format, starting with the "=" sign.

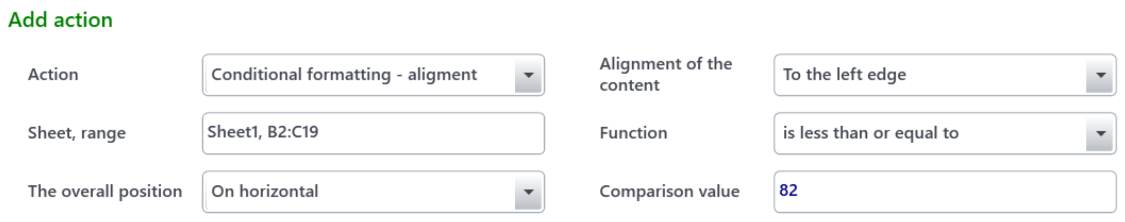

Add action

Action	Conditional formatting - color	Text color	#000000
Sheet, range	Sheet1, B1:C19	Function	is less than or equal to
Fill color	#00FF00	Comparison value	82

Figure 73. Example of setting "Conditional formatting - color" action

Conditional Formatting - Alignment action

The action applies the specified alignment to the specified cells on the selected sheet, depending on whether they satisfy the specified condition. In the Sheet, Range field, you enter a sheet and range or one cell from that sheet, separated by commas, to which the conditional formatting will be applied. In the General Position and Content Alignment fields you select the necessary alignment methods from the drop-down list, which will be applied to the cell if it satisfies the condition. The condition is made by selecting a function and specifying a value for comparison. The available functions are equal, not equal, greater than, less than, greater than or equal to, less than or equal to, formula. If the function "formula" is chosen, then in the field "Value for comparison" you should write the formula in Excel format, starting with the sign "=".

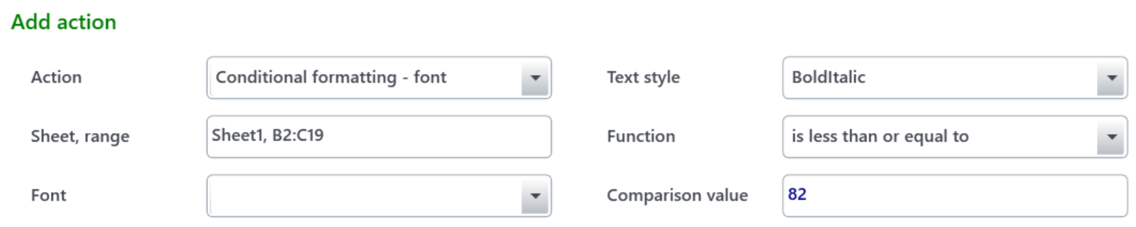

Add action

Action	Conditional formatting - aligment	Alignment of the content	To the left edge
Sheet, range	Sheet1, B2:C19	Function	is less than or equal to
The overall position	On horizontal	Comparison value	82

Figure 76. Example of setting "Conditional formatting - alignment" action

Conditional Formatting - Font action

The action applies the specified font settings (font itself, text style) to the specified cells on the selected sheet, depending on whether they satisfy the specified condition. In the "Sheet, range" field, you enter a comma separated sheet and range or one cell from this sheet, to which the conditional formatting will be applied. In the "Font" field select one of the freely distributed Microsoft fonts, in the "Text style" field select the text formatting - italic, bold and others, if necessary. The selected font and text style will be applied to the cell, if it will satisfy the condition. A condition consists of selecting a function and specifying a value to be compared. The available functions are equal, not equal, greater than, less than, greater than or equal to, less than or equal to, formula. If you have chosen the formula function, then, in the "Value for comparison" field, you should write the formula in Excel format, starting with the "=" sign.

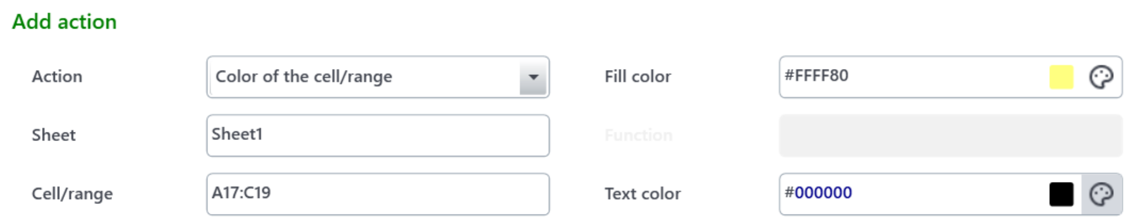

Add action

Action	Conditional formatting - font	Text style	BoldItalic
Sheet, range	Sheet1, B2:C19	Function	is less than or equal to
Font		Comparison value	82

Figure 79. Example of setting "Conditional formatting - font" action

Cell/Range Color Action

This action changes the color of the text fill and the text itself in the specified range/cell.

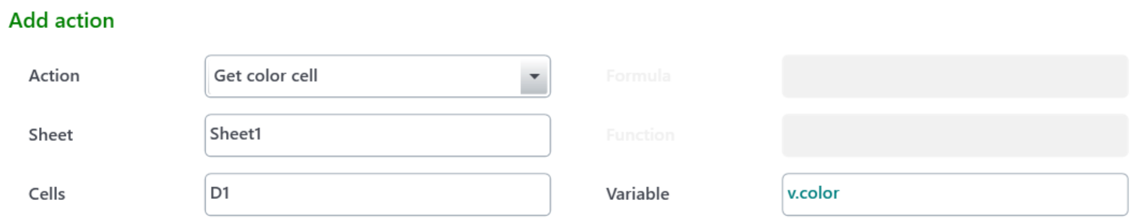

Add action

Action	Color of the cell/range	Fill color	#FFFF80
Sheet	Sheet1	Function	
Cell/range	A17:C19	Text color	#000000

Figure 82. Example of the "Cell/Range Color" action settings

Get cell color action

This action allows you to get the fill color of a cell, returning to the variable its name in HEX format (starting with the "#" character). If you specify a range of cells, the color of the last cell will be returned.

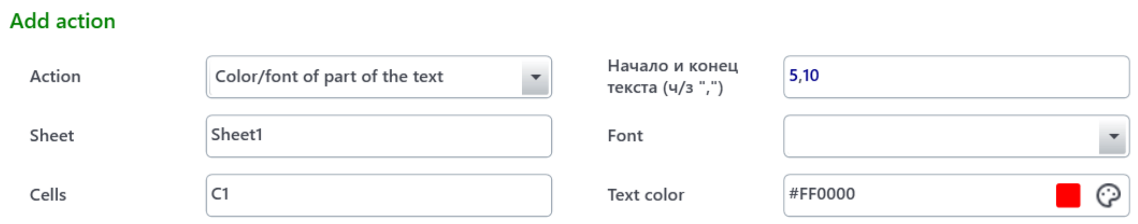

Add action

Action	Get color cell	Formula	
Sheet	Sheet1	Function	
Cells	D1	Variable	v.color

Figure 85. Example of the "Get cell color" action settings

Get color/font of text action

This action allows you to color the text and change its font partially. For this, you need to know the initial character of the coloring and its end (the length of the colored text + the initial character of the coloring), which are specified separated by commas. If you leave the font or color blank, the values that are already inherent to the text will be left intact.

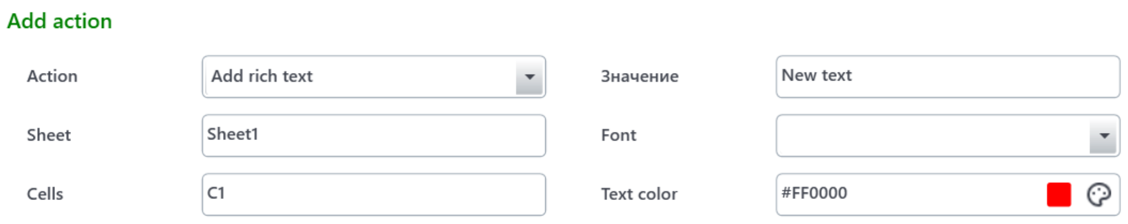

Add action

Action	Color/font of part of the text	Начало и конец текста (ч/з ",")	5,10
Sheet	Sheet1	Font	
Cells	C1	Text color	#FF0000

Figure 85. An example of the color/font setup of a text part

Add formatted text action

This action allows you to add a text to an already filled cell with different formatting.

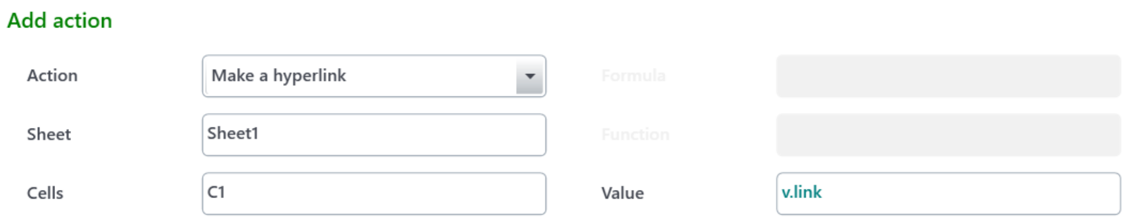

Add action

Action	Add rich text	Значение	New text
Sheet	Sheet1	Font	
Cells	C1	Text color	#FF0000 🟥

Fig. 86. Example of "Add formatted text" action settings

Make hyperlink Action

This action makes an already filled cell a hyperlink. If the "Value" field is filled, the hyperlink will have the name specified in it, otherwise it will be equal to the value in the cell. A range is available, but only one name can be used.

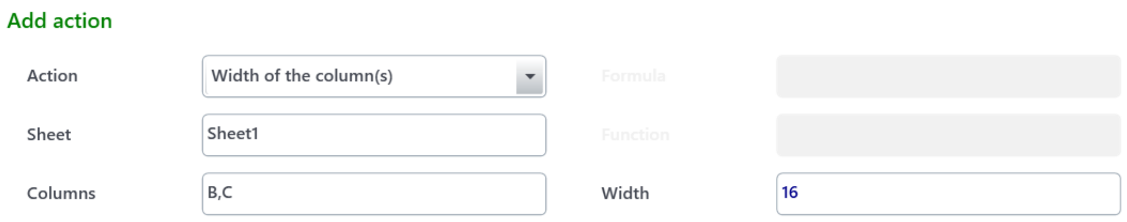

Add action

Action	Make a hyperlink	Formula	
Sheet	Sheet1	Function	
Cells	C1	Value	v.link

Figure 87. Example of setting the "Make hyperlink" action

Width of Column(s) action

The action changes the width of the specified columns. Columns can be listed by comma, write their numbers instead of names (from 1). The width is specified in the same unit used in Excel.

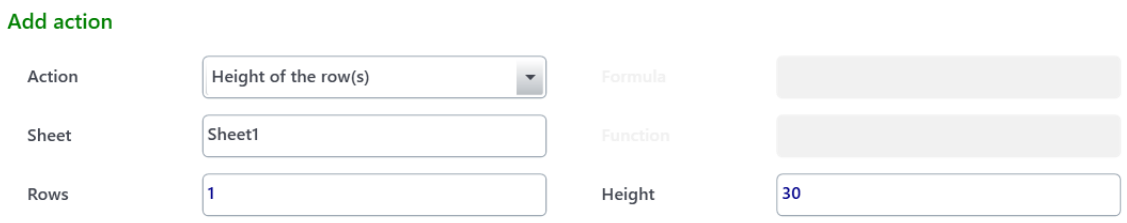

Add action

Action	Width of the column(s)	Formula	
Sheet	Sheet1	Function	
Columns	B,C	Width	16

Figure 88. Example of setting the "Column(s) width" action

Height of Row(s) action

The action changes the height of the specified rows. The same unit of measure as in Excel is used.


Add action

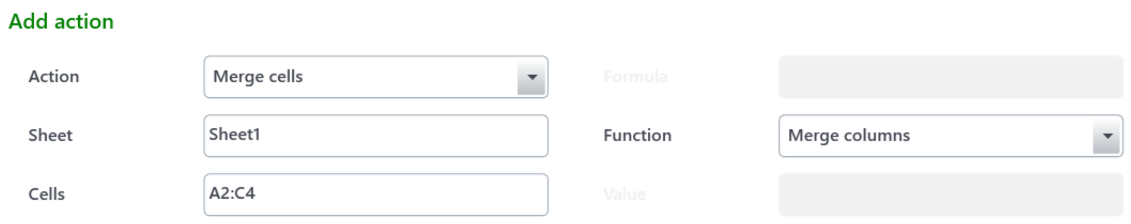
Action	Height of the row(s)	Formula	
Sheet	Sheet1	Function	
Rows	1	Height	30

Figure 91. Example of setting the "Row(s) height" action

Merge Cells action

The action combines the specified cells on the selected sheet. The module provides three types of unification:

- simple merge - that is, both columns and columns of a range are merged;
- union by rows;
- a union by columns.



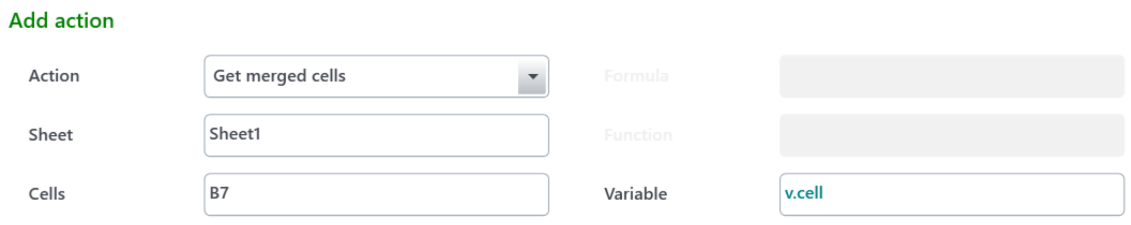
Add action

Action	Merge cells	Formula	
Sheet	Sheet1	Function	Merge columns
Cells	A2:C4	Value	

Fig. 94. Example of configuring the "Merge cells" action

Get merged Cells action

The action takes a cell (or a range of cells) as input and returns null for it (or each of the range) if the cell is not merged with other cells, or the range of the cell it belongs to.



Add action

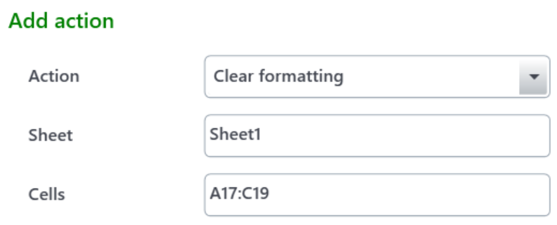
Action	Get merged cells	Formula	
Sheet	Sheet1	Function	
Cells	B7	Variable	v.cell

Figure 97. Example of setting the "Get merged cells" action

For example, for cells "B2", "B3", and "B4" from Figure 96, the result would be "B2:B4" and the result for cell B1 would be null.

Clear Formatting action

The action clears the formatting in the specified range of the selected sheet.



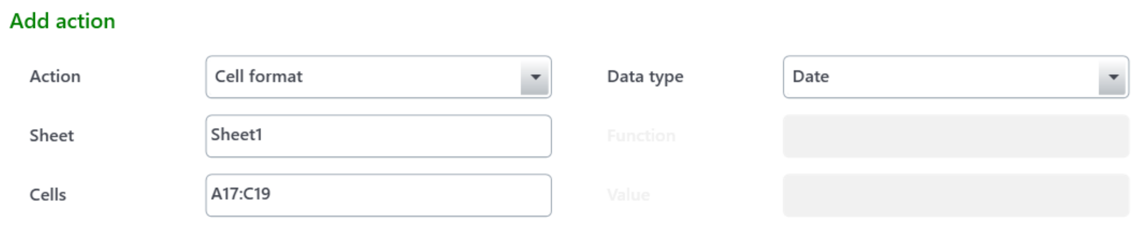
Add action

Action	Clear formatting
Sheet	Sheet1
Cells	A17:C19

Figure 98. Example of setting the "Clear formatting" action

Cell Format Action

The action changes the format of the specified cell to the selected format. By default, all values entered into Excel by the Studio have a string format, so to correctly recalculate formulas or sorting, you must bring the data to the desired format.



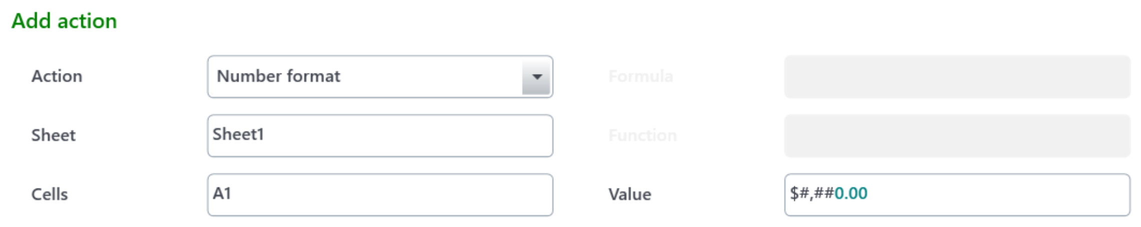
Add action

Action	Cell format	Data type	Date
Sheet	Sheet1	Function	
Cells	A17:C19	Value	

Figure 101. Example of setting the "Format cell" action

Number format Action

This action shows the number according to the specified Excel numeric format rules.

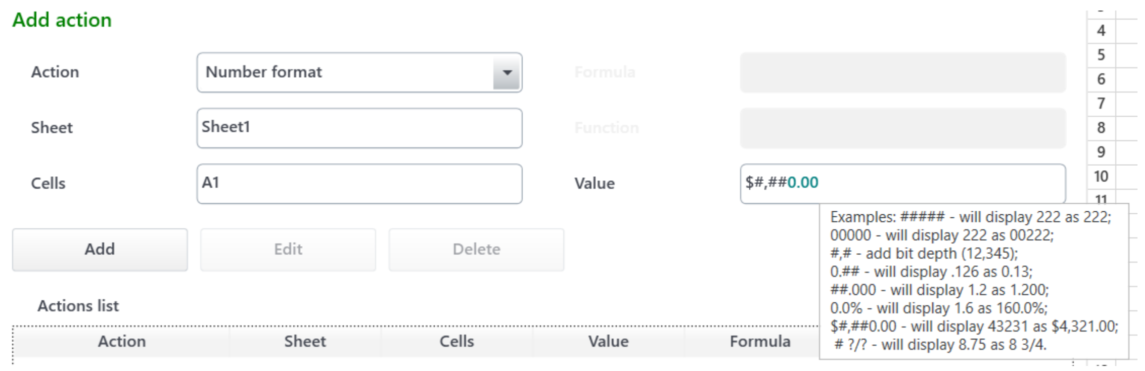


Add action

Action	Number format	Formula	
Sheet	Sheet1	Function	
Cells	A1	Value	$#,##0.00

Fig. 102. Example of "Number format" action settings

Examples of formats can be viewed in Excel by right-clicking the cell with the left mouse button - "Cell Format" - "Number" tab - (all formats):



Add action

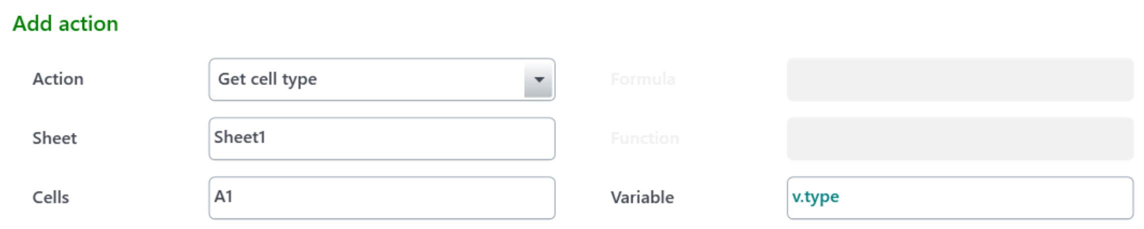
Action	Number format	Formula	
Sheet	Sheet1	Function	
Cells	A1	Value	$#,##0.00

Examples: ##### - will display 222 as 222;
00000 - will display 222 as 00222;
#,# - add bit depth (12,345);
0.## - will display .126 as 0.13;
##.000 - will display 1.2 as 1.200;
0.0% - will display 1.6 as 160.0%;
$#,##0.00 - will display 43231 as $4,321.00;
?/? - will display 8.75 as 8 3/4.

Figure 104. Viewing examples of number formats in the Studio

Get cell type action

This action allows you to get the type of the selected cell or range of cells. The resulting data is one of the strings - "DateTime" (date, time), "Numeric" (any numeric representation, including "money" type), "Text", "None" (in case of empty cell). Otherwise, it is a two-dimensional array of the types of the respective cells.



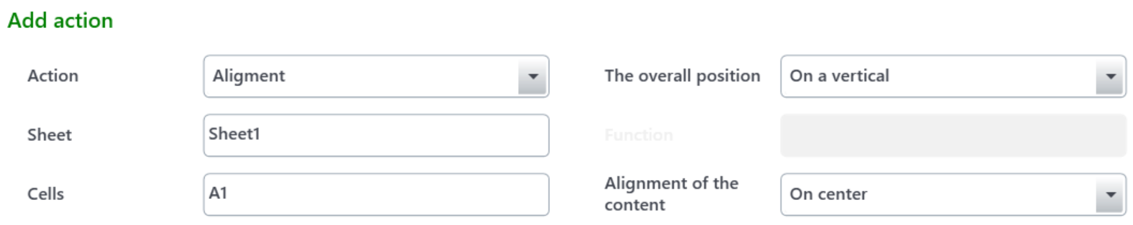
Add action

Action	Get cell type	Formula	
Sheet	Sheet1	Function	
Cells	A1	Variable	v.type

Fig. 107. The example of filling in the "Get cell type" action

Alignment action

The action applies the specified alignment in the specified cells. The "General position" field suggests you to choose one of two alignments - vertically or horizontally to apply the alignment selected in the "Content alignment" field, which, in its turn, suggests you to choose left/right, or top/bottom, or center alignment of the content.

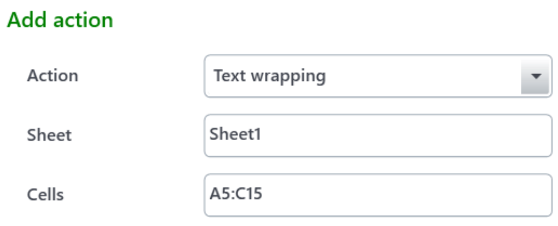

Add action

Action	Aligment	The overall position	On a vertical
Sheet	Sheet1	Function	
Cells	A1	Alignment of the content	On center

Fig. 108. Example of setting the "Alignment" action

Text wrapping action

The action adds to the specified range the "Text transfer" cell add-on, the function of which is to display the cell text in several lines, if it does not fit in one.

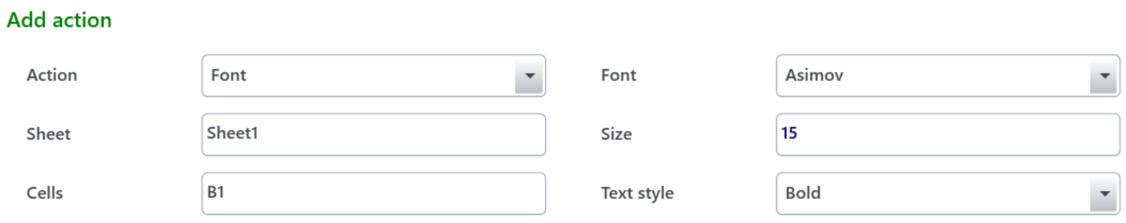

Add action

Action	Text wrapping
Sheet	Sheet1
Cells	A5:C15

Fig. 111. The example of Text wrapping action setting.

Font Action

The action changes the font, style or size of the selected cells on the specified sheet. All the available standard fonts in Microsoft Office package are selected.


Add action

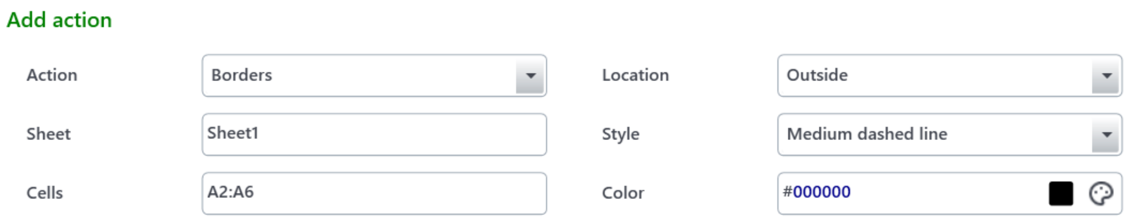
Action	Font	Font	Asimov
Sheet	Sheet1	Size	15
Cells	B1	Text style	Bold

Fig. 112.

Example of setting the "Change font" action

Borders action

The action adds or changes the borders of the selected cells on the specified sheet. It supposes configuring the borders location - external, internal, only right, etc., border color and style - thick, thin, dashed line, etc.

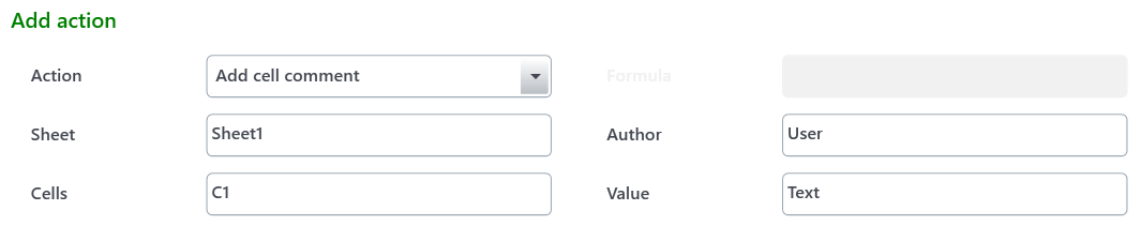

Add action

Action	Borders	Location	Outside
Sheet	Sheet1	Style	Medium dashed line
Cells	A2:A6	Color	#000000 ■

Fig. 115. An example of the Borders action settings.

Add cell comment action

This action allows you to add a note to a cell. It requires the author of the note, its value and the cell to which it is applied. Multiple notes cannot be added to the same cell.

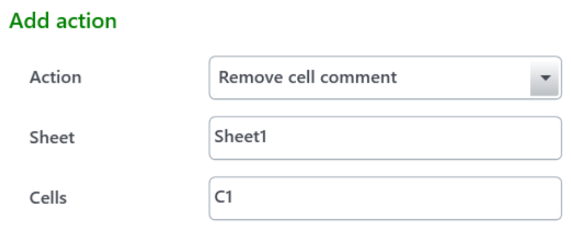


Action	Add cell comment	Formula	
Sheet	Sheet1	Author	User
Cells	C1	Value	Text

Figure 118. Example of configuring the "Add note to cell" action

Remove note to cell action

This action removes a note attached to the specified cell.

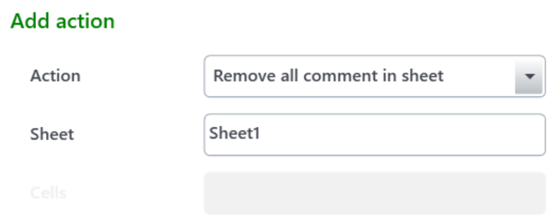


Action	Remove cell comment
Sheet	Sheet1
Cells	C1

Figure 119. Example of setting the "Remove note to cell" action

Remove all notes on the cell action

This action removes all notes from the selected sheet.



Action	Remove all comment in sheet
Sheet	Sheet1
Cells	

Figure 120. Example of the " Remove all notes on the sheet" action settings

WORD module

WORD module provides work with Microsoft Word documents of the Microsoft Office package. The module allows you to create and edit documents with popular text extensions, such as .doc, .docx, and .html.

Module interface

The window of the module consists of three main parts, the first is the part containing fields for creating commands and buttons to control them, the second is the table "List of commands", which will contain all the added actions to work with the file, and the third is the window with the preview of the uploaded file.

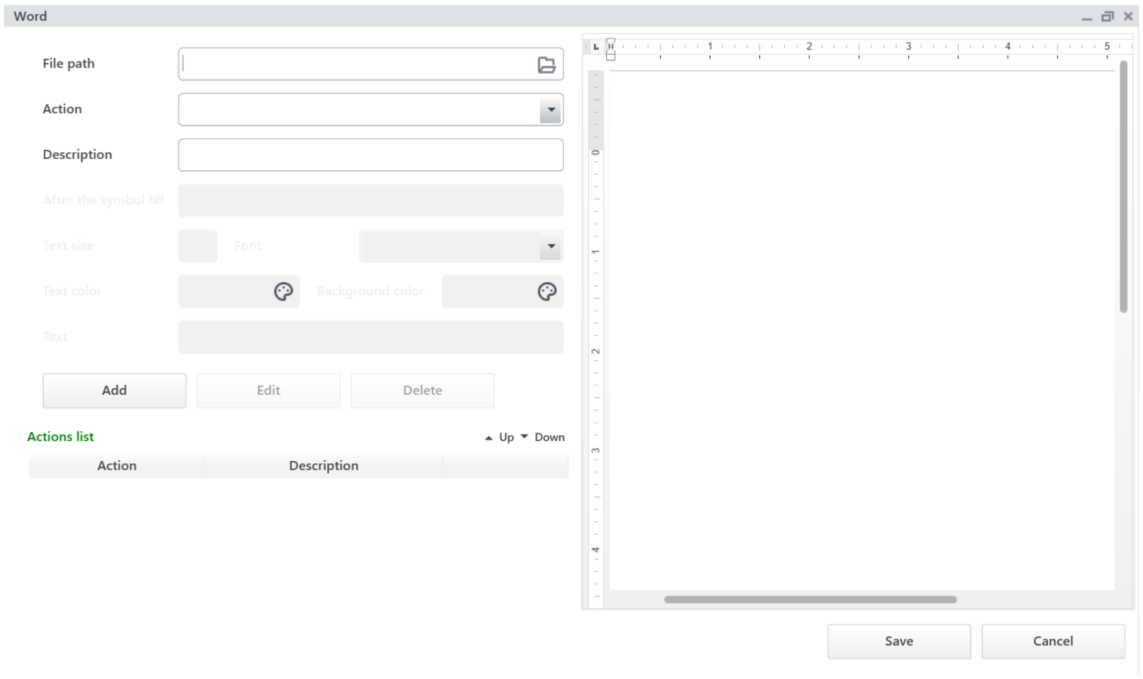


Fig. 1. Window interface

The preview window allows you to view the whole loaded document as it appears in Microsoft Word.

The module's settings section consists of several fields, the accessibility of which varies depending on the actions required for the work. The first three fields are basic and always available, and only two of them are mandatory:

- "File Path" - the field with the button that opens the file selection dialog box. This field is intended for selecting the existing file to work with, or for entering the path to be used for creating the new file. Obligatory field;
- "Action". - drop-down list of actions that can be applied to the uploaded file or create a new one with the path specified in the "File path" field. Obligatory field;
- "Description" - the field, necessary for convenient navigation through the actions, it is filled with the developer's comment about the created action. Optional field.

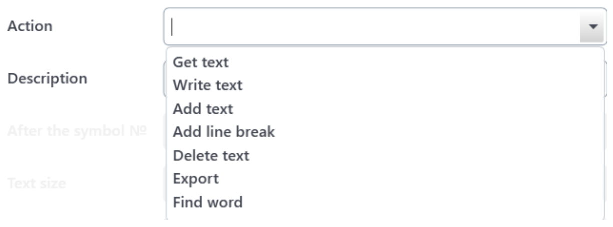
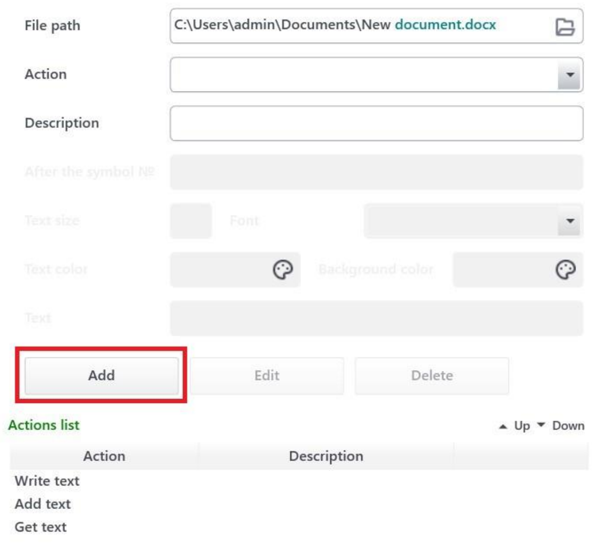


Fig. 2. List of available actions of the module

All possible actions will be described in the next chapter.

You can add an action to the "List of commands" table using the "Add" button under the action setting fields.

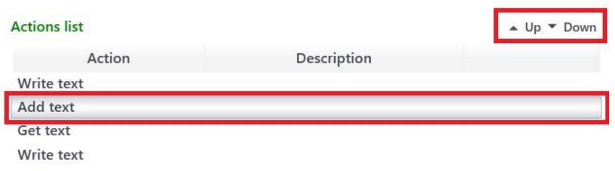

File path: C:\Users\admin\Documents\New document.docx

Action	Description
Write text	
Add text	
Get text	

Fig. 3. Location of "Add" button.

Press the button and the created action will be added to "List of commands". To edit existing action or remove it you should choose the line with the action in the table and press "Delete" to remove it or edit necessary fields of action and then press "Edit".

In order to swap existing actions you have to choose one action and press necessary buttons "Raise" or "Lower" located in the right corner above the table with commands.



Action	Description
Write text	
Add text	
Get text	
Write text	

Fig. 4. Buttons "Raise/Lower" and selection of lines in the table

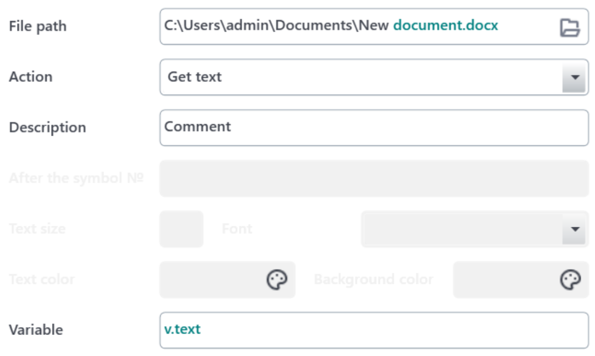
Module actions

Create file

This action creates a file with specified path, name and extension. To set up the action you should fill in the "File path" field with a string like "Path to file/File name. Required extension", for example, "C:/Reports/Documents/Test.docx".

Get text

"Get text" action allows you to read all the text from the loaded file into a variable. To configure the action, there appears another field besides the standard three - "Variable", in which you write the name of the variable (from "v."), in which the read text will be written.

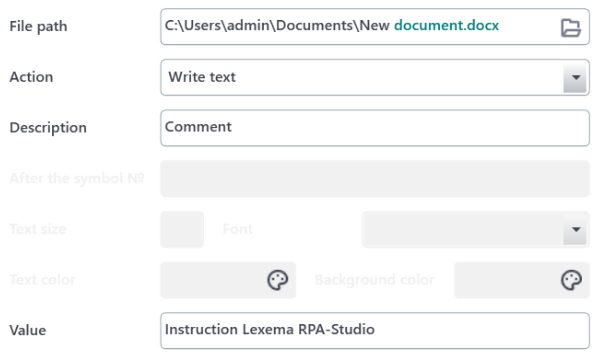


File path	C:\Users\admin\Documents\New document.docx
Action	Get text
Description	Comment
After the symbol №	
Text size	Font
Text color	Background color
Variable	v.text

Fig. 6. Example of setting the "Get text" action

Write text

Using this action you can write the existing text (e.g. read from another text file) to the currently loaded file. For setting it is obligatory to fill in the "Value" field, where either a variable (with "v.") containing the text or the text itself should be entered. Note that the text entered by this action will completely overwrite the existing text in the file.

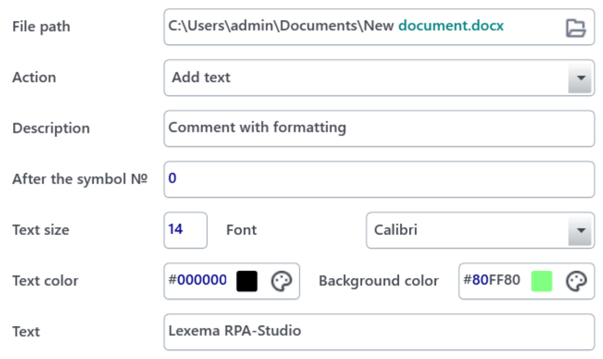


File path	C:\Users\admin\Documents\New document.docx
Action	Write text
Description	Comment
After the symbol №	
Text size	Font
Text color	Background color
Value	Instruction Lexema RPA-Studio

Fig. 7. Example of setting the "Write text" action

Add text

This action allows you to add a text to an existing text with the necessary formatting. All fields of the module are opened for editing. Each field, other than the main three, will be described in detail below.



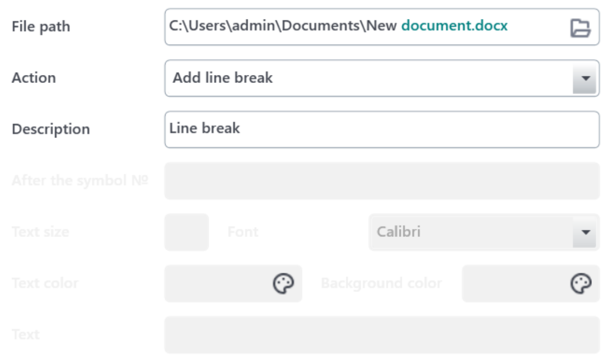
File path	C:\Users\admin\Documents\New document.docx		
Action	Add text		
Description	Comment with formatting		
After the symbol №	0		
Text size	14	Font	Calibri
Text color	#000000	Background color	#80FF80
Text	Lexema RPA-Studio		

Fig. 8. Example of the "Add text" action settings

- "After the symbol №" - This field expects to enter an integer number denoting the character after which the text should be inserted (including whitespace and line break characters or page break characters).
- "Text size" - similar to the field "Font size" in Microsoft Word - sets the size of the added text. Entry of an integer or decimal number is expected.
- "Font" - the dropdown list, which contains a set of standard fonts of the Microsoft Office package - sets the font of the added text.
- "Text Color" and "Background Color" - fields with a palette icon on the side, which opens a color selection dialog box - sets the color of the added text and its background, respectively. It is entered in HEX color format - hexadecimal representation of RGB.
- "Text" - the field that specifies the added text or the variable that contains it.

Add Line Break

This action adds a line break character to the file. No additional fields are required.

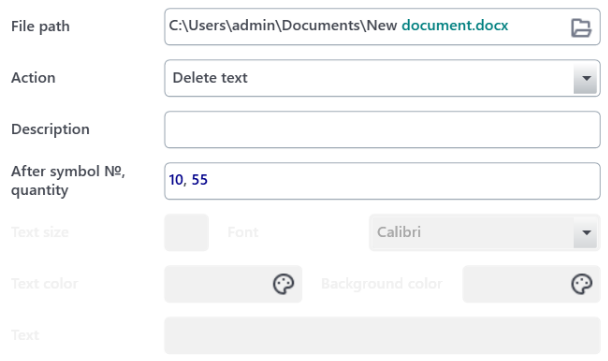


File path	C:\Users\admin\Documents\New document.docx		
Action	Add line break		
Description	Line break		
After the symbol №			
Text size		Font	Calibri
Text color		Background color	
Text			

Figure 9. Example of action settings: add line break

Delete text

Using this action you can delete the text, knowing the number of characters in it and its location. To set it up you should enter in the field "After character #, number" two whole numbers separated by comma, where the first number is the number of character after which you want to remove the text and the second number is the number of characters in the text you want to remove.

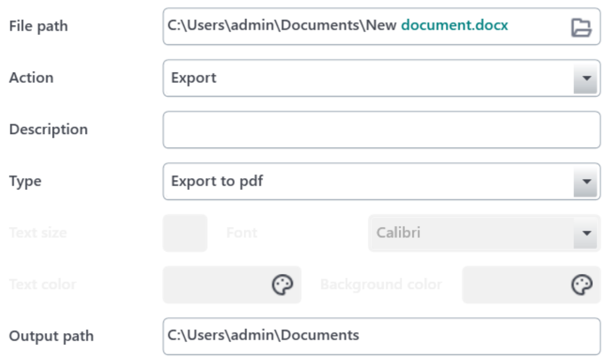


File path	C:\Users\admin\Documents\New document.docx
Action	Delete text
Description	
After symbol №, quantity	10, 55
Text size	
Font	Calibri
Text color	
Background color	
Text	

Fig. 10. Example of setting the "Delete text" action

Export

With this action, you can change the extension of the loaded file. To configure the action, select from the drop-down list "Type" the type of file to be exported, and specify the path with the name and extension of the future file in the "Output file path" field.



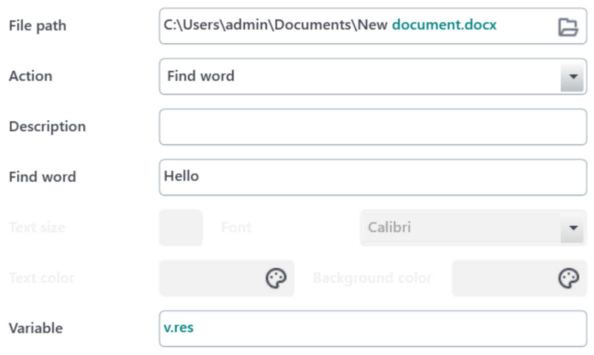
File path	C:\Users\admin\Documents\New document.docx
Action	Export
Description	
Type	Export to pdf
Text size	
Font	Calibri
Text color	
Background color	
Output path	C:\Users\admin\Documents

Fig. 11. Example of action settings "Export"

Find word

This action searches for a word or phrase (its exact match) in the text and returns all its occurrences in the text as an array of objects (or a table with named columns). Returned fields:

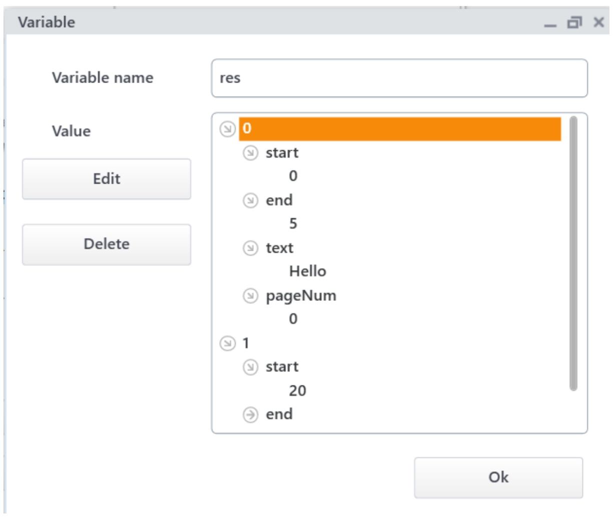
- start - start character of the word or phrase;
- end - the end character of the word or phrase;
- text - the word or phrase together with its characters;
- pageNum - number of the page where the word or phrase was found.



File path	C:\Users\admin\Documents\New document.docx
Action	Find word
Description	
Find word	Hello
Text size	
Font	Calibri
Text color	
Background color	
Variable	v.res

Fig. 12. Example of setting the "Find word" action

Search result will look like the following:



```
res
0
  start
    0
  end
    5
  text
    Hello
  pageNum
    0
1
  start
    20
  end
```

Fig. 13. Example of result variable

FILE module

FILE module is designed to change the location of files, copying, archiving, creating a folder and transferring files into it, in general, to manipulate files and directories.

Module interface

The module's window combines three groups of elements.

The first group of elements - a set of fields and buttons for creating a command. The second - the list of commands. And the third - is the area for previewing text files.

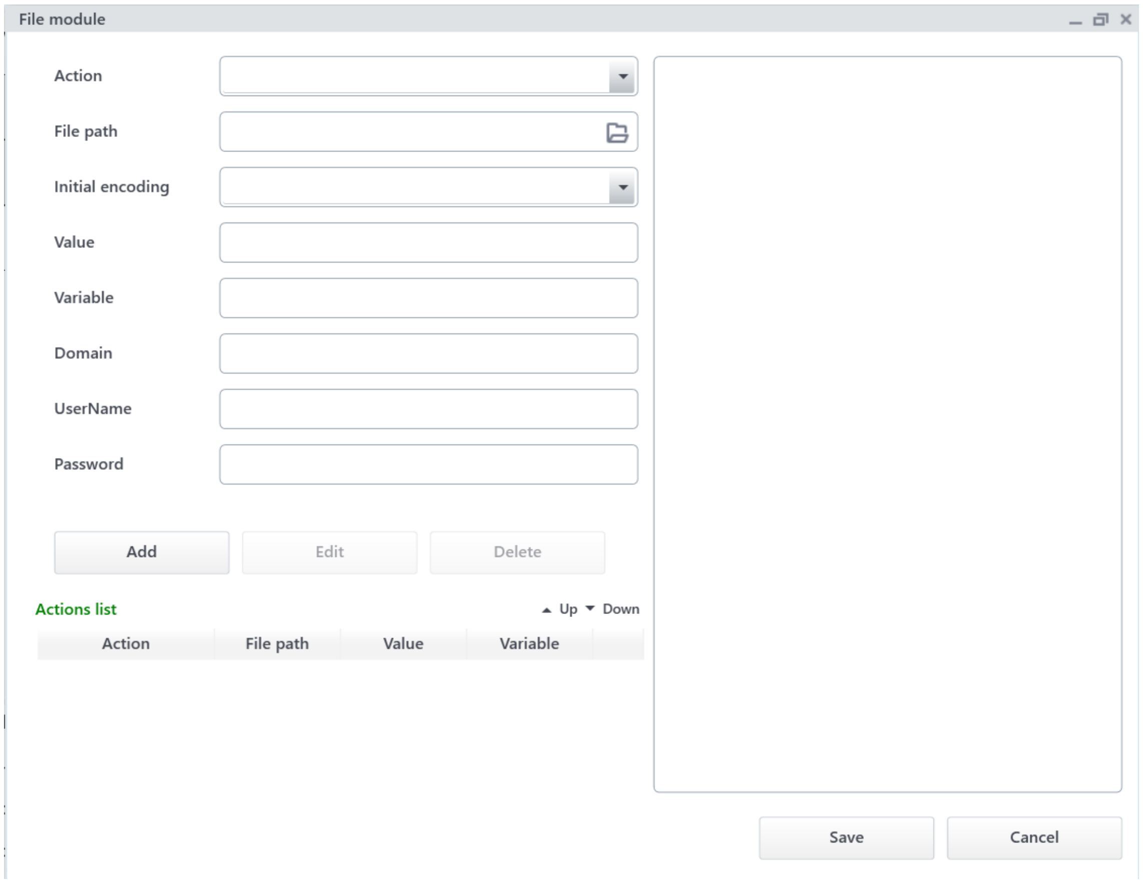


Fig. 1. Module window

The set of available fields changes dependы on the selected action.

Module functions

Create command

A command is created as follows - choose the necessary action, fill out the other fields - press the "Add" button to add the command to the "List of commands".

Edit command

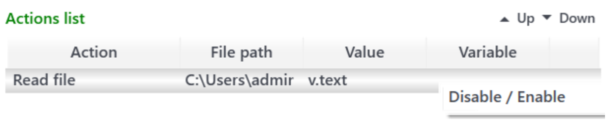
To edit a command, select it in the list of commands by clicking with the left mouse button. After that all fields in the upper left part of the "Working with files" window will be filled out according to the chosen command. Change the fields that you want to edit and, making sure that the desired action is still selected in the list of commands, click on the "Edit" button.

Delete command

To delete a command, select it in the list of commands and click on the "Delete" button.

Disable/enable commands

Created commands can be disabled and enabled again. Disabled commands will not be executed. You can do this by clicking on the line corresponding to the command to be disabled with the right mouse button and choosing "Disable/enable".



Actions list

▲ Up ▼ Down

Action	File path	Value	Variable
Read file	C:\Users\admir	v.text	

Fig. 2. The command context menu

Selecting a file/folder

The 🗀 button in the populated fields opens a standard file or folder selection dialog box, depending on the selected action.

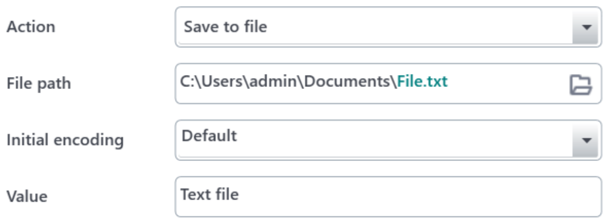
Module actions

The module contains 19 actions that allow you to work with folders and files, each of which consists of a different set of fields for settings. All actions of the module are described below.

The module has the possibility to work with network folders, for this purpose the "Domain", "User" and "Password" fields are provided. The rest of the fields are filled out according to the chosen action.

Save to file Action

Action overwrites existing text file, writing into it a string, specified in field "Value".

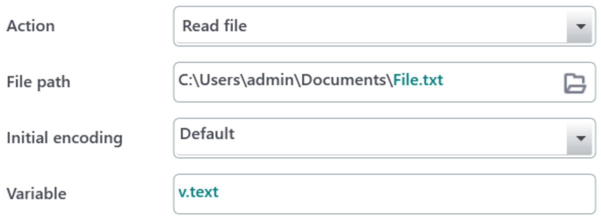


Action	Save to file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Value	Text file

Fig. 3. Example of action settings "Save to file"

Read file Action

The action reads all the text from the selected file and writes it into the specified variable.



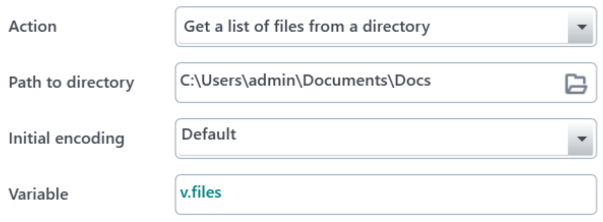
Action	Read file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Variable	v.text

Pic. 4. Example of "Read file" action settings

Field "Initial encoding" is in charge of chosing the encoding, which will be used while reading from the file.

Get list of files in the folder Action

The action creates the variable list, which contains paths to each file in the folder.

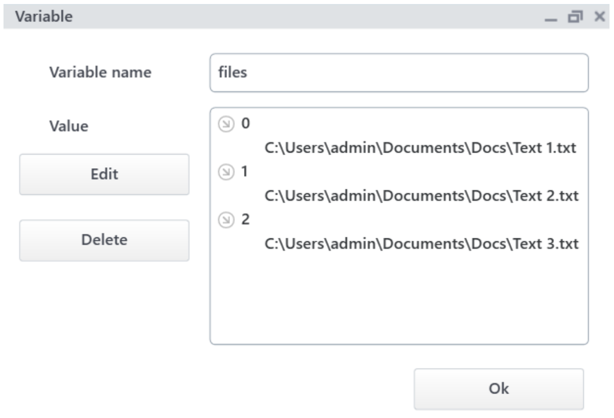


Action	Get a list of files from a directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Variable	v.files

Figure 5. Example of action settings "Get file list".

Let's assume that we have a Robot folder that contains 3 Excel files. Then, in Excel, we can work with each of the files, for example, to work with the first one, we should write v.files[0] in the "File Name" field.

The variable after performing the action will look like Figure 6:



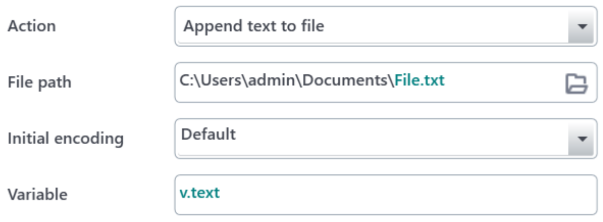
Variable

Variable name	files
Value	0 C:\Users\admin\Documents\Docs\Text 1.txt 1 C:\Users\admin\Documents\Docs\Text 2.txt 2 C:\Users\admin\Documents\Docs\Text 3.txt

Figure 6. Example of variable, containing list of files

Add text to file Action

The action adds text to the existing text in the file without overwriting its data.

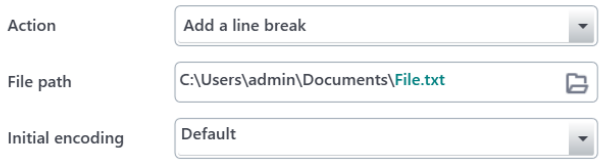


Action	Append text to file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Variable	v.text

Fig. 7. Example of action settings "Add text to file".

Add line break action

This action adds a line break in the file text.



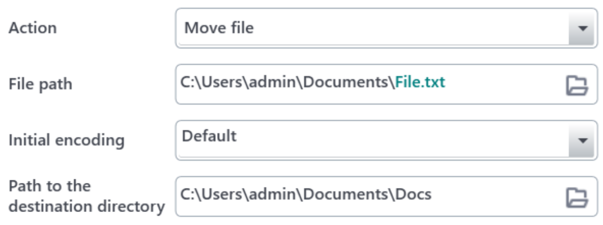
Action	Add a line break
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default

Fig. 8. Example of "Add line break" action settings

Move file Action

The action moves the selected file to the selected folder. In the "Variable" field you can, but not necessarily, enter the name of the variable where the new location to the file will be returned (in the example below it is C:\Users\user\Desktop\Documents\text.docx).

If the file already exists in the destination directory - it will be overwritten.



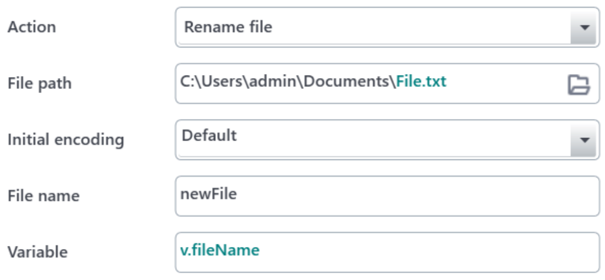
Action	Move file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Path to the destination directory	C:\Users\admin\Documents\Docs

Fig. 9. Example of "Move file" action settings

Rename file Action

The action changes the name of the selected file to the one, specified in "File name" field.

If the file with the same name already exists - it will be overwritten.



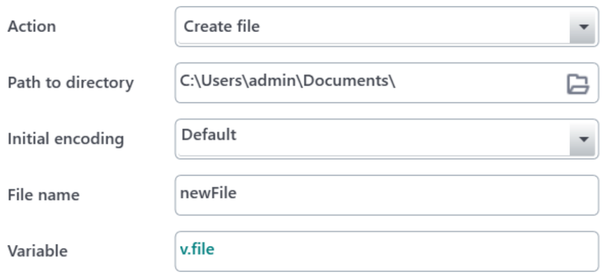
Action	Rename file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
File name	newFile
Variable	v.fileName

Fig. 10. Example of "Rename file" action settings

Action "Create file" creates a new text file with the same name.

The action creates a new text file with the default extension .txt. If any other extension is specified, the file will be created with the proper extension. If the "Variable" field is filled in, the location of the created file will be returned to the variable specified in it.

If the file with that name already exists - it will be overwritten.

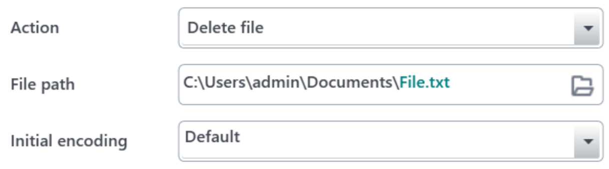


Action	Create file
Path to directory	C:\Users\admin\Documents\
Initial encoding	Default
File name	newFile
Variable	v.file

Fig. 11. Example of "Create file" action settings

Delete file Action

This action deletes the selected file.



Action	Delete file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default

Fig. 12. example of "Delete file" action settings

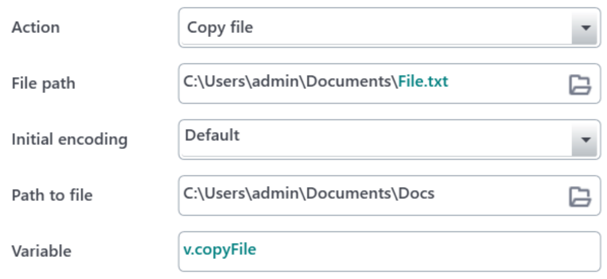
Copy file Action

The action copies the selected file to the specified directory.

In the "Path to destination file" field you must enter the path to the folder where the file is to be copied and its name with an extension as if the file already exists. The new name may be different from the original one.

If the "Variable" field is filled in, the location of the new file will be returned to the variable specified in it.

If the file with this name already exists, it will be overwritten.

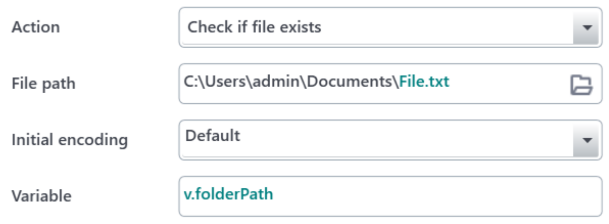


Action	Copy file
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Path to file	C:\Users\admin\Documents\Docs
Variable	v.copyFile

Fig. 13. Example of making the "Copy file" action settings

Check if file exists Action

The action checks if the stated file exists and writes the result into a variable. The result is of bool type. i.e. true if the file exists or false otherwise.



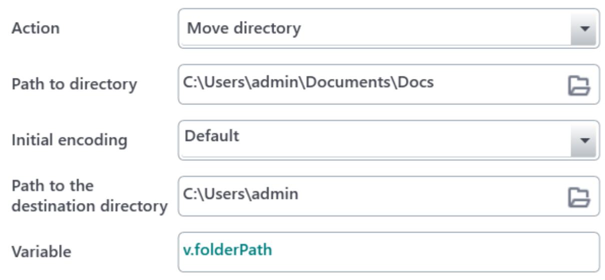
Action	Check if file exists
File path	C:\Users\admin\Documents\File.txt
Initial encoding	Default
Variable	v.folderPath

Fig. 14. Example of action settings "Check file existence".

Move folder Action

The action moves the folder specified in the "Path to folder" field to the directory specified in the "Path to destination folder" field. If the "Variable" field is filled, the new location of the moved folder will be returned to the variable specified in it.

If the folder with such name already exists in the target folder, the program will generate an error and will not move the folder.



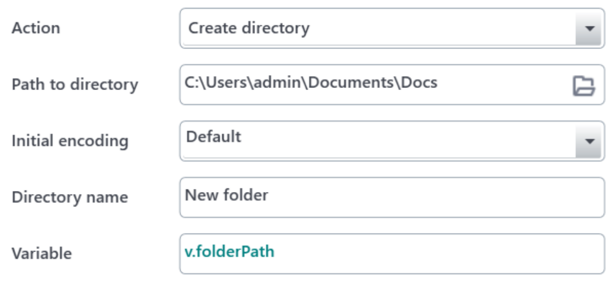
Action	Move directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Path to the destination directory	C:\Users\admin
Variable	v.folderPath

Figure 15: Example of setting the "Move folder" action

Create folder Action

The action creates a folder of a specified name in the selected directory. If the field "Variable" is filled, the location of a new folder will be returned to the variable, specified in it.

If the folder to be created already exists - the program will not create a new folder and will not generate an error. Check the presence of the folder before creating it with other actions of the module.

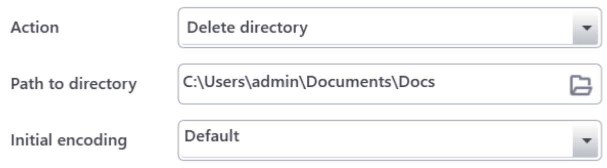


Action	Create directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Directory name	New folder
Variable	v.folderPath

Figure 16. Example of action of creating a folder

Delete folder Action

This action deletes the selected folder.

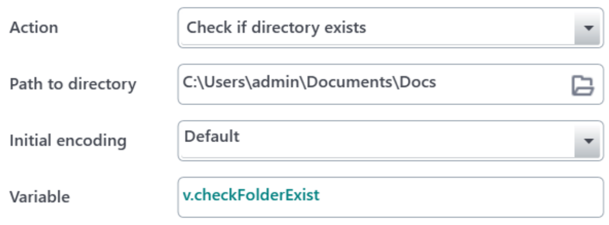


Action	Delete directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default

Figure 17. Example of action settings: "Delete folder".

Check if the folder exists action

This action checks existence of a selected folder and stores the result to a variable. The result is of bool type, i.e. true - if the folder exists, otherwise false.

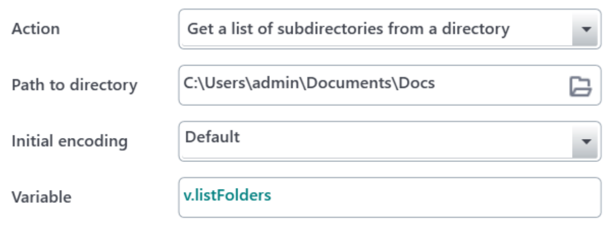


Action	Check if directory exists
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Variable	v.checkFolderExist

Fig. 18. Example of configuring the "Check that a folder exists" action

Get list of folders in a folder action

The action creates a variable list, which contains paths to each file in the folder.



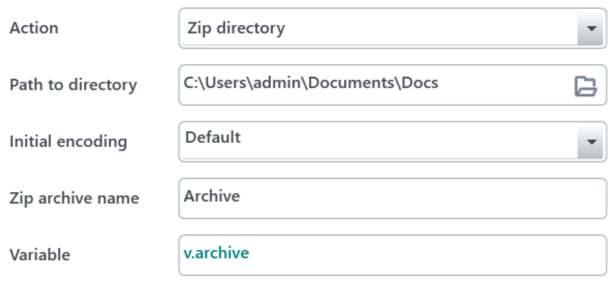
Action	Get a list of subdirectories from a directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Variable	v.listFolders

Fig. 19. Example of action settings "Check existence of folder"

Check up existence of folder Action

The action creates ZIP archive with specified name, adding into it contents of the selected folder. If the "Variable" field is filled, the location of the created archive will be returned to the variable specified in it.

If the archive with the same name already exists, the program will return an error.

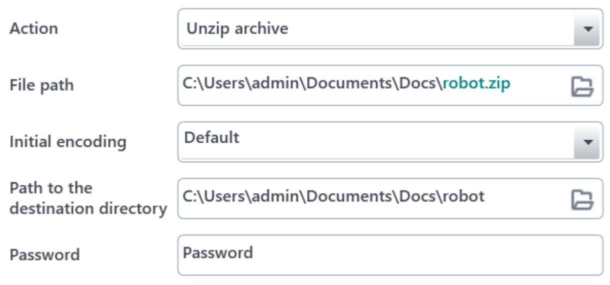


Action	Zip directory
Path to directory	C:\Users\admin\Documents\Docs
Initial encoding	Default
Zip archive name	Archive
Variable	v.archive

Fig. 20. Example of setting the "Archive folder" action

Extract archive action

The action unpack archives in the specified folder. The program supports the following archive types: 7z, zip. If the password is set, it is entered into appropriate field "Password".

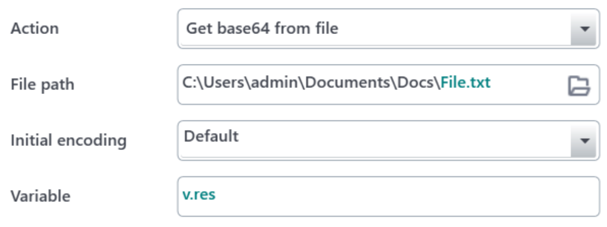


Action	Unzip archive
File path	C:\Users\admin\Documents\Docs\robot.zip
Initial encoding	Default
Path to the destination directory	C:\Users\admin\Documents\Docs\robot
Password	Password

Fig. 21: Example of setting the "Extract archive" action.

Get base64 from file Action

This action converts the file into a Base64-encoded string. It may be useful when loading the file into the database, sending in Web requests and so on.



Action	Get base64 from file
File path	C:\Users\admin\Documents\Docs\File.txt
Initial encoding	Default
Variable	v.res

Fig. 22. Example of setting the "Get base64 from file" action

PDF module

This module allows you to work with PDF files. Its capabilities include obtaining text from the document, images, and merging multiple PDF files.

Module interface

The module consists of 4 fields, the availability of which is determined by the selected action, buttons to control the created action and table-list of created actions.

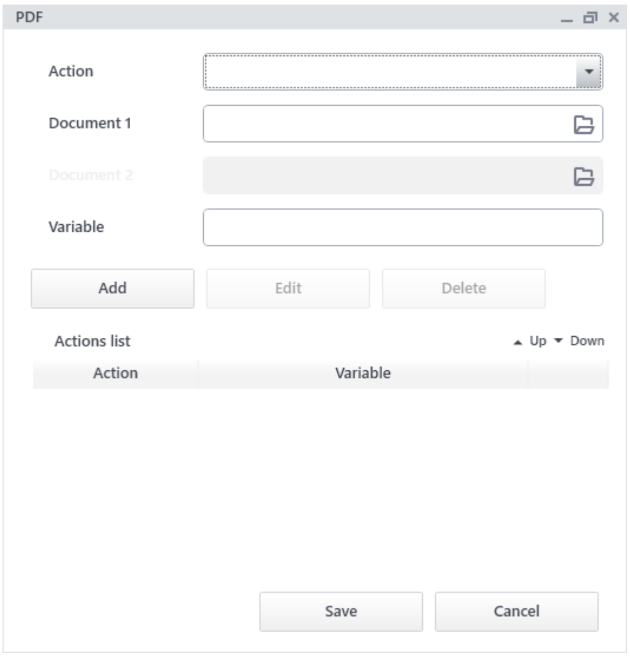


Fig. 12. PDF module interface

"Action". This is a drop-down list of actions available in the module.

"Document 1" field. The field where you enter the path to the working file. The button in the field opens a file selection dialog.

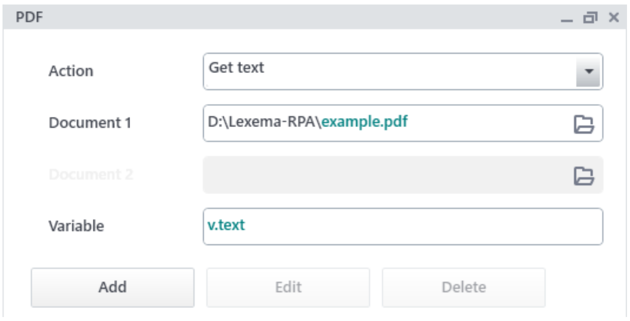
"Document 2" field. The field in which you enter the path to the second working file is used for actions requiring several files (e.g. merging 2 PDFs). The button in the field opens a file selection dialog box.

Variable field. The field, to enter the name of the variable, which will return the result of the module.

List and description of actions

Get text

The action allows you to get text from a PDF file, if it is a text file, not an image. The resulting text will be written to the specified variable.

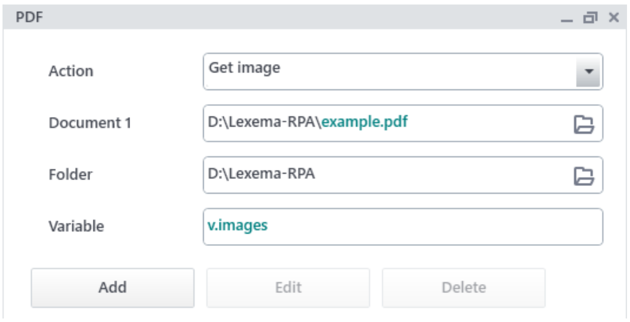


Action: Get text
Document 1: D:\Lexema-RPA\example.pdf
Document 2
Variable: v.text

Fig. 13. Example of setting the "Get text" action

Get image

The action gets all the images from the PDF file and saves them in .jpg format to the directory specified in the "Folder" field. A list of paths to the selected images will be written to the specified variable.



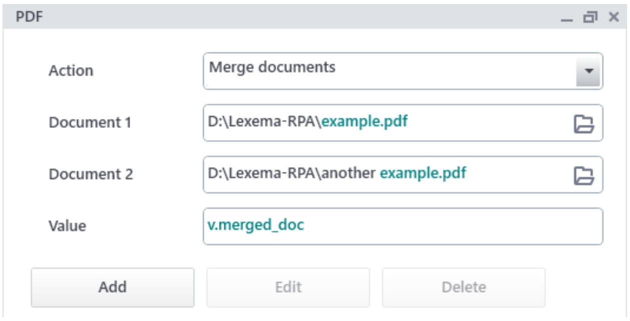
PDF

Action	Get image
Document 1	D:\Lexema-RPA\example.pdf
Folder	D:\Lexema-RPA
Variable	v.images

Fig. 14. Example of setting the "Get image" action

Merge documents

The action merges the two specified PDF files and saves them in a file, which you should specify in the "Value" field.



PDF

Action	Merge documents
Document 1	D:\Lexema-RPA\example.pdf
Document 2	D:\Lexema-RPA\another example.pdf
Value	v.merged_doc

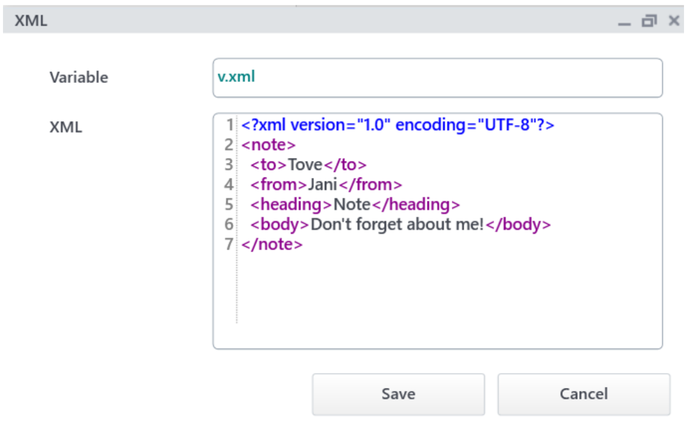
Fig. 13. Example of setting the "Merge documents" action

XML module

XML module is designed for parsing XML text. The result of the module's work is an object with fields and properties specified in the source text.

Module interface

The module window consists of two fields - the "Variable" and "XML" fields.



XML

Variable: v.xml

XML:

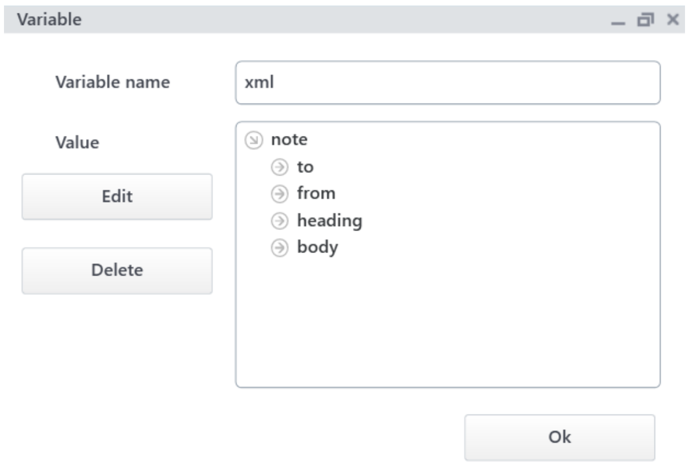
```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Note</heading>
  <body>Don't forget about me!</body>
</note>
```

Fig. 10. XML Module window

Field "Variable" is filled with the name of the variable, in which the object - the result of module's work - will be returned.

The "XML" field is filled with the text in XML format or with the variable, containing this text.

The result of the module when configured as shown in Figure:



Variable name: xml

Value:
- note
  - to
  - from
  - heading
  - body

Fig. 11. XML parsing result

TEXT ANALYSIS module

TEXT ANALYSIS module offers several methods for comparing text. Some tasks require knowledge of how much difference the text makes between the reference and the specified text. These are tasks related to computer linguistics and artificial intelligence.

Module Interface

The module window consists of a command composition part, command control buttons, and a list of commands in the form of a table. The command composition part consists of the following fields: the "Action" drop-down list with available set of text analysis methods, two fields "Line 1" and "Line 2" to enter two lines of text or variables containing text, and "Result Variable" field - to name the variable in which the result of module's work will be placed.

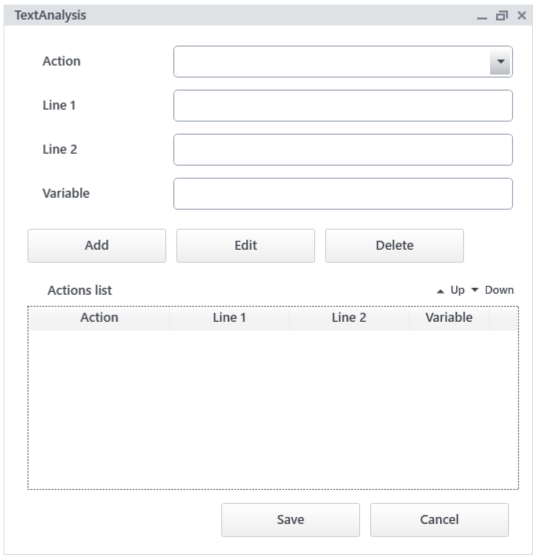


Fig. 1. Module window

If you press the "Add" button the created command will be placed to the "List of commands" table. If you want to edit the command from the list you should choose it in the table, change necessary fields and press the "Edit" button. To delete a command you have to select it in the list and press the button "Delete". Using the arrows in the corner of the table you can swap commands.

Methods of text analysis

The module offers the following methods of text analysis:

- *Levenshtein's disambiguation* - calculates the difference between two lines. For example - "Lexema RPA" and "Lexema SR" differ by 3 characters - the words "Lexema" coincide completely, the other characters are different, that is, the result written in the variable will be equal to 3;
- *3-grams* - a method based on working with n-grams, in our case n=3 - the similarity of every 3 characters is evaluated. The higher the number (up to 1), the greater the similarity of the strings. In the example "Lexema RPA" and "Lexema SR" the result will be the number 0.52.
- *Jarot-Winkler similarity* is a measure of string similarity to measure the distance between two character sequences. The smaller the Jaroe-Winkler distance for two strings, the more similar the strings are to each other. For the example "Lexema RPA" and "Lexema SR" the result is a number 0.5.

TEXT RECOGNITION modules

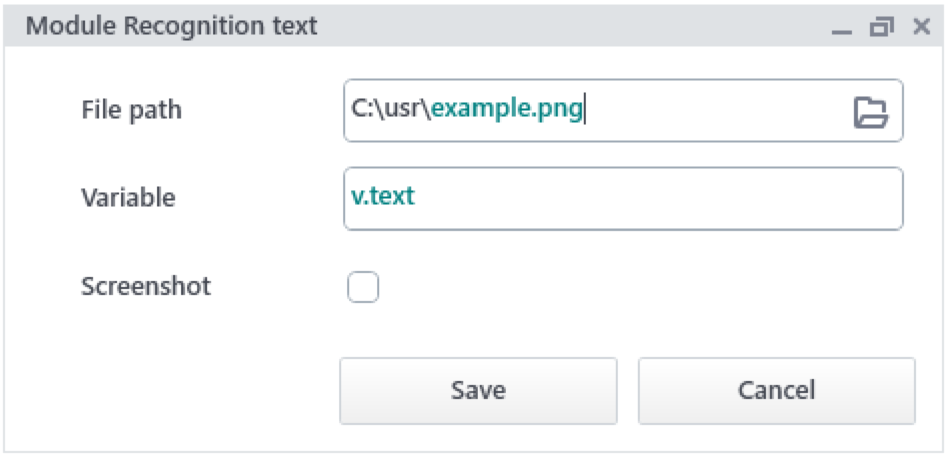
Tesseract OCR

Based on free OCR library https://tesseract-ocr.github.io/

TESSERACT OCR module is designed to read text from a specified image and present the resulting text as an object.

Module Interface

The module has two fields - "Path to file" and "Variable", and a checkbox "Screenshot".


Module Recognition text

File path: C:\usr\example.png

Variable: v.text

Screenshot: ☐

Fig. 2. Module window

“Path to file" filed is meant to be filled with the path to the file you want to recognize. It can be an image or a PDF document.

The "Variable". Filled in with the name of the variable, starting with "v.", in which the result of the recognition will be placed.

Checkbox "Screenshot". The checkbox is checked when you want to recognize a screenshot of the screen when this module is running during the execution of the robot. In this case, you don't need to specify the "File path" field.

The structure of the resulting object is an array of document sheets, each element of which is an object consisting of two fields - number and words. The number field contains the sheet number of the document (from the 1st page), the words field is an array of objects of all the words from the page. The structure of the word object is value, x, y, page. The value contains the word, the x and y coordinates of the word x and y respectively, and the page is the page number on which the word is contained.

EXAMPLE

Suppose that we have the following image in jpeg format. Feed it into the text recognition module.

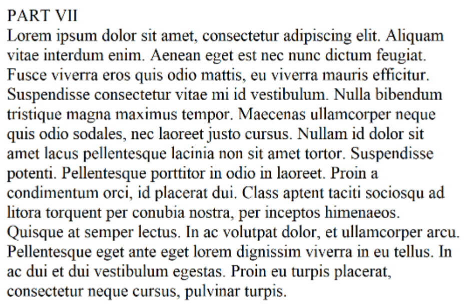
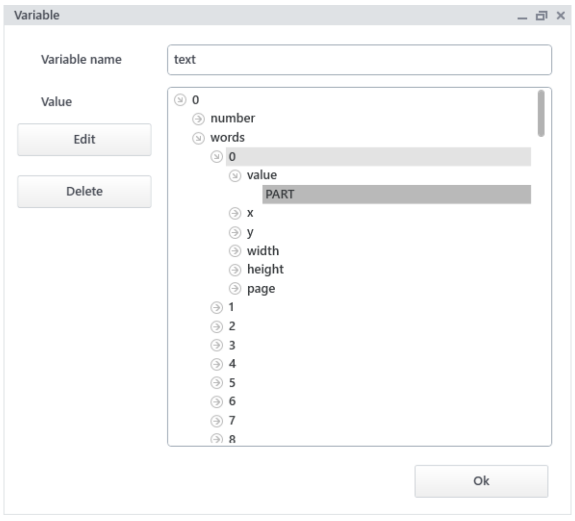

PART VII
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam
vitae interdum enim. Aenean eget est nec nunc dictum feugiat.
Fusce viverra eros quis odio mattis, eu viverra mauris efficitur.
Suspendisse consectetur vitae mi id vestibulum. Nulla bibendum
tristique magna maximus tempor. Maecenas ullamcorper neque
quis odio sodales, nec laoreet justo cursus. Nullam id dolor sit
amet lacus pellentesque lacinia non sit amet tortor. Suspendisse
potenti. Pellentesque porttitor in odio in laoreet. Proin a
condimentum orci, id placerat dui. Class aptent taciti sociosqu ad
litora torquent per conubia nostra, per inceptos himenaeos.
Quisque at semper lectus. In ac volutpat dolor, et ullamcorper arcu.
Pellentesque eget ante eget lorem dignissim viverra in eu tellus. In
ac dui et dui vestibulum egestas. Proin eu turpis placerat,
consectetur neque cursus, pulvinar turpis.

Fig. 3. Image submitted to OCR

The result written to the variable when the image is recognized:


Variable

Variable name: text

Value

```
0
  number
  words
    0
      value
        PART
      x
      y
      width
      height
      page
    1
    2
    3
    4
    5
    6
    7
    8
```

Fig. 4. Example of filling a variable

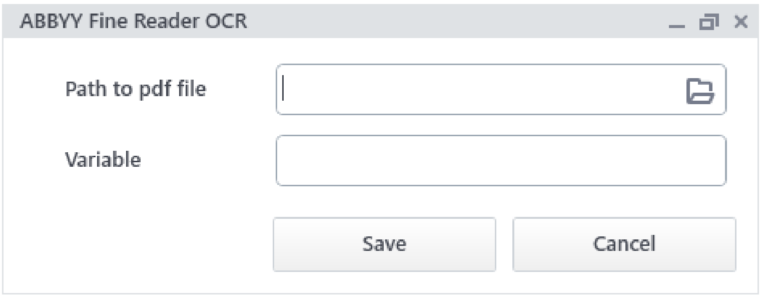
The result contains the information about the pages, coordinate and value of each word in the image.

Example of getting some word from the second page of the document: v.text[1].words[100].value, where v.text - variable, that stores the result of the module, [1] - page number in the document (numbering starts with 0 for list variables), [100] - the 101th word in the file, value - the value of the word itself.

FROCR

Module interface

This module consists of two fields - "Path to PDF file" and "Variable".


ABBYY Fine Reader OCR

Path to pdf file

Variable

Fig. 5. Module window

The "Path to PDF file" field. This field contains the path to the recognized file/image.

The "Variable" field. You can enter the variable name, starting with "v.", in which the result of the recognition will be placed.

A PDF file can consist of several pages, so the variable will be a list of pages. For example, v.text[0] is the first recognized page of the file.

The entire recognized text is divided into two parts: the main text and the table part.

Plain text

To refer to the text, you need to call the rows_word[0] field (in this entry, [0] means you are referring to the first text part, other ABBYY versions may use a larger text breakdown). Then you specify, as a separate field, the line number and, separated by a dot, the number of the word in the line - v.text[0].rows_word[0].2.4. Then, after selecting the desired word, you can get its value by accessing the field value - v.text[0].rows_word[0].2.4.value - such line will give the value of the word on the first page of the recognized document, the second line being the fourth on the left.

Tables

To access the table, you need to refer to the field tables[0] ([0] is necessary for the correct work, its semantic part is embedded in the ABBYY product). Then, you need to enter a table number in a separate field, since you can have several tables on a page - v.text[0].tables[0].0. After that, you enter the cell number in the table as a single number. If you want to find a cell and know its column and row number, you can get the index of the cell in question variable as "row,column" by accessing the field index - v.text[0].tables[0].0.5.index (5 - the fifth cell in the array of recognized cells). To retrieve a value from a cell, you need to refer to its value and then either collect the entire string written to the cell or refer to a specific word by its sequence number - v.text[0].tables[0].0.5.value.1.value - the value of the second word from the fifth cell of the first table from the first page of the recognized text.

To get the number of cells in the table, use the following construction - v.text[0].tables[0].0.length, that is use the "length" method after the field after which the field whose number of elements you want to know comes.

Use v.text[0].tables[0].0.5.length to get the number of words in the cell.

SCREEN AND INPUT CONTROL modules

CLICKER module

CLICKER module is designed to record a sequence of actions performed by the user, namely clicks on specific places on the screen and keystrokes on the keyboard or mouse. With this module, you can make integrations between the most different programs, because it accurately imitates the user's actions, that is, if you can do it, "Clicker" can do it, too.

Using clicker robots that look for coordinates or images is recommended on computers with the same resolution as the one on which this robot was written.

Module interface

CLICKER module consists of several areas - action selection area, action settings (not visible in this screenshot, it appears when you select an action), and the command list.

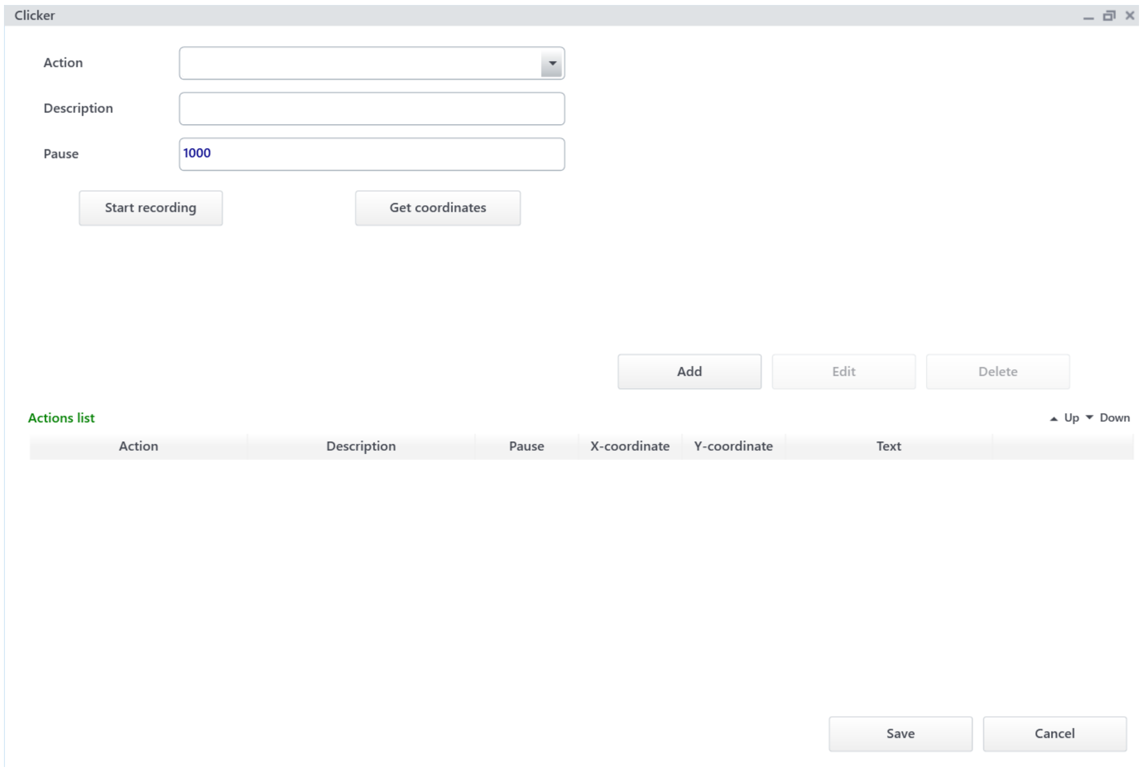


Fig. 1. Interface of the "Clicker" module

Actions section

This section is used to select the necessary action, optionally fill in its description (comment), and set the pause value - the time that must pass after the action is performed.

Action

The Action field consists of the list of available actions for writing a macro - mouse and keyboard clicks, most frequent button combinations (copy, paste). All actions available in the module are given below.

Pause

The "Pause" field is intended for creating a waiting effect after the command being executed. The value is specified in milliseconds.

For example, after clicking on the browser icon the robot should wait several seconds for the browser to start and be ready for work - in this example in the command to open the browser (by clicking its icon) you should enter a certain value in the "Pause" field, for example, 5000, i.e. 5 seconds.

Description

Description field allows you to write comments not to confuse the commands.

Start recording button

Pressing this button minimizes the Lexema Studio interface and records every click you make on the monitor, creating actions on each click and adding them to the "Command List". To stop the recording you need to open the program and click the same button (its name will be changed to "Stop Recording"). It should be noted that the actions of opening the studio will also be recorded. Most likely, you will have to delete them manually.

Get coordinates button

This button allows you to fill in the fields with coordinates for the selected action. To do this, select the desired action, press this button, point to the desired location on the screen and press the "TAB" button on the keyboard. The program will substitute the selected values in the required fields.

Action settings and Command list Sections

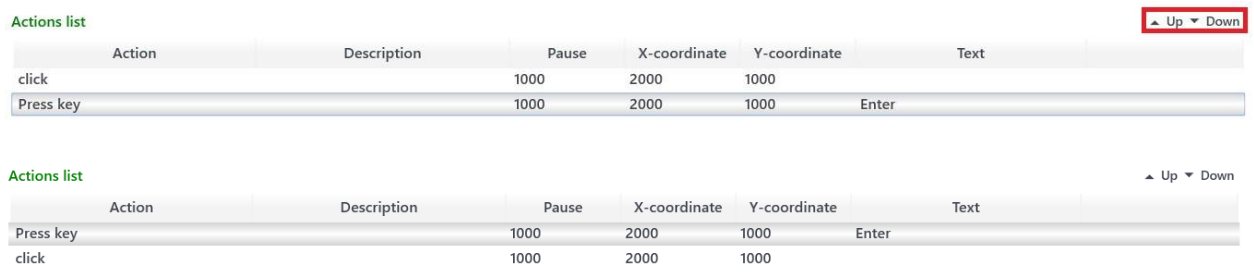
In this section you can customize the previously selected actions. The set of fields in this section is not always the same - it changes depending on the selected action. After action filling in, you should press the button below the setting - "Add", after that action will be added to the "List of commands".

To edit an existing action, click on it in the list of commands, change the desired fields and click "Edit". If you press "Add", it will be added to the end of the list of commands and the selected action will not be changed, so you can copy actions.

To delete a command, click on the command in the list of commands and then click on the "Delete" button,

Change the order of the actions

Buttons "Raise" and "Lower" are provided for changing the order of actions. The buttons are located on the top right of the list of commands. Changing the order of actions is necessary when you want to add an action, which is not provided at once and was not included in the list of commands. In this case, you can add an action in the standard way, and then move it.



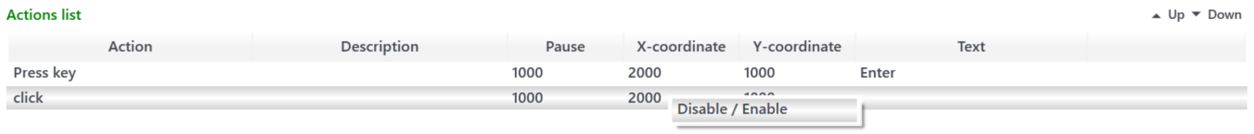
Action	Description	Pause	X-coordinate	Y-coordinate	Text
click		1000	2000	1000	
Press key		1000	2000	1000	Enter

Action	Description	Pause	X-coordinate	Y-coordinate	Text
Press key		1000	2000	1000	Enter
click		1000	2000	1000	

Figure 2. Changing the order of actions

Deleting a command

If you don't want to delete the command because you want to check how the robot works without it and then reset it, you can disable the command by right clicking on its line and disable or enable it.



Action	Description	Pause	X-coordinate	Y-coordinate	Text
Press key		1000	2000	1000	Enter
click		1000	2000	1000	

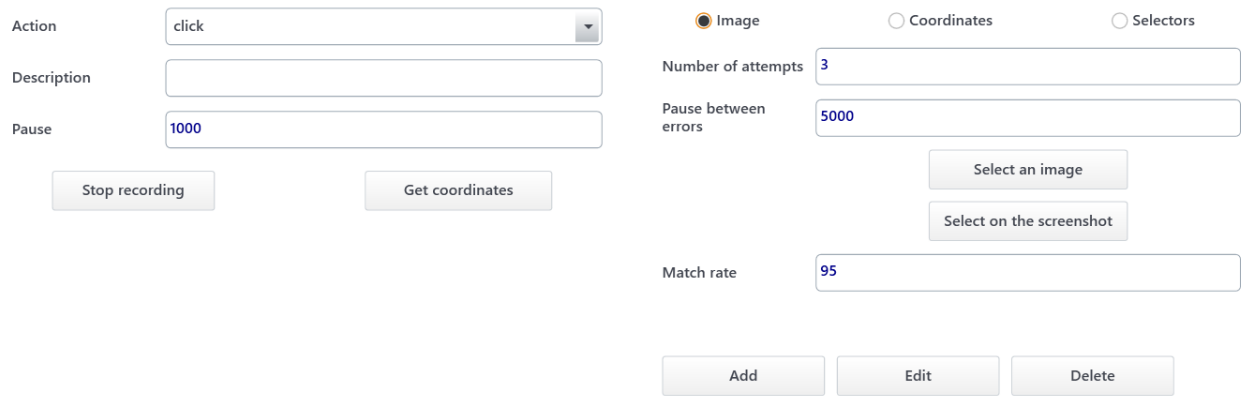
Fig. 3. Command context menu

Fields for working with coordinates

Image", "Coordinates" and "Selectors" switches

When creating click or pointing actions, you are prompted to choose whether you want to click on certain coordinates or on the image. If the program you are working with is static and does not change the position of its buttons, you can use click by coordinates. To fill in these fields, use the "Get coordinates" button located in the action selection section.

If the program can change the location of its elements, you can use click on the image. To do this, click on the "Image" switch.



Action	click
Description	
Pause	1000

[● Image] ○ Coordinates ○ Selectors

Number of attempts	3
Pause between errors	5000
Match rate	95

Fig. 4. Command creation

When you change the switch the lower part of the window will change - the buttons "Select image", "Select image" and the input field "Match percentage" will appear.

The image to be clicked to perform the action can be set up in two ways:

1) load an existing image by clicking the "Select image" button

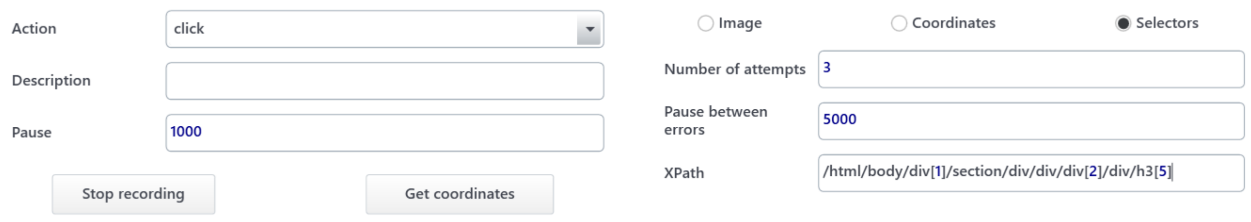
2) select the fragment of the window in which the action will be reproduced, using the "Select image" button. After clicking on the button, the studio will make a screenshot and offer you to select a fragment of the resulting image.

When using the click on the image, note that if there are several such images on the screen, the action will be applied to the first one found!

Field "Percentage of matching" is necessary for cases when the searched image can differ a little from what it can be in reality (for example, presence of a selection around the button, if it was chosen earlier, and absence of such selection). In such cases, reduce the match percentage until the image is in both cases. However, if the match percentage is reduced too much, the module may find something that is not planned

The "Number of attempts" and "Pause between errors" fields are for cases when the program was unable to find a given image. You can specify how many times to try to search for the desired image and how many seconds to wait between attempts. For example, after launching the browser, you should click on the button with the "Plus" icon - creating a new tab. If the browser will load too long, more than 5 seconds (this time is specified in the pause command of the browser opening), then the program will start searching for the "plus", which does not yet exist. Then the program will wait the number of seconds specified to it between attempts and try again. Once the image is found, the program will wait the number of seconds specified in the "Pause" field of this command and move on to the next one.

The last option for finding a place to point the mouse is "Selectors". This option is considered the most reliable, but is supported by a narrow range of programs. Windows selectors allow you to find the path to an item on the screen using not just coordinates or an image, but the path relative to the parent component of the screen using special markup. They work with the computer system and some desktop applications. Usage is similar to coordinates - just press "Get Coordinates", point to the item you are looking for and press "TAB".



Action	click
Description	
Pause	1000

Image [✓ Selectors] Coordinates

Number of attempts	3
Pause between errors	5000
XPath	/html/body/div[1]/section/div/div/div[2]/div/h3[5]

Fig. 6. Creation of command with selectors in Windows

Actions available in the module

Click action

The main action of the module is to click with the left mouse button on the specified coordinates or image.

Double-click action

Double-click on a specified coordinate or image, for example, to open an application from the desktop.

Right-Click action

Right-click, for example, on an application on the desktop to open its context menu.

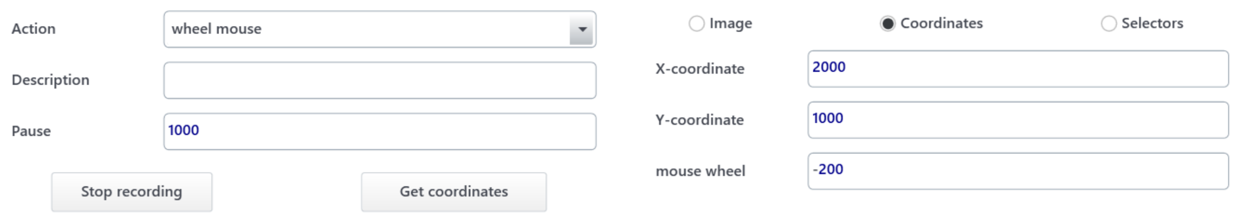
Mouse wheel click action

In some programs it is possible to use a mouse wheel click to perform certain actions, such as closing a tab in a browser.

Mouse Scroll action

Scroll with the mouse wheel down or up the page. Here the coordinates play the role of where the cursor will be when scrolling the wheel, since there can be several sliders on the screen. When you select this action, the "Mouse Scroll" field appears. In this field, enter the number of "wheel scrolls" you want the mouse to scroll by. To scroll the screen downwards, a negative value (-150) must be entered in the "Mouse Scroll" field, and a positive value (150) must be entered upwards. The number of scrolls is measured in the number of times you scroll with the mouse wheel.

To scroll down a particular slider, point the cursor at it, press the "TAB" key on the keyboard, select the "Mouse Scroll" action, and fill in the "Mouse Scroll" field, as shown in Figure 8:



Action	wheel mouse
Description	
Pause	1000

Image [✓ Coordinates] Selectors

X-coordinate	2000
Y-coordinate	1000
mouse wheel	-200

Figure 7. Example of command setup

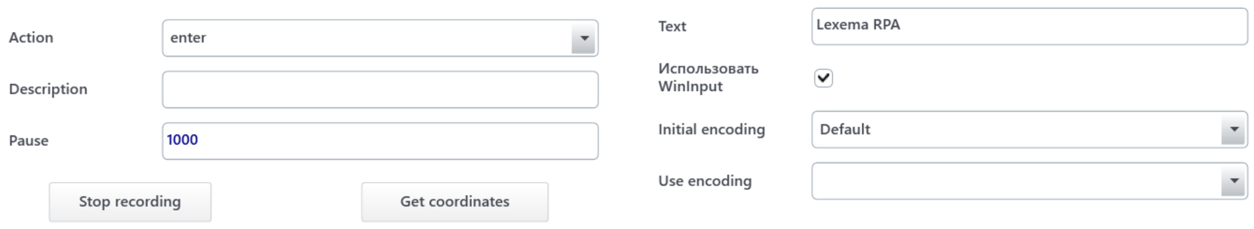
Enter text action

This action enters text, which is written in the "Text" field. The field, where the text is entered, should be selected (with focus and carriage).

Some programs require specific text input - in a particular encoding. In such cases it is necessary to choose encoding from offered in the drop-down list "Use encoding", or specify your own. Text can lie in a variable and be copied from some program which uses a specific encoding. In this case, when pasting the copied text into another program, you must first read it in the encoding of the original program. The encoding of the source program is entered or selected in the Initial Encoding field. Filling of these fields is not obligatory.

The "Use WinInput" checkbox allows to enable a more correct and quick text input mode, but it may be not supported by some programs. Its use is optional.

Example of filling in the action:

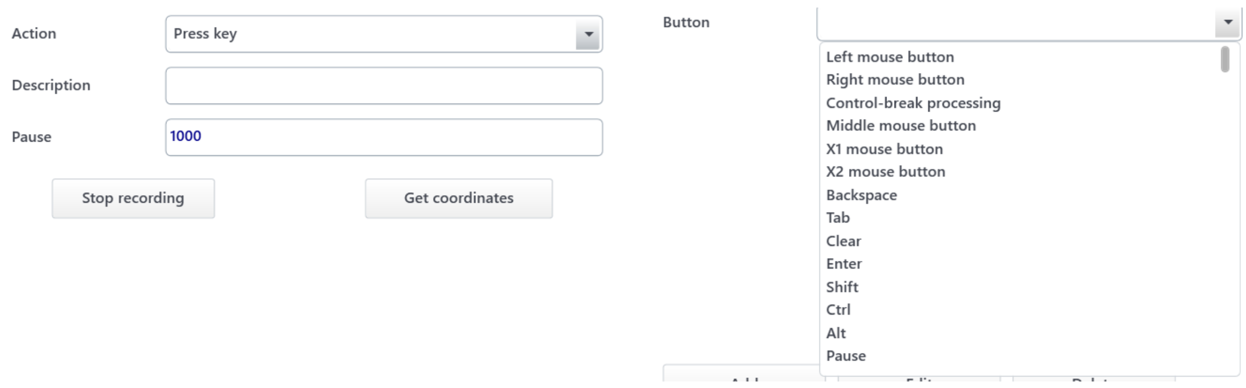


Action	enter
Description	
Pause	1000
Text	Lexema RPA
Использовать WinInput	✓
Initial encoding	Default
Use encoding	

Figure 8. Example of filling

Press Key Action

The action imitates the pressing of the keyboard key. The necessary key should be chosen from the drop-down list "Button". All possible keys, which can be present both on simple and extended keyboards (volume control, music start/stop and other non-standard keys) are suggested in the list.



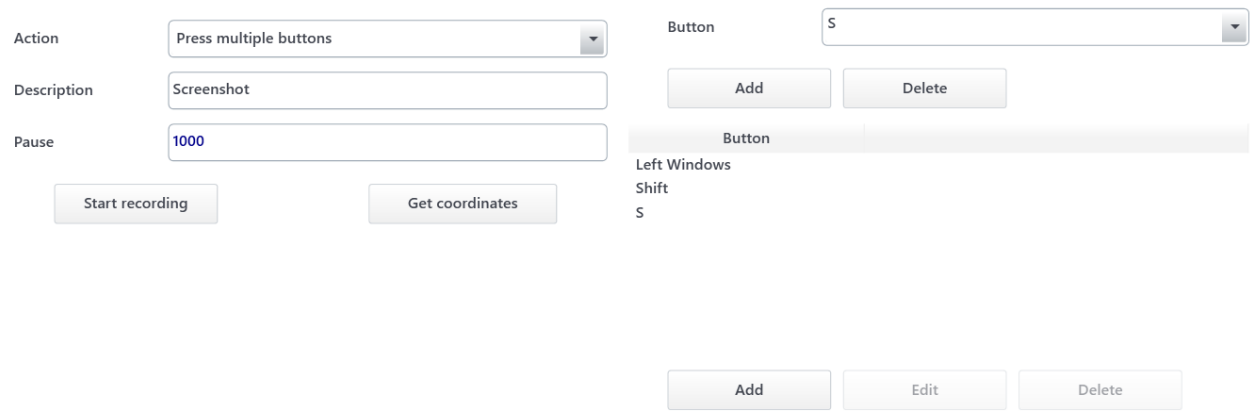
Action	Press key
Description	
Pause	1000
Button	

Left mouse button
Right mouse button
Control-break processing
Middle mouse button
X1 mouse button
X2 mouse button
Backspace
Tab
Clear
Enter
Shift
Ctrl
Alt
Pause

Fig. 9. Action "Press key".

Action Press several keys

It simulates keyboard shortcuts, e.g. CTRL+A - selection of all text in the field. The necessary buttons are selected from the Button drop-down list and added to the table below it. The list of keys is the same as in the action "Press Button".



Action	Press multiple buttons
Description	Screenshot
Pause	1000
Button	S

Button
Left Windows
Shift
S

Fig. 10. Action "Press several keys".

"Add" Button under the list of selected buttons adds action to the common list of actions.

Copy Action

Imitates pressing the CTRL+C shortcut key. It copies the selected value to the clipboard and, if necessary, writes it to the variable specified in the "Copy Into" field. This field is optional.

Paste Action

Imitates keyboard shortcut CTRL+V. Pastes the value from the clipboard into the selected field.

Till Image Disappears Action

Expects the image to be missing for the time allowed to find the error (value of field "Number of Attempts" multiplied by the value of field "Delay between Errors").

Get text from field action

Gets text from the field on the screen that is searched by XPath and writes it into the specified variable.

IMAGE SEARCH module

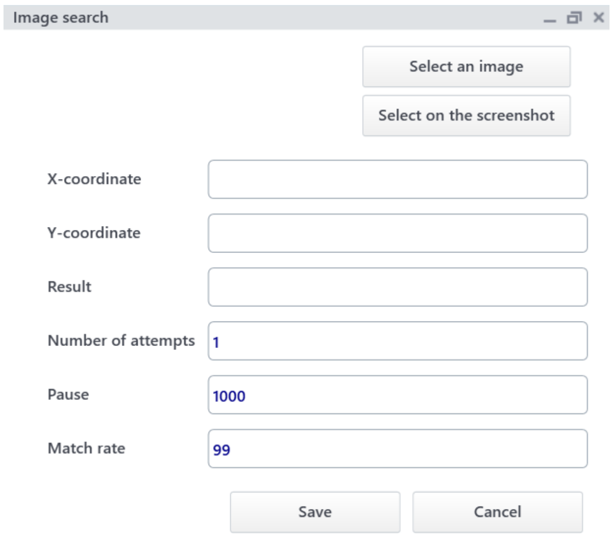
Gets text from the field on the screen that is searched by XPath and writes it into the specified variable.

The "Image Search" module

The "Image Search" module is mainly used in conjunction with the "Clicker" module. Its task is to determine whether a given image exists on the current screen, and if so, to return the coordinates of its center.

Module interface

The module window looks as follows:



X-coordinate	
Y-coordinate	
Result	
Number of attempts	1
Pause	1000
Match rate	99

Fig. 10. Module window

The "Select image" and "Select image" buttons allow you to select an image for search.

Field "x-coordinate". A variable (beginning with "v.") is inputted into this field, in which the x-coordinate of the searched image will be written, if it is found.

The "y-coordinate" field. Similarly to the "x-coordinate" field, in this field you must specify the variable in which the y-coordinate will be written.

Result field. In this field enters the name of the variable, which will be the final result of the search for the image (after using all attempts to find it. The resulting value will be represented as a string: "true" - found, "false" - not found.

Fields "Number of attempts", "Pause" and "Match percentage" are similar to fields of module "Clicker".

Field "Tries". This field specifies the number of attempts to find the image.

The field "Pause" is intended for indicating the pause between search attempts in milliseconds.

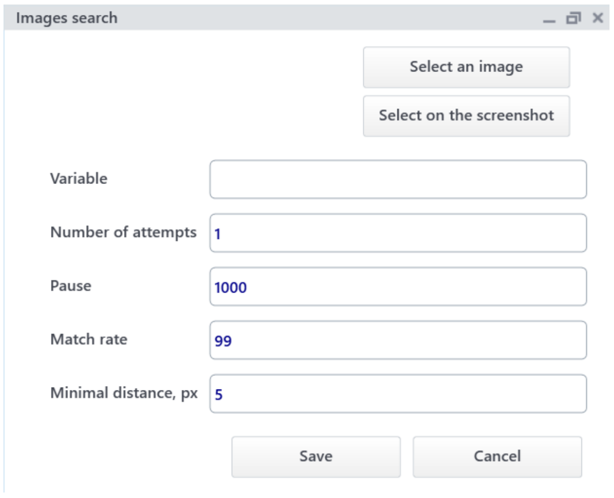
Field "Percentage of match" - how much the image to be searched coincides with the fragment of the image on the current screen. A number from 0 to 100 is entered; the higher the number, the greater the similarity between the images should be.

IMAGES SEARCH module

IMAGES SEARCH module is similar to the IMAGE SEARCH module, the only difference being that it returns a list of all images found, not just one.

Module interface

The interface of this module is very similar to the interface of the module "Image Search", but it has only one field instead of the fields "X coordinate", "Y coordinate" and "Result" - "Variable", as well as the field "Minimum distance" appeared.



Variable	
Number of attempts	1
Pause	1000
Match rate	99
Minimal distance, px	5

Fig. 11. Module Window

In the "Variable" field, the name of the variable with "v." symbols, in which the search result will be written.

"Number of retries" field is filled with a number, indicating how many times to try to find images, if none could be found from the first time.

"Pause" field is intended for indicating the pause between search attempts in milliseconds.

In the field "Percent match" you enter a value from 0 to 100, indicating how much the image to be searched for matches the image fragment on the current screen. The higher the number, the more they must match.

The "Minimum distance" field is filled with a number indicating how many pixels there should be between two identical images. In some tasks, you may need to cut off images that are close enough to each other and get only the first of those images (starting from the upper left corner). Note, if this distance is very small (at least 5 pixels) or equal to 0, one image can be found several times!

Output data

The specified variable will contain a list of objects equal to the number of found images on the screen. Each object corresponds to a found image. To refer to a particular image of the list, you need to specify its number (numbering from 0) in square brackets after the variable name, for example v.listImages[0].

Each object contains three fields: "x", "y" and "similarity". In the field "x" contains a coordinate of x of the chosen image, in the field "y" - coordinate of y, and in the field "similarity" - percent of coincidence of the found image with the searched.

CUT IMAGE SECTION module

CUT IMAGE SECTION module allows you to resize the image by setting the coordinates of the upper left and lower right corners of the fragment of interest.

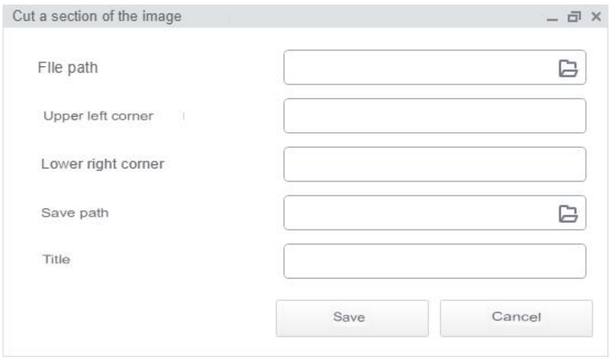


Fig. 25. "Cut section" module interface

Description of the fields of the module:

- File path - path to the main image;
- The left top corner - coordinates of the left top corner of the required fragment of the image, format - x;y;
- Right bottom corner - coordinates of the right bottom corner of the required fragment of the image, format - x;y;
- Saving path - the path to the folder where the image fragment is to be saved. It may be specified together with the final file name (must have an extension);
- Name - the name of the resulting fragment of the image, if it is not specified in the path above. If the extension is not specified, it will be saved in bmp format.

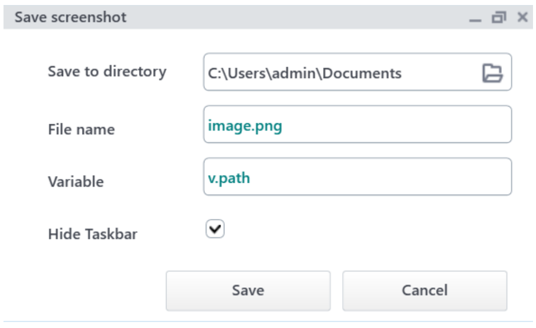
SAVE SCREENSHOT module

SAVE SCREENSHOT module allows you to take a screenshot of the screen while the robot is running, which is useful for debugging clicker robots (robots that mainly run on the Clicker module).

The module window consists of two fields:

The "Save to folder" field should contain the path to the folder where the screenshot will be saved. The Folder button in the field calls a directory selection dialog box;

The "File name" field must contain the name of the screenshot. If the extension is not specified, the image will be saved in png format.



Save to directory: C:\Users\admin\Documents
File name: image.png
Variable: v.path
Hide Taskbar: ✓

Fig. 14. The "Create screenshot" module window

CONTROL FLOW modules

The modules described in this section do not do anything on their own, they must be used together with any other modules. The modules described (except for the "Break" module), apart from the usual "Notes", have several other parameters that must be filled in, but there is no separate window with settings.

CONDITIONAL module

"Conditional" module allows you to perform certain actions depending on the specified condition. A condition can be any expression, which is set to "True" or "False".

Module interface

The module consists of the "Note" and "Condition" fields. Actions that should be executed if the condition is "True" are added to the "THEN" section, if the condition is "False" - to the "ELSE" section. Actions are added to the module by dragging them from the menu or another place of the work area.

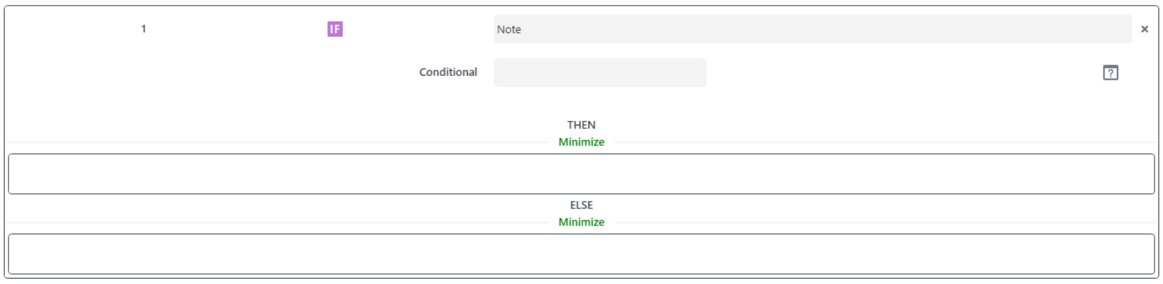


Fig. 1. Conditional module

The "Note" field is used to enter a comment from the robot's developer.

The "Condition" field is filled out according to JS programming language rules. For those users, who don't know that language, we offer help in the form of "Condition Builder", accessible by clicking on the button to the right of the condition.

Conditional constructor

You can use Conditional constructor to build conditions according to JS syntax.

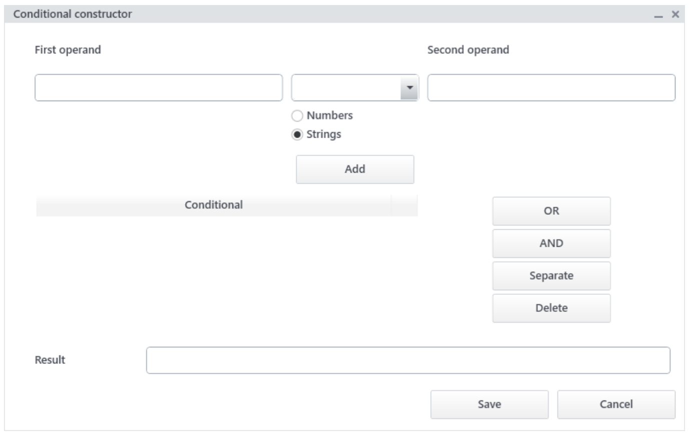
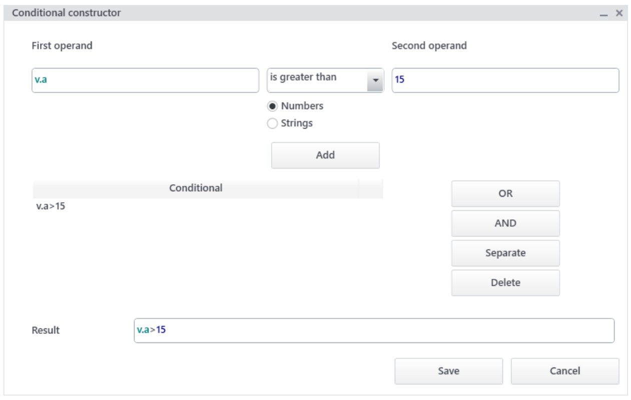


Fig. 2. Conditional constructor

Creating

To make a condition you need to fill in both operands, select the type of operands (string or number), select the operator between them from the offered ones and click the "Add" button:



First operand: v.a
is greater than
Second operand: 15
● Numbers
○ Strings

Conditional
v.a>15

Result: v.a>15

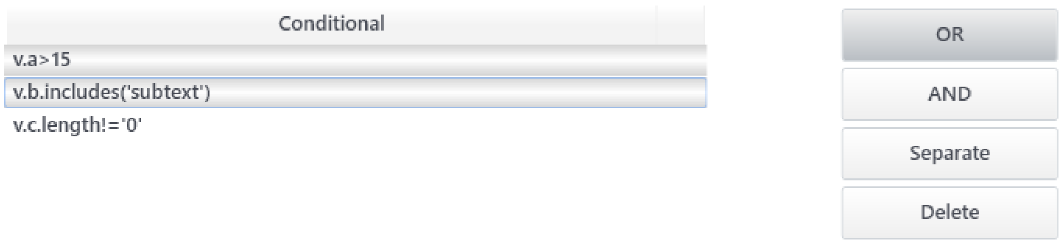
Figure 3. Adding conditions

The list of operations for strings and numbers differs, so select the operand type first, and then the operation.

Binding

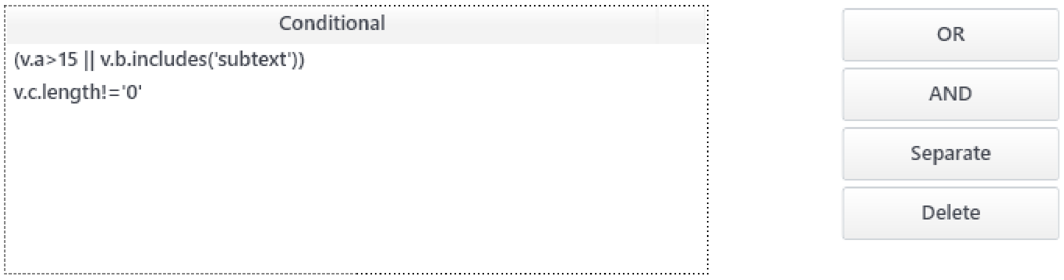
You can bind several conditions with Boolean "AND" or Boolean "OR".

Hold Shift and select several conditions to be joined by the same operator, then click the button with the corresponding logical operator:



Conditional
v.a>15
v.b.includes('subtext')
v.c.length!='0'

Fig. 4. Binding conditions



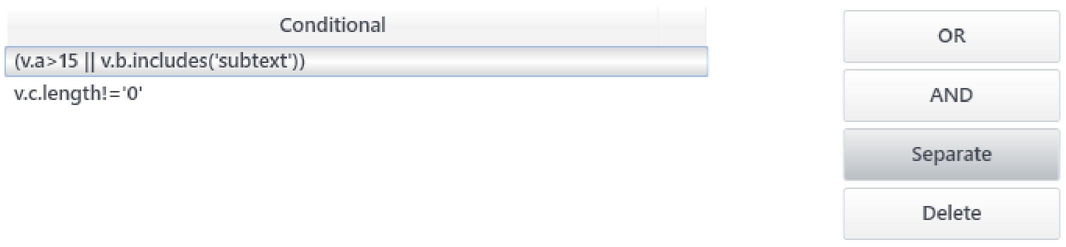
Conditional
(v.a>15 || v.b.includes('subtext'))
v.c.length!='0'

fig. 5. Result of Binding Conditions

||" stands for logical OR, "&&" for logical AND.

Separating

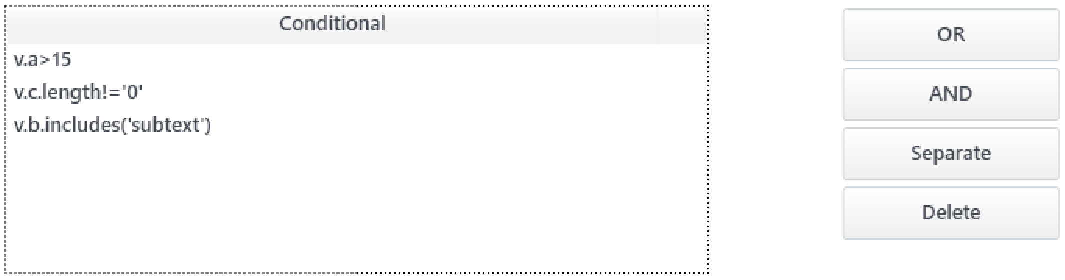
Conditions consisting of several simple conditions, i.e. those using logical AND or OR, can be separated by clicking on the SEPARATE button. The disconnection is carried out by the logical operators.


Conditional

(v.a>15 || v.b.includes('subtext'))

v.c.length!='0'

Figure 6. Conditional disconnection.


Conditional

v.a>15

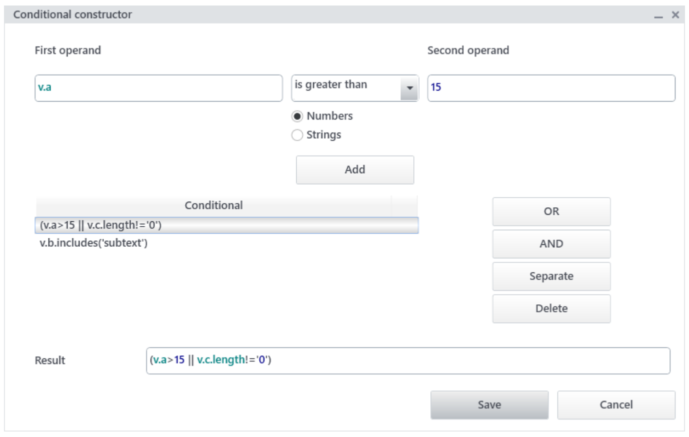
v.c.length!='0'

v.b.includes('subtext')

Fig. 7. Results of separating conditions

Saving

You can save the condition by clicking the "Save" button. The condition will be inserted to the "Condition" field of the conditional module:


Conditional constructor

First operand: v.a

is greater than

Second operand: 15

● Numbers
○ Strings

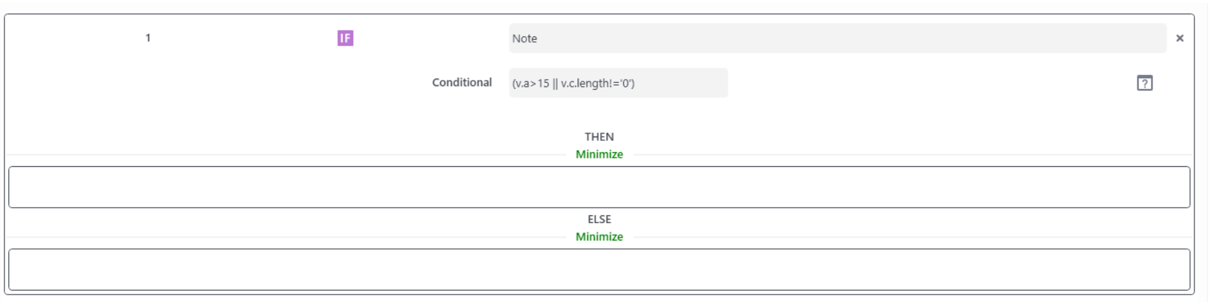
Conditional

(v.a>15 || v.c.length!='0')

v.b.includes('subtext')

Result: (v.a>15 || v.c.length!='0')

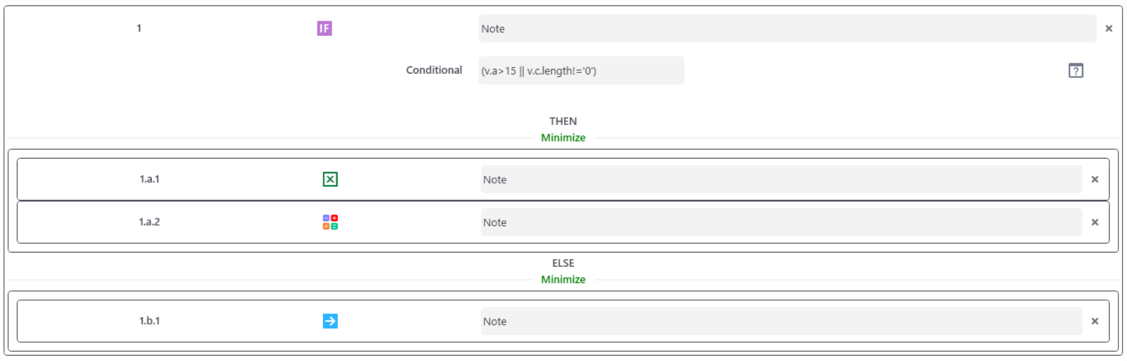
Fig. 8. Saving condition


1 IF Note

Conditional: (v.a>15 || v.c.length!='0')

fig. 9. Displaying a Condition

The completed module can look as follows:

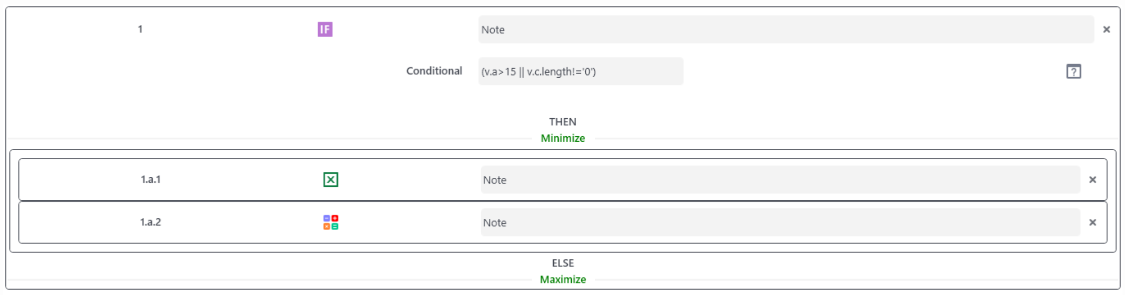


Conditional (v.a>15 || v.c.length!='0')

Fig. 10. Completed "Condition" module

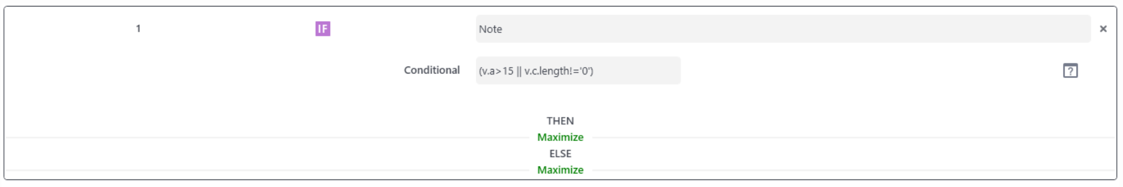
Minimizing

To reduce the space occupied in the working area of the project, you can collapse each block of this module using the "Collapse" button.



Conditional (v.a>15 || v.c.length!='0')

Figure 11. Collapsed ELSE block.



Conditional (v.a>15 || v.c.length!='0')

Fig. 12. Collapsed THEN and ELSE.

FOR LOOP module

"FOR loop" is used to repeatedly (cyclically) perform a set of actions. Actions added to the module will be executed until the number of iterations specified in the settings passes.

Module interface

The module consists of the "Note", "Number of iterations" and "Variable" fields. Actions to be performed in the loop are added to the block at the bottom of the module. Actions are added to the module by dragging them from the menu or elsewhere in the workspace.

The module looks as follows:

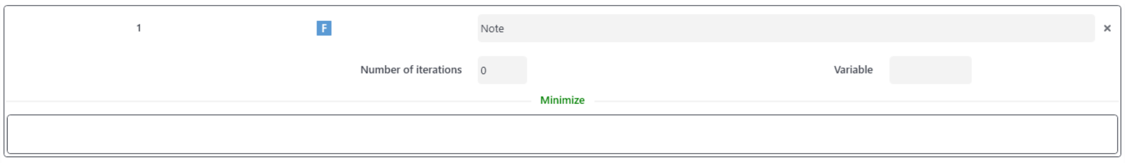


Figure 13. View of the module

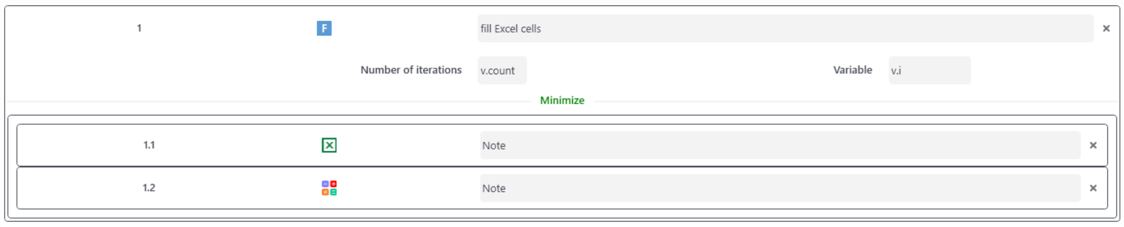
The "Note" field, as in all other modules, is used to enter comments from the robot's developer.

The other fields are unique to this module.

The "Number of iterations" field is intended to enter the number that means how many times to execute the actions contained in the module.

The "Variable" field is a variable, which will store the number - the current step of the cycle. The cycle starts with zero.

An example of a completed loop is shown in Figure 14.

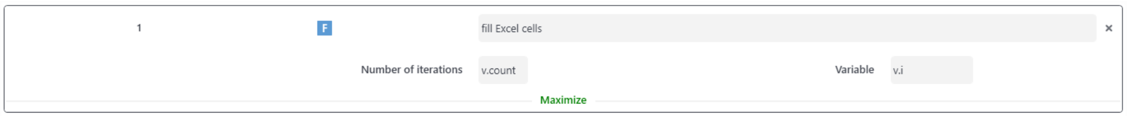

1 fill Excel cells

Number of iterations: v.count

Variable: v.i

Fig. 14. Example of filling in the module

Using the "Minimize" button, you can collapse the loop actions to display the command in a more compact way.


1 fill Excel cells

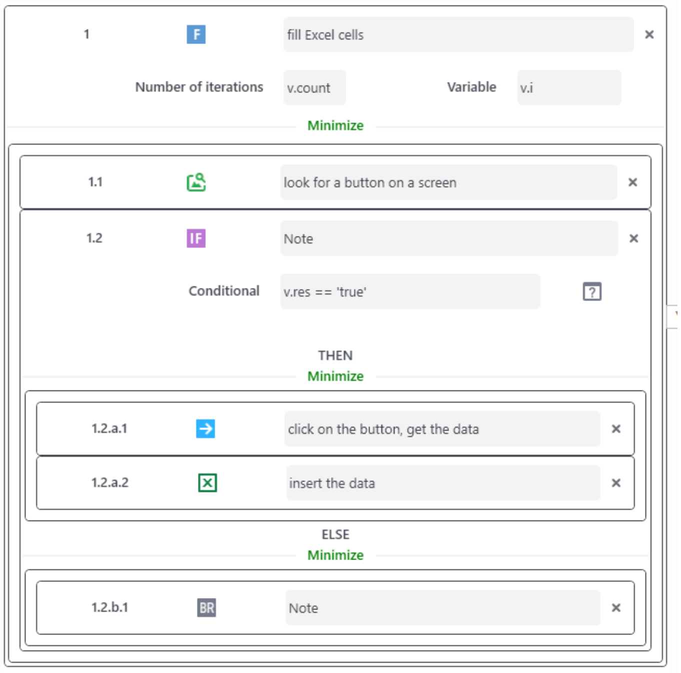
Number of iterations: v.count

Variable: v.i

Figure 15. Module Minimized view

BREAK module

For "emergency" exit from the simple cycle the module "Break" is used. For example, the cycle searches for a particular image, and if that image is found, you should click on it, read some data and go to the next action, otherwise - exit the cycle and not play the next action. For such a scenario, you need to create a loop with the "Condition" module, one of the branches of which will contain the "Break" module (the "Break" module is represented as the last module in the example):



1 F fill Excel cells
Number of iterations v.count Variable v.i
1.1 look for a button on a screen
1.2 IF Note
Conditional v.res == 'true'
THEN
1.2.a.1 click on the button, get the data
1.2.a.2 insert the data
ELSE
1.2.b.1 BR Note

Fig. 16. Example of using the "Break" module.

Module "Break" does not require configuration, but can be used only within the FOR loop and WHILE loop modules.

RETURN Module

There are situations when it is necessary to terminate the work of the robot under certain conditions, for example, you need to process a letter which should have arrived at the mailbox, but it did not. For such purposes, there is the Return module.

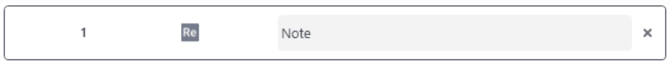

1 Re Note ×

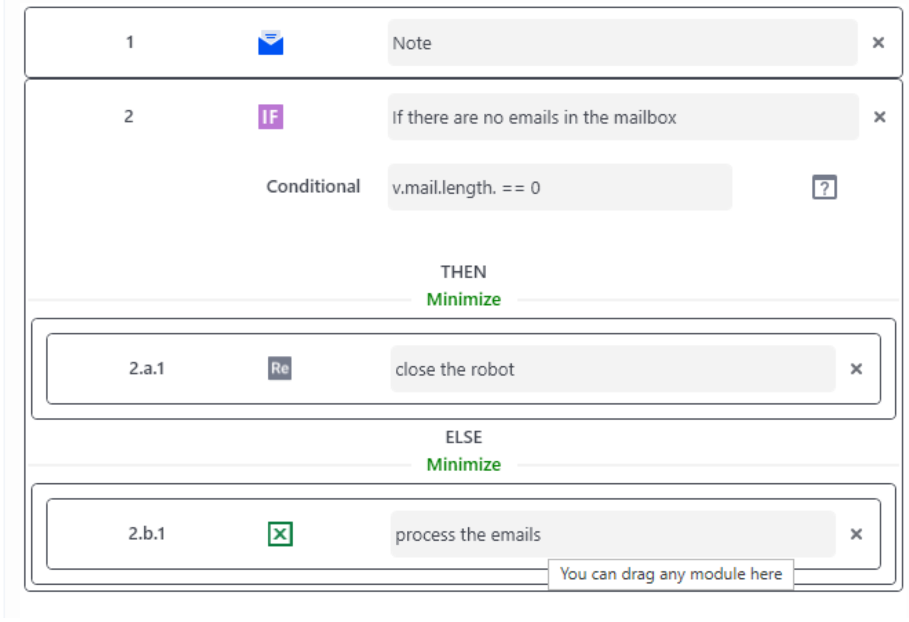
Fig. 17. Module appearance

This module terminates the entire robot and does not require configuration. It can be used anywhere.

When coding robot functions for their subsequent use in the module with the same name, the Return module is used to return the resulting variable into the main robot. To do that, open the module Return by double-clicking and entering the name of the variable returned from the function.

EXAMPLE 1

The figure below shows an example of using the module: let all the e-mails for a certain period of time be read using the "Read e-mails" module. All subjects of letters and their contents should be entered in the Excel file. But what if there are no emails? To do this, we check for the presence of e-mails and if there are no e-mails, the robot will stop working, otherwise it will continue.


1 Note ×

2 IF If there are no emails in the mailbox ×

Conditional v.mail.length. == 0

THEN
Minimize

2.a.1 Re close the robot ×

ELSE
Minimize

2.b.1 process the emails ×

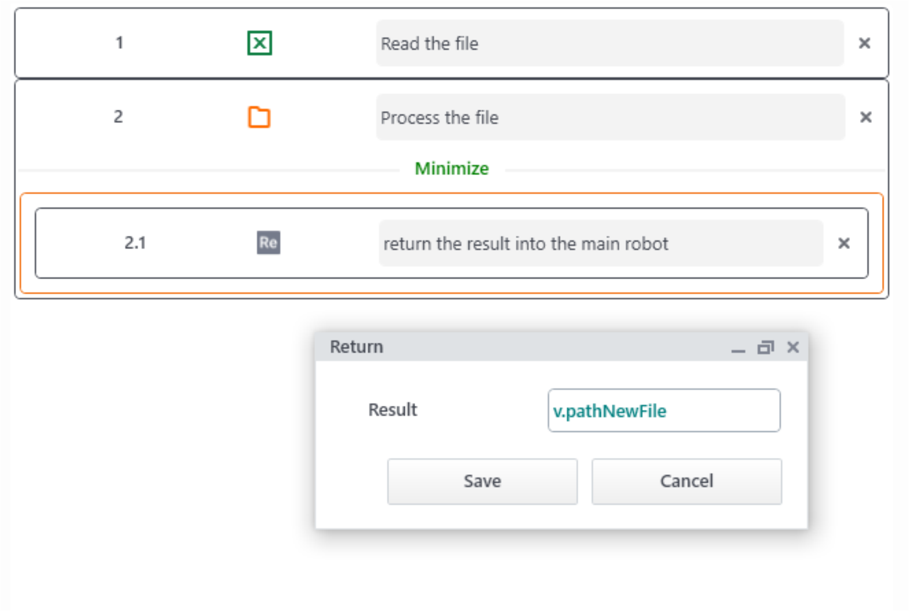
You can drag any module here

Fig. 18. Example of Using the "Return" Module

Using this module, you can write the robot in a common workspace rather than in the branches of the Condition module. In this way, the robot looks neater and clearer.

EXAMPLE 2

When used in a robot function: Suppose there is a robot function that processes different but identical Excel files. The input is the path to the file, and the output is the path to the new, processed (or consolidated) file.


1 Read the file

2 Process the file

Minimize

2.1 return the result into the main robot

Return

Result: v.pathNewFile

Fig. 19. Example return value in module "Return".

CONTINUE module

The "Continue" module allows you to continue to the next loop iteration without using the "Condition" module or other methods of branching the algorithm. The module does not require additional configuration. The only requirement - the module must be inside the module "Cycle" or inside any module, which, in its turn, is inside the cycle (the nesting of modules is not limited).

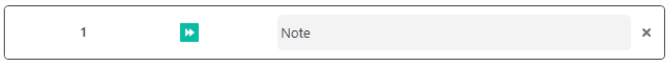
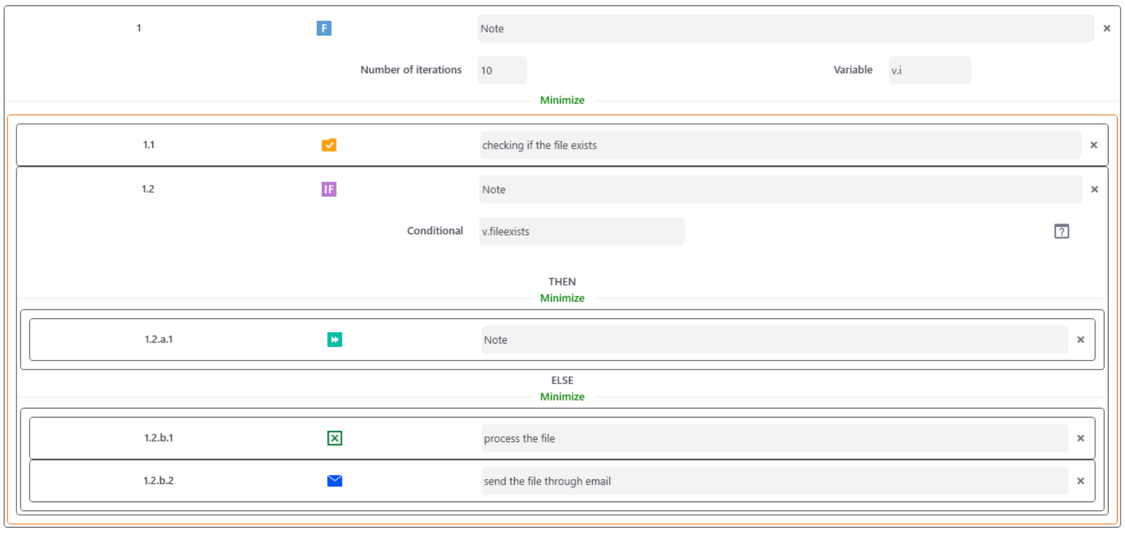


Fig. 20. The "Continue" module.

EXAMPLE

Suppose that you want to perform actions with the file, but only if it is present in a particular folder. To do this, simply check at the beginning of the cycle to see if the file is there, and if it is not, then proceed to the next iteration of the cycle.



1 F Note

Number of iterations: 10

Variable: v.i

Minimize

1.1 checking if the file exists

1.2 IF Note

Conditional: v.fileexists

THEN

Minimize

1.2.a.1 Note

ELSE

Minimize

1.2.b.1 process the file

1.2.b.2 send the file through email

Fig. 21. example of working with "Continue

The only difference from working with the condition is better readability of the algorithm (i.e., without the Continue module, all actions over the file would have to be done within the ELSE branch of the Condition module).

WHILE LOOP module

The "While loop" module performs the actions added to it until the specified condition returns "True".

Module interface

The "Note" and "Condition" fields are located in the module's window. The block at the bottom of the module contains the actions to be executed in the loop.

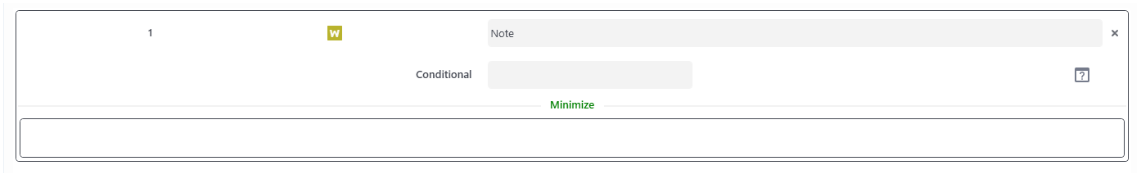


Fig. 22. module view

Field "Conditional". You should write it according to the JS syntax. This button opens the Condition Builder (described in the Conditional Module chapter).

Adding actions to the module is done by drag-and-drop from the modules menu or from the working area of the project.

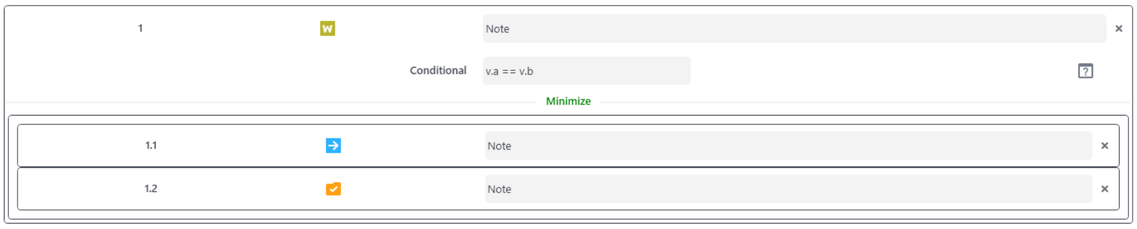


Fig. 23: Adding actions.

When you click on the "Collapse" button, the actions described in the loop are minimized to a more compact display of the command.

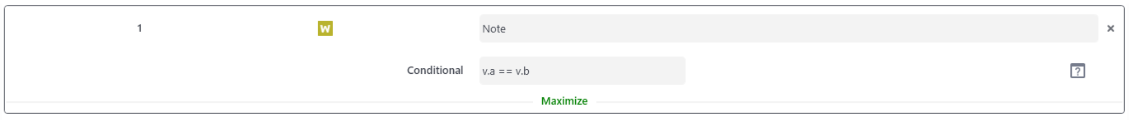


Figure 24. The minimized view of the module.

TRY/CATCH module

This module allows you to handle errors that may occur during the execution of a particular module.

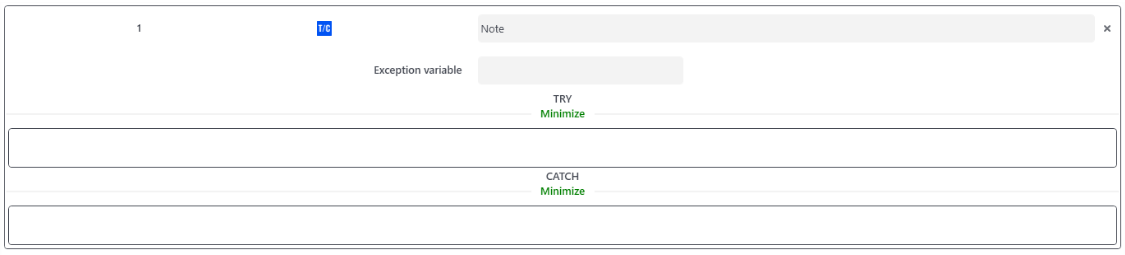


Fig. 25. Exterior view of the Try/Catch module

The module consists of two blocks - TRY and CATCH. The TRY block contains a set of modules or the whole robot to be executed, the CATCH block contains a set of modules in case an error occurs in the execution of modules of the TRY block.

The module has two text fields, the first is the "Note" field, where you can enter a module description for easier navigation of the robot, and the second field is for saving the error that occurred in the TRY module. Since it is possible to know which error occurred, every possible error can be handled correctly in the CATCH block.

EXAMPLE

Let's assume that the task is to download an email from the mail with a certain attachment name to a predefined folder and process that attachment with the Excel module, and the path to the file will always be the same in the Excel module. For example, the attachment is called "Report for the day" and the path to the folder is "C:\Reports", then Excel will always open the file "C:\Reports\Report for the day.xlsx" and attachments from mail will always be downloaded to the aforementioned folder.

If the letter with the required attachment has not come yet, the Excel module will give an error "C:\Reports\Day Report.xlsx document could not be opened". In that case, you need to handle that error somehow, suggest that the user run the robot later, or tell him to wait for a while and try again.

To handle the error with the output of the information window, you need to add the "Try/Catch" module, where the TRY block will contain modules for reading the letter and Excel, and the CATCH block will contain the "Interface" module for outputting the user's window.

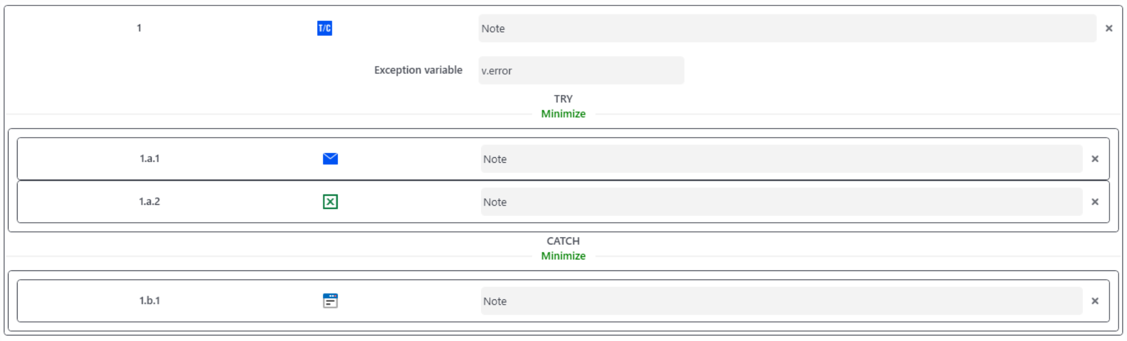


Fig. 26. Example of using Try/Catch module

The result of such robot's work is either a standard message "The script worked successfully!" if the mail was in the mailbox, or a window with the text "The mail has not come yet" in other case. The v.error variable will contain the text "Failed to open document C:\Reports\Day.xlsx".

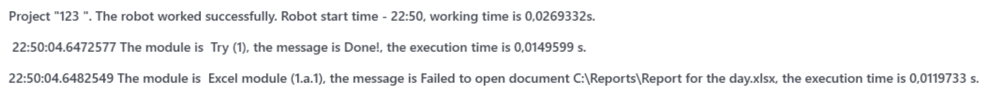

```
Project "123 ". The robot worked successfully. Robot start time - 22:50, working time is 0,0269332s.
 22:50:04.6472577 The module is  Try (1), the message is Done!, the execution time is 0,0149599 s.
22:50:04.6482549 The module is  Excel module (1.a.1), the message is Failed to open document C:\Reports\Report for the day.xlsx, the execution time is 0,0119733 s.
```

Fig. 27. Example of error handling

SWITCH module

The Switch statement is a selection statement that chooses for execution one case section from the list of candidates, comparing them to the expression specified in the switch itself. It is similar to several Condition modules, as if one value had to be compared to several others and each had a separate set of actions to perform, that is, it is used when there are more than two condition forks, but the comparison is with the same value.

Only Case blocks can be inside a Switch block and vice versa - Case blocks can only be inside a Switch block.

Module Interface

The module Switch has a field "Condition", in which a variable is written, with which the sub-modules of case will be compared in the future to select the desired algorithm. By the "Add" button these very submodules are added.

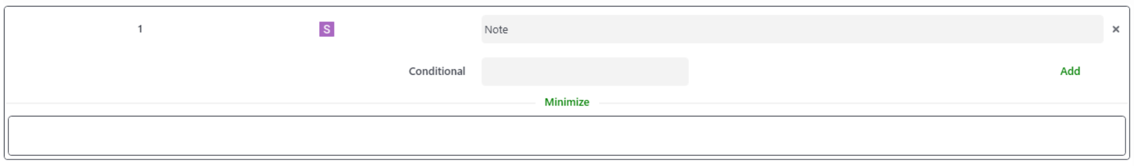


Fig. 28. The module "Switch"

After clicking the "Add" button, the block Case appears inside the module, which can only be used inside the block Switch. The Case blocks also contain a "Condition" field, but in this case, they contain the expected value of the variable specified in the Switch module. If the variable from Switch equals the value of Case, that Case will be executed, other blocks of Case will not be executed (even if they contain the same value).

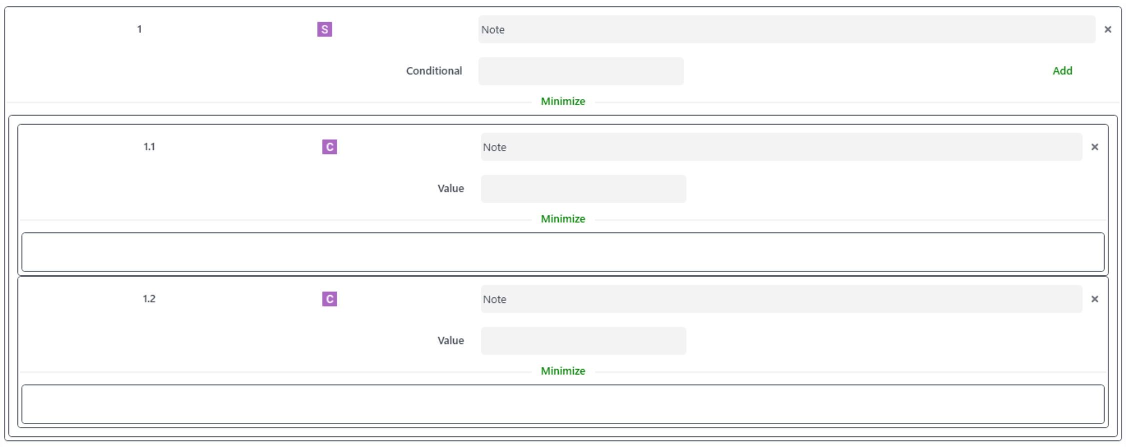


Figure 29. Switch - Case modules

Any program modules, like the "Group" module, are placed inside the "Case" blocks.

WEB modules

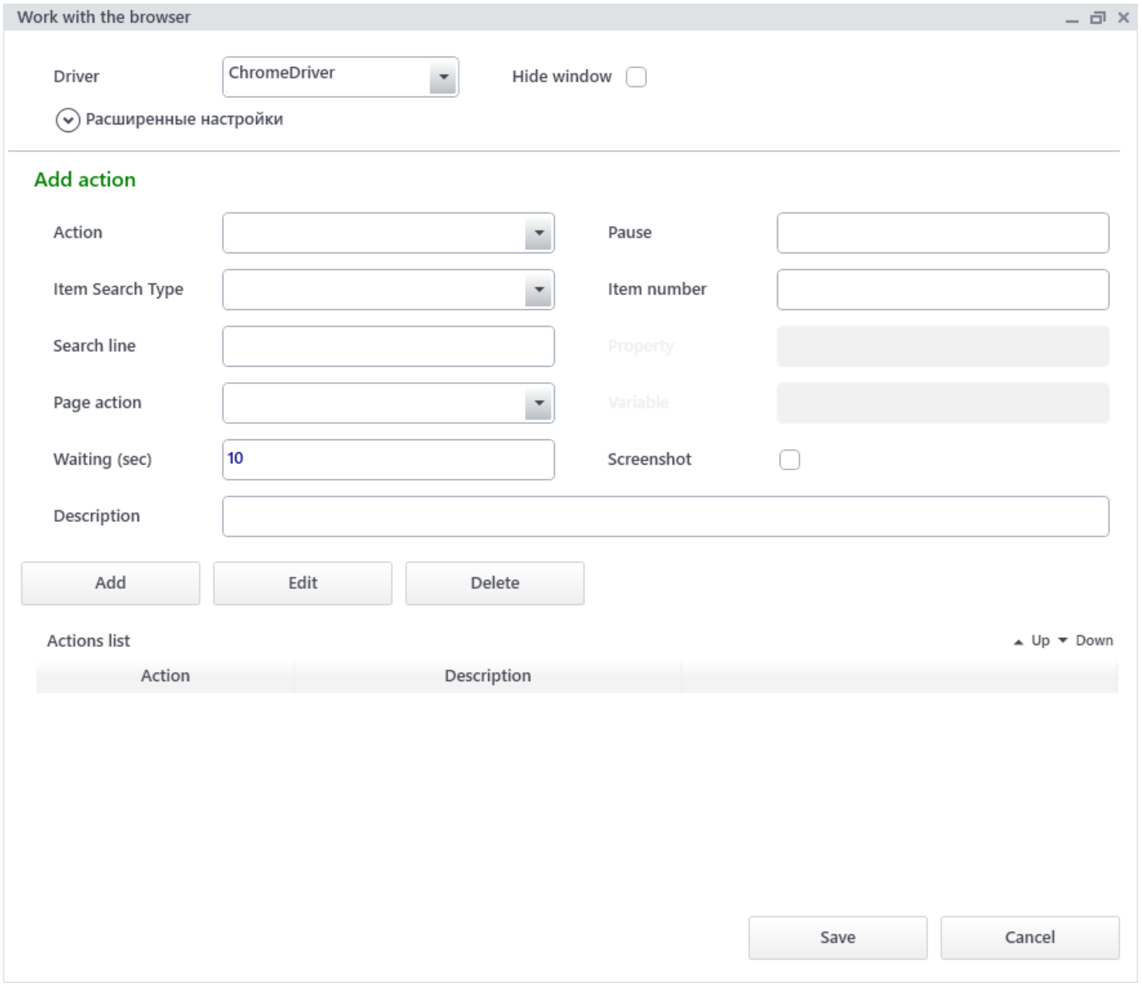
BROWSER module

Module highlights

The "Browser manipulation" module currently allows you to work with two web browsers - Google Chrome and Mozilla Firefox. It is based on Selenium WebDriver and allows you to implement the basic methods of working with the browser page - open a tab, switch to a tab, find an item on the page by its key and process it somehow.

Module interface

The module's window is divided into a block of browser settings, command creation area to work with the browser, a group of buttons to control actions and a table with a list of created commands. The command creation area consists of several fields for filling, and the number of fields available for editing changes depending on the selected action in the Action field. Group of buttons consists of three buttons - "Add", "Edit" and "Delete".



Driver: ChromeDriver

Waiting (sec): 10

Fig. 1. The "Browser manipulation" module window.

The creation of the command begins with the choice of the action. After you have chosen it, the list of available fields in the window may change. You should fill out the fields necessary for the chosen action and press the "Add" button. Created action will be displayed in the "List of commands" table. To edit a command you should select it in the list of commands, change the necessary fields and press the "Edit" button. If you press the "Add" button, a new action will be added that is similar to the one you've selected in the list of commands but with the changes you've made. It is possible to change nothing and to press the button "Add" and then the full copy of command will be obtained.

To delete unnecessary commands you need to select the command in the table and click "Delete".

The robot will execute all the commands in sequence, so you can use the buttons in the upper right corner of the table to change the order of commands in the table. You can do it in the following way: choose a line, which position you want to change, and then press Up or Down depending on where you want the line to be.

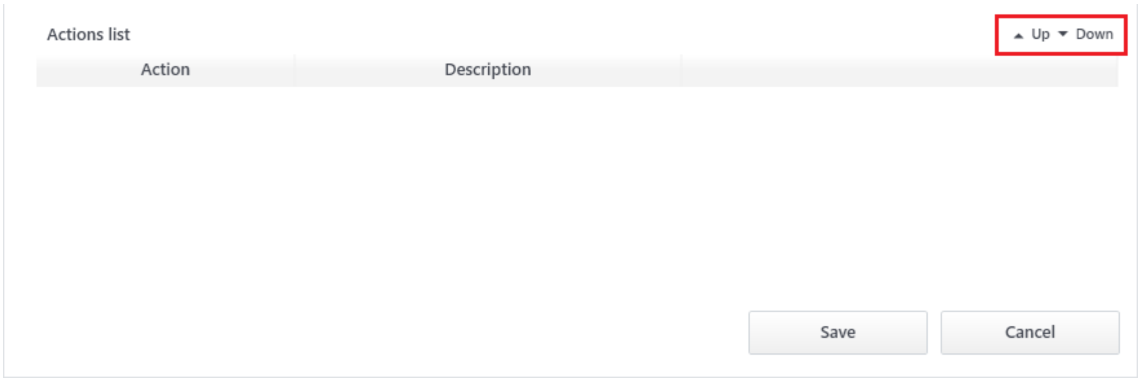


Fig. 2. Buttons for changing the order of commands

Some fields remain unchanged for all actions, they are all fields for browser settings, "Action" and "Note".

Browser customization fields

The Driver field is currently uneditable and is set to Chrome by default, but other browsers are planned to be added to the list for the foreseeable future (FireFox for example).

The "Proxy Server" - "Proxy Password" fields are intended for entering the proxy server settings, if necessary.

The "Arguments" field - if you need to start the browser with some arguments, for example, to open with a certain language or through the incognito mode (the necessary ones can be found on the Internet), then these arguments should be entered in this field.

The "Agent" field is intended for advanced users - it is responsible for the User Agent (the identification string of the client application, using a specific network protocol), under which the browser will be launched. If the field is empty, the agent will be read from the program configuration, where its default value is specified.

Field "Path to folder with user profiles" - in order to run not a new instance of the browser, but with an authorized profile (with history saving of the browser, authorization on sites, etc.) you must specify the path to the folder with this profile. If the specified path does not exist, a new folder will be created with the profile, in which the current activity in the browser will be saved. Folders with user profiles in Google Chrome are located in the directory "C:\Users\Username\AppData\Local\Google\Chrome", where your Windows user must be in place of Username. The field must contain the path to one of those folders that are located in the above directory (for example, "C:\Users\Username\AppData\Local\Google\Chrome\User Data").

The field "Path to Browser" is necessary in cases of special use of the browser (a different instance, non-standard Chrome) or chromium-based browsers (for example, chromium-gost). If this field is not filled in, the Google Chrome address is taken from the Windows registry. If for some reasons the value cannot be obtained, it is taken from config of Lexema RPA Studio.

Drivers for working with browsers

When working with Google Chrome, the module runs on the chromedriver driver that requires updating every time the version of the browser it works with is upgraded. When launching the module, if the driver is out of date, the program will display one of the errors:

"The chromedriver version was different from the browser version. The update was performed, run the script again" - the program managed to update the driver on its own, in this case you need to run the robot again;

"The driver version does not correspond to the browser version. Update chromedriver in the program folder by downloading it by following the link - https://chromedriver.storage.googleapis.com/{version number}/chromedriver_win32.zip" - the program failed to update the driver by itself, but managed to determine the required version for your browser, you should follow the instruction in the error description;

"The driver version does not match the browser version. Update chromedriver in the folder with the program by downloading it via the link - https://chromedriver.chromium.org/" - the program failed to update the driver by itself, you need to determine the version of your browser and find the appropriate version of the driver on the above link, then download chromedriver_win32.zip and unzip it into the folder with Lexema-RPA Studio.

When working with Mozilla Firefox the module works on the driver geckodriver, which also depends on the version of the driver used. This driver has to be updated independently. To do that you need to go to https://github.com/mozilla/geckodriver/releases, find the version you are interested in, find the Assets section in it, download geckodriver-version_number-win32.zip and unzip it into the folder with Lexema-RPA Studio.

Module actions

Module offers 8 basic actions - go to page, reload page, get page address, go back or forward, work with page, work with tabs and close browser. Each action requires filling in certain fields.

Note that the module creates its own instance of the browser, without authorized accounts, if the "Path to folder with user profiles" field is not filled in. The created browser closes itself at the end of the robot. You can not close the browser by itself!

Description of the fields that do not change depending on the action:

The "Action" field - is a drop-down list of possible actions of this module. They will be described in more detail in the next chapter.

The "Pause" field waits for a specified number of milliseconds after the action is performed.

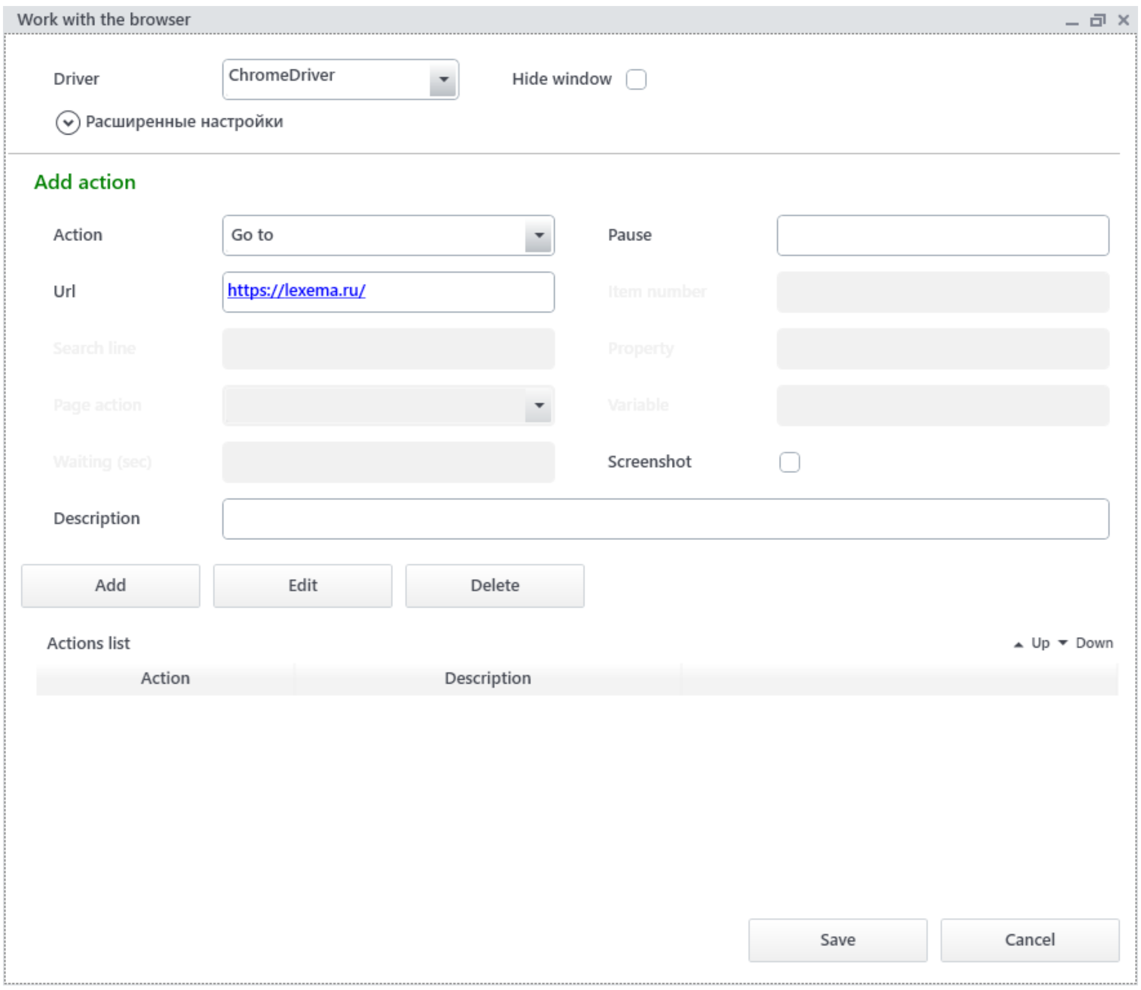
Field "Screenshot" - if checked, takes a screenshot after the command execution and saves it to the folder screen, located in the folder with the program. Name of the image - description of the action + ".bmp".

The "Note" field is the line with the comment to the command.

List and description of actions

"Go To" Action

Using the "Go to" action you can jump to the specified page. In the area of command's creation the field - "URL" will be opened. URL must be entered with protocol, i.e. http/https.


Work with the browser

Driver: ChromeDriver

Hide window ☐

Action: Go to

Url: https://lexema.ru/

Screenshot ☐

Pic. 3. Example of "Go to" action setting

If any page is already open, the browser will switch from it to the page specified in the "URL" field, i.e. the original page will be replaced by the new one. If the browser has not yet been launched, then it will be opened from that page.

The "Reload Page" action

The action allows you to reload a page that is already open. Before using this action, make sure there is an open page in the browser.

The action does not have any additional fields to configure.

The "Get page address" action

The action gets the address of the currently open page and writes the result to a specified variable.

Go Back action

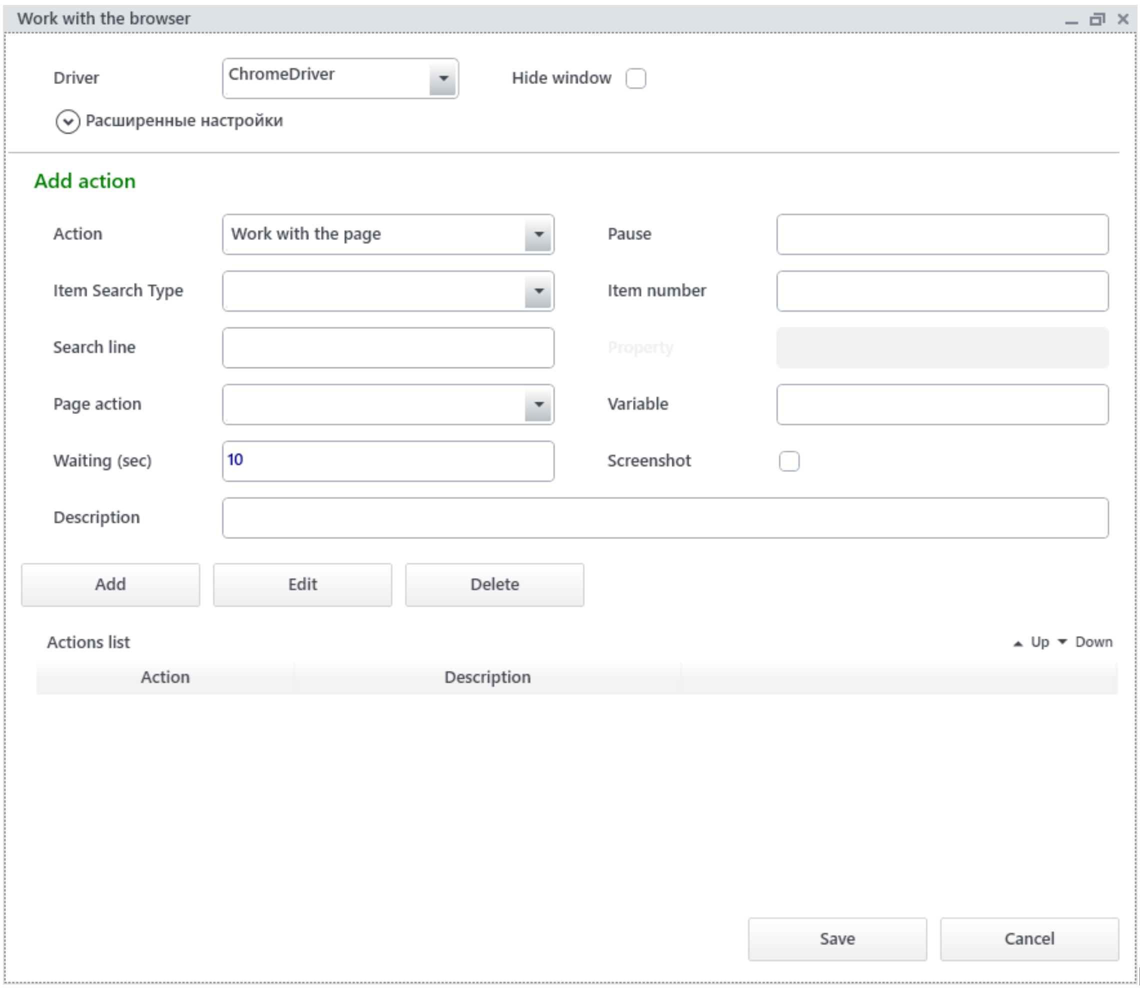
The action allows you to jump to the page that was open in the browser before the current one.

Go Forward action

The action allows you to jump to the page that was opened in the browser after the current one, if there was one.

Work with webpage action

This action allows you to work with an individual page element. The following fields appear to configure the action: "Item Search Type", "Search String", "Action on Page", "Waiting" and "Item Number". Two more fields - "Property" and "Variable" become available depending on the choice of action on the page.



Driver	ChromeDriver	Hide window	☐
Action	Work with the page	Pause	
Item Search Type		Item number	
Search line		Property	
Page action		Variable	
Waiting (sec)	10	Screenshot	☐
Description			

Action	Description

Fig. 4.

Fields for setting up the "Working with page" action

Field "Element search type" is a drop-down list with a list of methods of searching the element on the page, it is used together with the field "Search line".

Pay attention! Variables and ready-to-use values can be used in the "Search String" field, but there is no possibility to compose complex lines of code directly in this field. To compose the code you need to use the "Calculations" module, write the resulting code into a variable and then feed the variable itself into this field in the module.

All types of element search with examples of filling the "Search string" field will be listed below:

- GetElementsByClass - Find elements by their class properties.
- GetElementsByCssSelector - Finds elements by their CSS selector.
- GetElementByLinkText - search for link elements by their text. Only elements with a complete match are searched.
- GetElementsByPartialLinkText - search for link elements by their text. All elements containing the given wrapper are searched.
- GetElementsByTagName - Find elements by their tag.
- GetElementsByXPath - search for elements in the HTML language - xpath.

To get the xpath string for the element you need in the developer console in the browser, you need to right-click on this element in the code - copy - XPath:

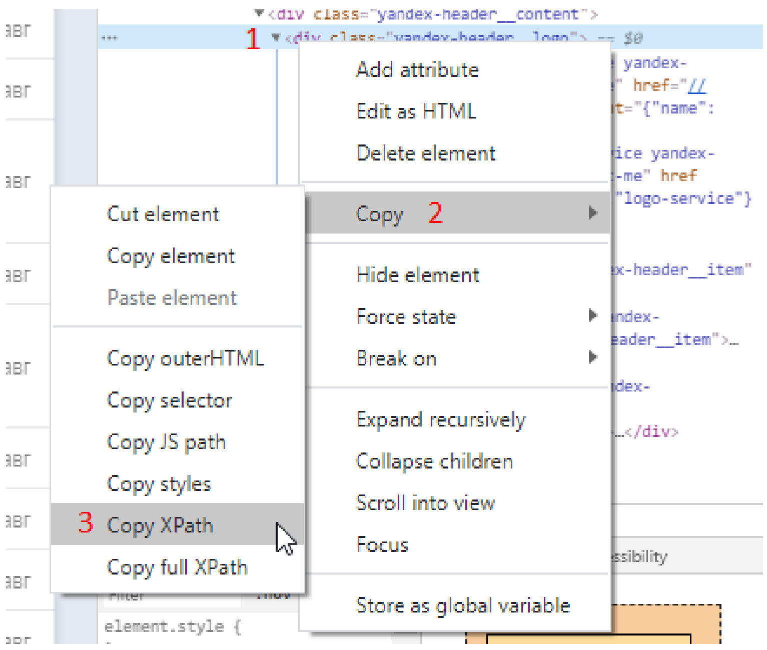


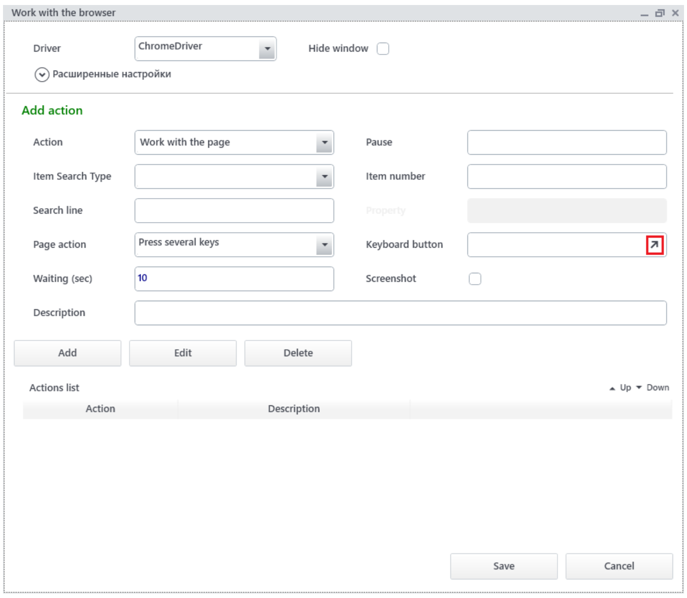
Figure 5. Getting the xpath string from the browser console

You can also use XPath to write your own queries to find an element on a page based on some fields, parts of them, or other properties of the element. В интернете много ресурсов по описанию языка XPath, как пример, один из них: https://msiter.ru/tutorials/xpath/syntax.

The "Action on the page" field offers a choice of actions from a drop-down list to be performed on the elements found. The following actions are available:

- Write - enters the text specified in the "Variable" field (for example, after clicking on a particular element to enter text);
- Get text - gets the value of the "text" property of all found or only the selected element and returns the result as a list to a variable;
- Send - pressing the key "ENTER" - application of the recorded data (for example, you can enter some text in the search line with the action "Write" and then "Send" it);
- Click - left mouse button clicking on a specified element (it is obligatory to fill the "Element number" field);
- Hover - placing the mouse cursor over the specified element (filling of the "Element number" field is obligatory);
- Get Property - the "Property name" field appears when you select the action - it returns the value of the property specified in the "Property name" field for all the found or the specifically defined element. The result will be a list;
- Get CSS Property - when you choose the action, the CSS Property field appears - it returns the value of the CSS property specified in the CSS Property field of all the found elements or the one specified. The result will be a list;
- Get Attribute - the "Attribute" field appears when you choose the action - it returns the value of the specified in the "Attribute property" field property of all the found elements or the one you've specified. The result will be a list;
- Get Elements - returns the list of specified elements for further work with them;
- Execute JavaScript - this action allows you to execute JavaScript code in the browser to retrieve some information. To write the result of code execution into a variable, you need to add "return" at the beginning of the code, for example "return document.querySelectorAll("[height='24']").length" (such a query will return the number of elements on the page whose height is 24);

- Get screenshot - takes a screenshot of the current screen and saves it to the path specified in the "Variable" field. The path must be specified in full - with the name and extension of the image to be saved;
- Press button on the keyboard - allows you to press the button on the keyboard inside the browser. Buttons are pressed in relation to the specified item, not just the browser page;
- Press multiple buttons on the keyboard - allows you to press combinations of buttons on the keyboard within the browser. Buttons are pressed in relation to the specified element, not just to the browser page. To fill this action, you need to click on the arrow in the corner of the "Keys" field, and then a window will open, in which you need to add to the table part, successively, those buttons that are involved in the desired combination. The "Keys" field itself is not available for editing.


Work with the browser

Driver: ChromeDriver
Hide window
Расширенные настройки

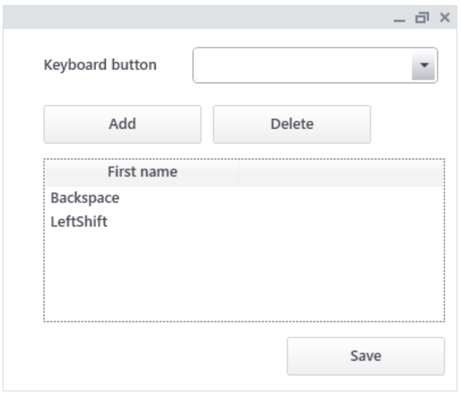
Add action

Action: Work with the page
Pause:
Item Search Type:
Item number:
Search line:
Property
Page action: Press several keys
Keyboard button:
Waiting (sec): 10
Screenshot
Description:

Actions list

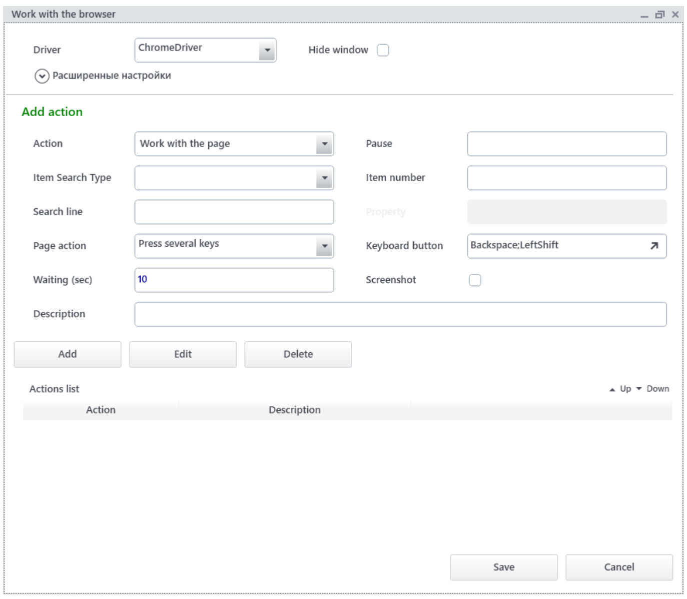
Action	Description

Fig. 6. Button to get to the window to select buttons for combination


Keyboard button:

First name
Backspace
LeftShift

Fig. 7. Key selection window for duplication



Work with the browser

Driver	ChromeDriver	Hide window	☐

Расширенные настройки

Add action

Action	Work with the page	Pause	
Item Search Type		Item number	
Search line		Property	
Page action	Press several keys	Keyboard button	Backspace;LeftShift
Waiting (sec)	10	Screenshot	☐
Description			

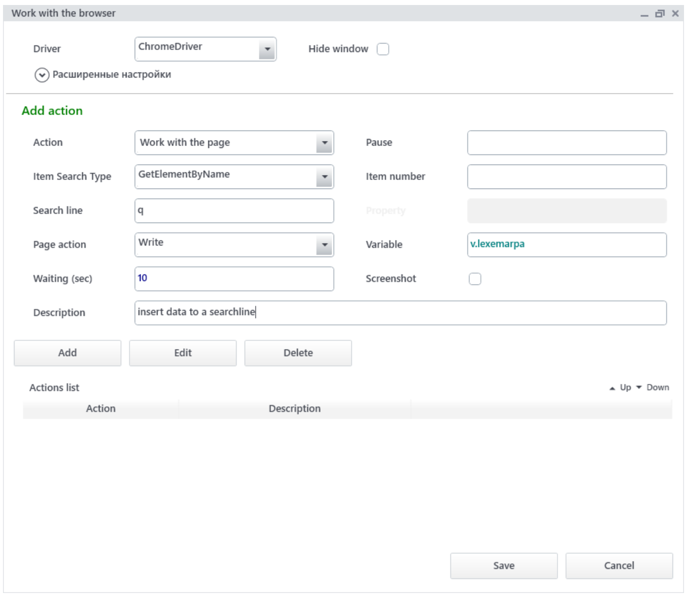
Fig. 8. The result of the key selection in the "Keys" field

In the "Variable" field you must enter the variable name, beginning with "v.", in which the result of the command will be placed or, conversely, from which the values for its execution will be taken.

Field "Waiting (sec)". In the field enter the number of seconds given to the program to find the element on the page.

Field "Item number" is intended for indicating of an index of the found item to which you want to apply an action.

Example of completed action:



Work with the browser

Driver	ChromeDriver	Hide window	☐

Расширенные настройки

Add action

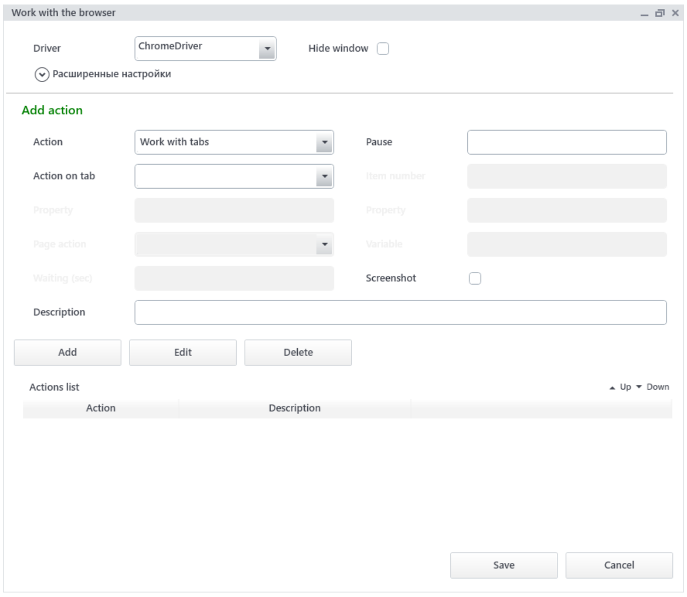
Action	Work with the page	Pause	
Item Search Type	GetElementByName	Item number	
Search line	q	Property	
Page action	Write	Variable	v.lexemarpa
Waiting (sec)	10	Screenshot	☐
Description	insert data to a searchline		

Fig. 9. An example of the action

Figure 9 presents an example of the action configured to search an element with the name "q". After the element is found, the text "Lexema-RPA" will be inserted into it.

Working with Tabs action

This action allows you to manipulate the tabs in the browser window. At least one field - "Action" opens to configure the action, the rest open depending on the action selected in it.



Driver: ChromeDriver

Action: Work with tabs

Fig. 10. Fields for configuring "Work with tabs" action

Available actions for working with tabs:

- Add tab - creates a new tab and it becomes active;
- Close tab - closes the active tab;
- Go to tab - when selecting the action, the "Tab number" field appears - goes to the tab, the number of which is specified in the "Tab number" field (starting from 0);
- Get number of tabs - the "Variable" field appears when choosing the action - returns to the specified variable the number of tabs in the browser.

Close Browser action

Closes the browser opened with the module, does not require additional settings.

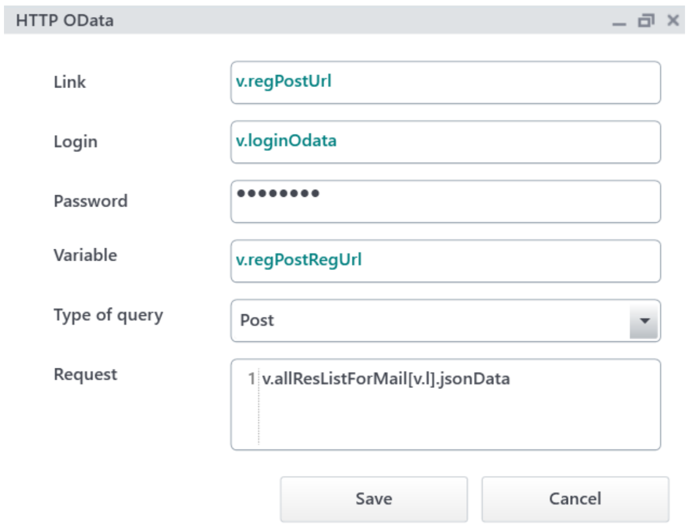
HTTP ODATA module

Http-requests - requests for transferring information between sites (or servers) and clients (such as us).

Module "HTTP OData" is designed to send requests via http to the integration service 1C - OData.

Module interface

This module consists of the following fields: "Link", "Login", "Password", "Variable" and "Type of request.



Link	v.regPostUrl
Login	v.loginOdata
Password	••••••••
Variable	v.regPostRegUrl
Type of query	Post
Request	v.allResListForMail[v.l].jsonData

Pic. 7. Example of configuration of the GET request

The field "Link. In this field the necessary URL is entered.

Field "Variable". In the field, the name of the variable, in which the response will be placed.

Fields "Login" and "Password". You must enter the login and password for authorization.

Field "Request type". Allows you to choose the necessary type of request. Methods "GET", "PUT", "POST" and "PUTCH" are available.

Field "Request". Into the field entered HTTP-message, according to the general rules of its structure. It's allowed only if you choose the "PUT", "POST" and "PUTCH" request types.

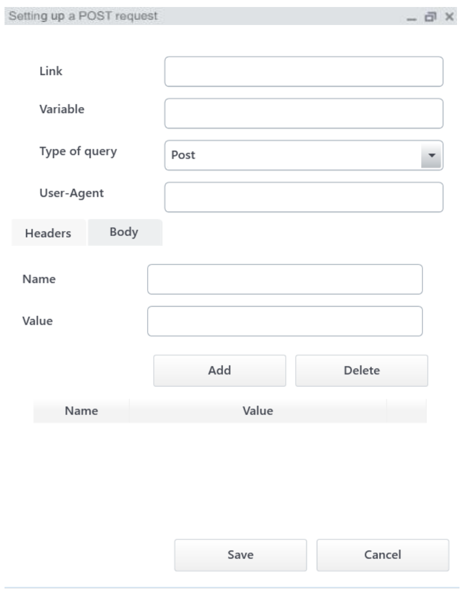
HTTP REQUEST module

Http-requests are requests for transferring information between websites (or servers) and clients (for example, us).

Module "HTTP Request" is intended for sending requests via http protocol.

Module interface

This module contains fields: "Link", "Variable", "Request Type" and tabs Headers and Body with the same Name and Value fields. At the bottom of the module there is a table with the added values of arguments.



Pic. 8. Interface of the "HTTP request" module

Field "Link". In this field you must enter the necessary URL.

Field "Variable". In the field enter the name of the variable, in which the response will be placed.

Field "Request type". Allows you to select the desired type of request. Methods "GET" and "POST" are available.

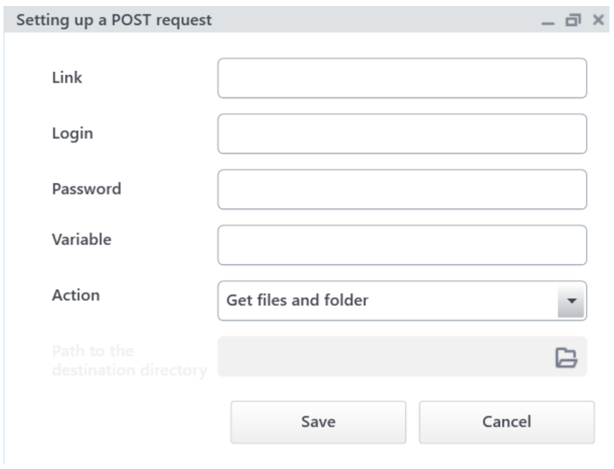
Fields "Name" and "Value" of tabs "Headers" and "Body" are used to compose the http-request.

FTP module

FTP module allows you to send requests using the FTP protocol.

Module interface

The module consists of several fields for connecting to the server, entering the name of a variable and a drop-down list with the choice of action.



Setting up a POST request

Link	
Login	
Password	
Variable	
Action	Get files and folder
Path to the destination directory	

Fig. 9. Interface of the module "FTP request"

The "Link" field is intended for entering a link to connect to the FTP server.

The "Login" field is filled with the login of the user connecting to the server.

The Password field is filled with the password of the user to access the server.

The "Variable" field is used to enter the name of the variable, in which the result of the module's execution will be placed.

The "Action" field provides a list of available actions with the FTP server.

RUCAPTCHA modules

Lexema-RPA Studio contains a set of modules that allow you to enter captcha on browser pages. This functionality works with the paid service RuCaptcha - https://rucaptcha.com/.

RuCaptcha upload image module

This module is needed to upload a screenshot of the captcha and start the character recognition process.

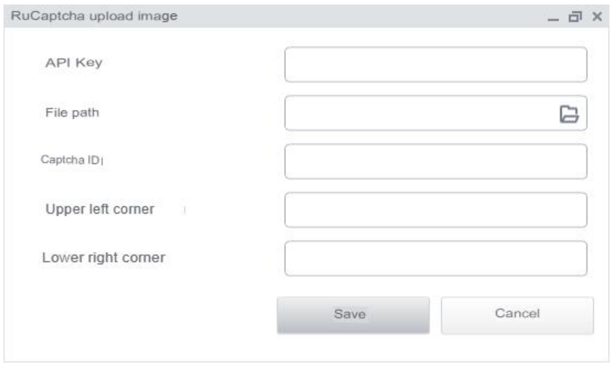


Fig. 24. interface of the RuCaptcha upload module

Description of the module fields:

API key - enters the key, which is given at the service's website when it is purchased;

Path to file - path to captcha screenshot;

Captcha Id - specify a variable to identify the captcha - this identifier will later be used to obtain the result;

Left top corner, right bottom corner - if you have uploaded a screenshot, where not only a captcha image is presented, you have to first calculate the coordinates of the left top and right bottom corners of the captcha. If you have uploaded only a captcha, then these fields do not need to be filled. Format of filling - x;y.

RuCaptcha get reply module

This module allows you to get response from service, which has been launched earlier.

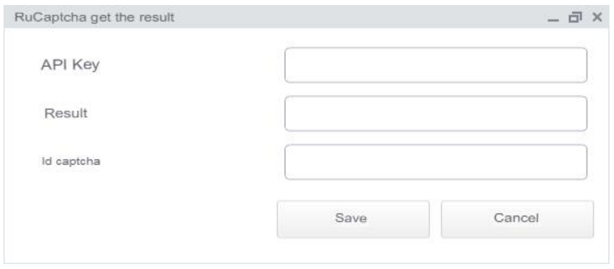


Fig. 25. interface of "RuCaptcha get reply" module.

Description of the fields of the module:

API key - the key is entered, which is issued on the service's website when it is purchased;

Result - enter a variable in which the result of captcha recognition will be placed;

Captcha Id - you must specify the captcha identification variable, the one specified in the captcha loading module.

TELEGRAM module

TELEGRAM module allows the use of the messenger Telegram in robots - with it you can read and send messages, attachments using telegram-bots.

Please notice: the module is currently under development and works solely from the program itself (does not work through the orchestrator).

Module interface

The module interface consists of a settings block, area for adding commands, table with the list of commands and buttons "Save"/"Cancel".

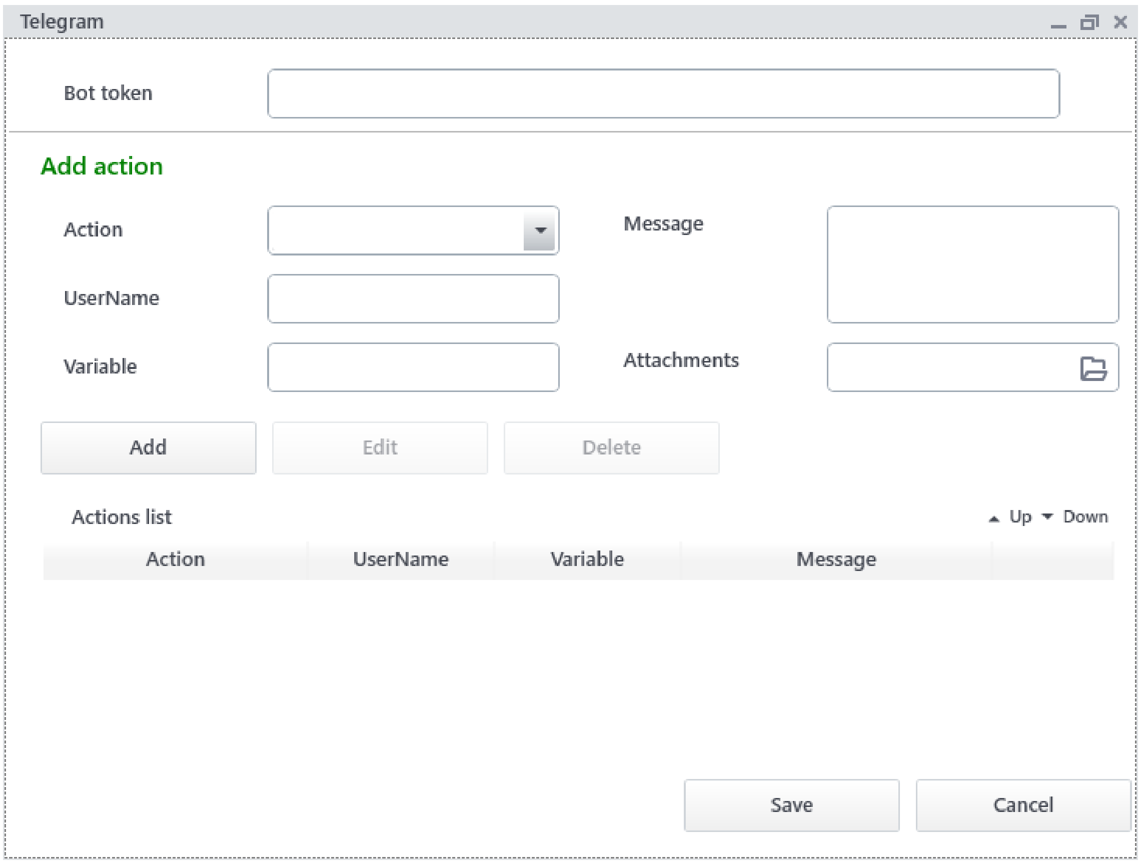


Fig. 1. Window of the Telegram module

The settings block consists of one field - "Bot Token". In this field you should enter the bot's token that is issued when registering the bot.

Command creation

A command is created as follows - you should choose the necessary action, fill out other fields, and press the "Add" button. The command will be added to the "List of commands".

Editing a command

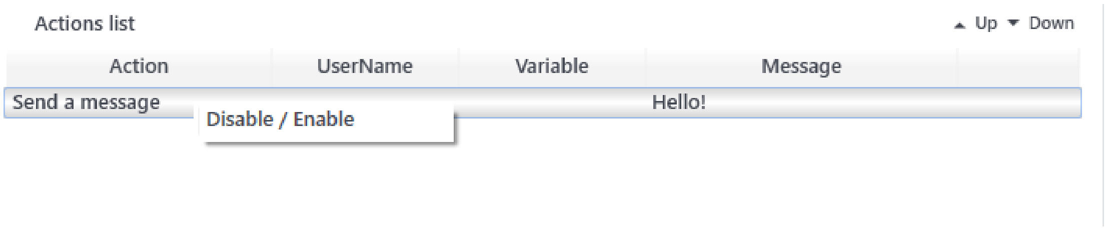
To edit a command, select it in the list of commands by clicking the left mouse button. After that all fields in the Add command area will be filled out according to the selected command. Change the fields that you want to edit and, having made sure that the desired action is still selected in the list of commands, click the "Edit" button.

Deleting a command

To delete a command, select it in the list of commands and click on the "Delete" button.

Disable/enable commands

Created commands can be disabled and enabled again. Disabled commands will not be executed. You can do this by right-clicking on the line corresponding to the command you want to disable and choosing "Disable/enable".



Actions list

Action	UserName	Variable	Message
Send a message			Hello!

Fig. 2. Context menu of the command

Selecting a file/folder

The button in the "Attachments" field opens a standard file selection dialog box.

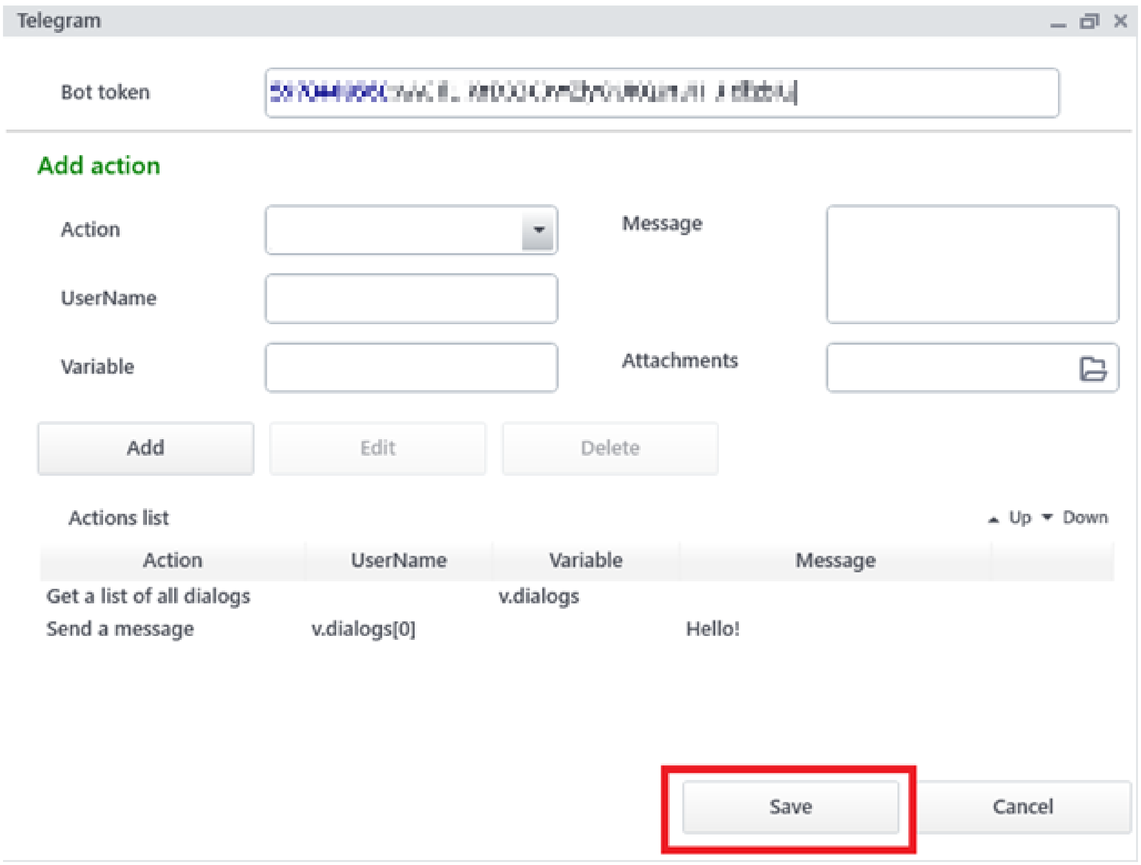
Changing the order of actions

The buttons "Raise" and "Lower" are provided to change the order of actions. These buttons are located at the top right of the list of commands.

Changing the order of actions is necessary when you want to add an action that was not foreseen before and was not included in the list of commands. In this case, you can add an action in the standard way, and then move it.

Saving and exiting

After all the necessary actions have been added to the list of commands, the module must be saved. This is done by pressing the "Save" button in the bottom right corner of the module.



Telegram

Bot token: [redacted]

Add action

Action:

UserName:

Variable:

Message:

Attachments:

Actions list

Action	UserName	Variable	Message
Get a list of all dialogs		v.dialogs	
Send a message	v.dialogs[0]		Hello!

Figure 3. "Save" Button

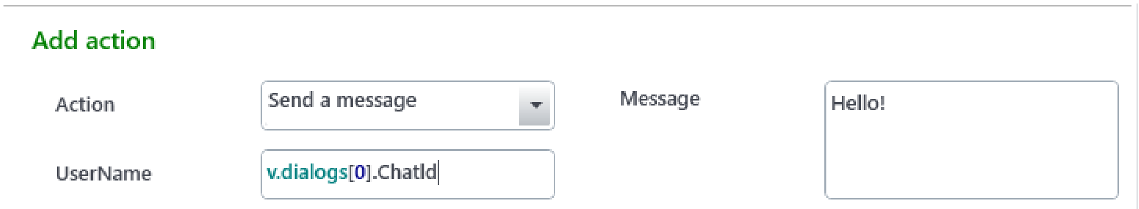
Actions

Send message action

This action allows you to send a message to the user or to the chat.

In the "User" field you should specify the identifier of the dialog or the name of the group the message should be sent to. The identifier can be obtained only with another module action, it is not visible from the application.

To send several attachments, you should separate the path to each attachment with ";". Each attachment will be sent as a separate message. If the message text is also filled in, then each attachment will be sent with this message text. If you need to send both text and multiple attachments, it is better to split it into two separate commands - sending attachments and sending text.


Add action

Action	Send a message
UserName	v.dialogs[0].ChatId
Message	Hello!

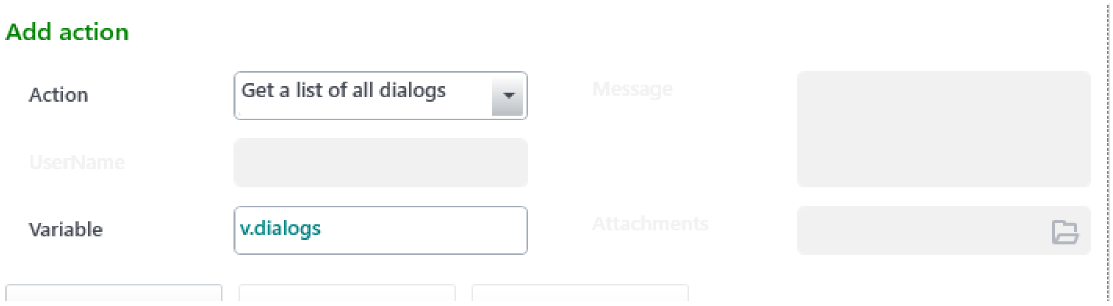
Fig. 4. Example of completing the "Send Message" action

Get a list of all dialogs action

Using this action you can get a list of all unread messages of the bot with users. Setting up the action consists only in specifying a variable where the result of the action will be written.

The variable will be an array of objects with the following fields: Message, Date, From, ChatId and ChatName. Field From is an object with fields id, first_name, username and language_code that stores information about the user who sent the message. The id from From is equal to ChatId, the second one was added to use less nesting.

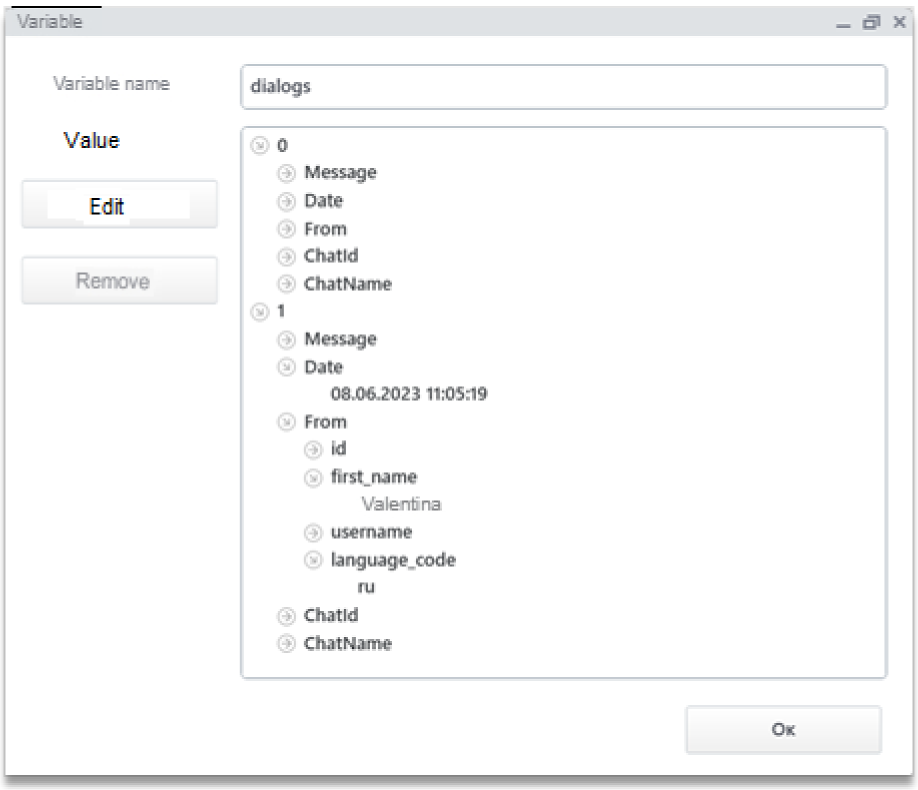
If the message is taken from a dialog with a user, the ChatId field is filled in, which can later be used to send him a message. If the message is taken from the chat/group, then the ChatName field is filled in, which should be used to send the message there.


Add action

Action	Get a list of all dialogs
UserName	
Variable	v.dialogs
Message	
Attachments	

Fig. 5. Example of filling the "Get the list of all dialogs" action

The result of filling the variable is shown in the next figure.



dialogs

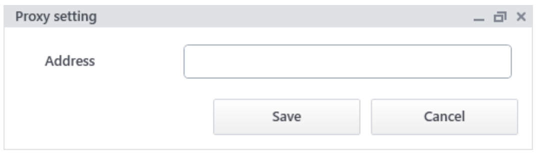
```
0
  Message
  Date
  From
  ChatId
  ChatName
1
  Message
  Date
    08.06.2023 11:05:19
  From
    id
    first_name
      Valentina
    username
    language_code
      ru
  ChatId
  ChatName
```

Fig. 6. Example of filling the variable in the result of the "Get list of all dialogs" action

PROXY module

The Proxy module creates the connection to the proxy server.

To use the module, fill in the address of the proxy server.



Pic. 17. Window of the PROXY module.

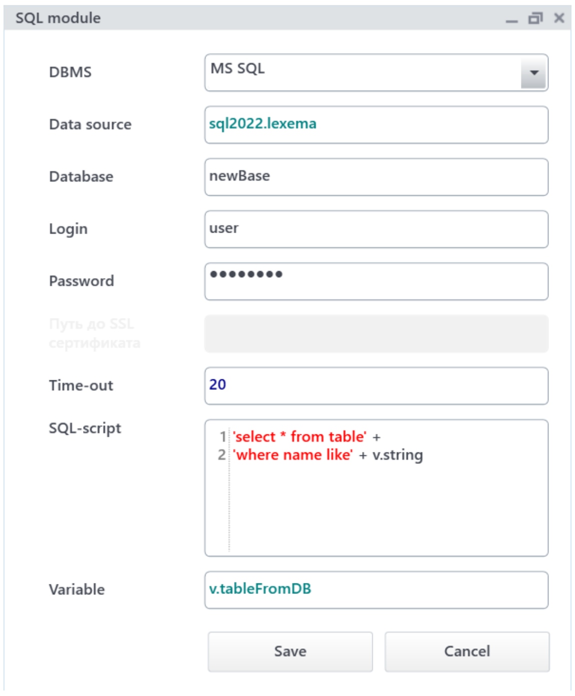
SQL module

SQL module provides work with databases. Supported DBMS: MS SQL, PostgreSQL and Oracle.

Module interface

This module provides work with the databases. To work with the database, you must fill in all the fields of the "Module SQL" window.

The window consists of the following fields: "DBMS", "Data source", "Database", "Login", "Password", "Waiting time", "SQL script" and "Variable".


SQL module

DBMS	MS SQL
Data source	sql2022.lexema
Database	newBase
Login	user
Password	••••••••
Путь до SSL сертификата	
Time-out	20
SQL-script	'select * from table' + 'where name like' + v.string
Variable	v.tableFromDB

Fig. 6. The example of filling the module

The fields "DBMS", "Data Source", "Database", "Login" and "Password" provide the connection to the database. It is necessary to enclose the line in quotes (for example, 'localhost:1521') when specifying the server and port through a colon.

Note: the "Database" field is not to be filled in for the Oracle database. In the "Login" field you can add the line ";Connect Mode=SYSDBA" to work with the system user.

The field "Waiting time". In this field you should enter the time value in seconds, which should be enough for the request to be executed. Without this field, the robot may work infinitely long because of exceptional situations, so filling this field is mandatory. For small queries the value may be 20 seconds, for larger queries it may be a minute or more.

SQL script field. A script in SQL language is entered in this field, the input of studio variables is supported. If you use studio variables, you must supply a string to this field in the following form: the whole script must be taken in quotes, except for the studio variables (as shown in Figure 11.40.1).

Variable field. The name of the variable in which the response from the database will be written in the form of a table is entered into this field. Only one table can be returned in one SQL module, subsequent ones will be ignored.

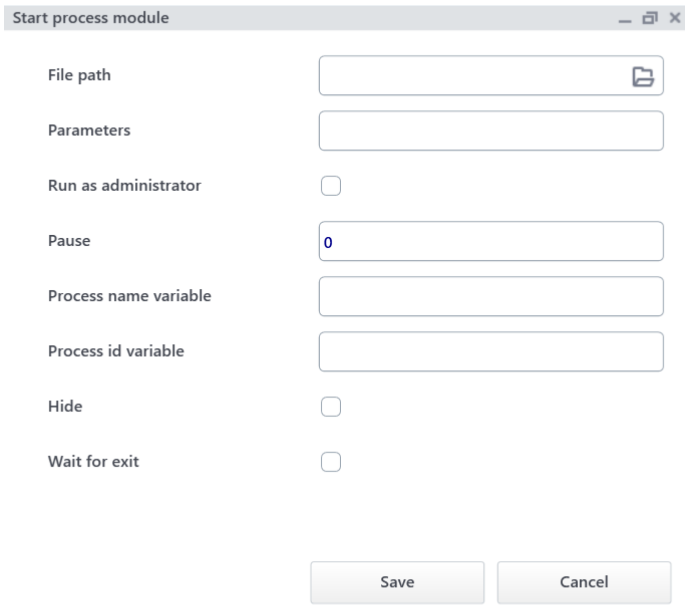
OPERATING SYSTEM modules

START PROCESS module

START PROCESS module is intended for launching any application. It can be a browser, MS Word, 1C, etc.

Module interface

Module window consists of "Path to file", "Options", "As administrator", "Pause", "Variable with process name" and "Variable with process id" fields as well as "Save" and "Cancel" buttons.

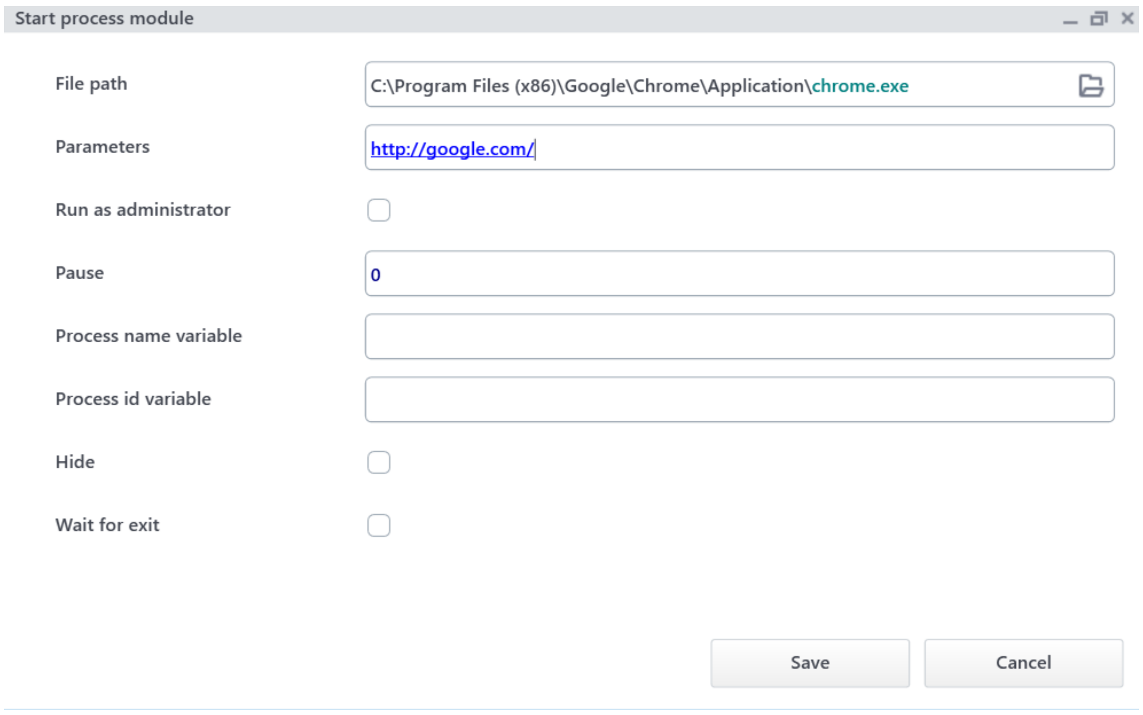

Start process module

File path	
Parameters	
Run as administrator	☐
Pause	0
Process name variable	
Process id variable	
Hide	☐
Wait for exit	☐

Fig. 3. Module window

Field "Path to file". In this field you should either enter the name of a standard Windows process (for instance, notepad) or the path to the root directory of the program, e.g. the standard path to Google Chrome browser is C:\Program Files (x86)\Google\Chrome\Application\chrome.exe.

The "Parameters" field is optional, it contains the arguments of the called application. For example, you don't need arguments to open a new notepad or a blank Excel sheet, but if you want to open a particular file, the name of the file is passed in the arguments, and if you open a browser, the argument can be a link to a website.


Start process module

File path	C:\Program Files (x86)\Google\Chrome\Application\chrome.exe
Parameters	http://google.com/
Run as administrator	☐
Pause	0
Process name variable	
Process id variable	
Hide	☐
Wait for exit	☐

Fig. 4. Example of filling

"As administrator" checkbox starts the process with permissions which allow it to make changes to the computer's system files. If you do not trust the application, do not check this box.

"Pause" field sets the waiting time after the process starts.

"Variable with process name/id" filed. Variables, where the name and id of the process to be started are placed respectively, are entered in this field. These parameters are needed to be able to terminate (close) specified processes later. If your purpose is just to start processes you can leave these fields empty.

"Hide" checkbox launches the process with "hide" property, i.e. it starts the background process.

The "Wait for completion" checkbox allows this module to wait for the process to finish before continuing the robot's work.

END PROCESS module

END PROCESS module is used to end a previously started process by its name and identifier.

Module interface

The window consists of two fields - process name and identifier of the terminated process.

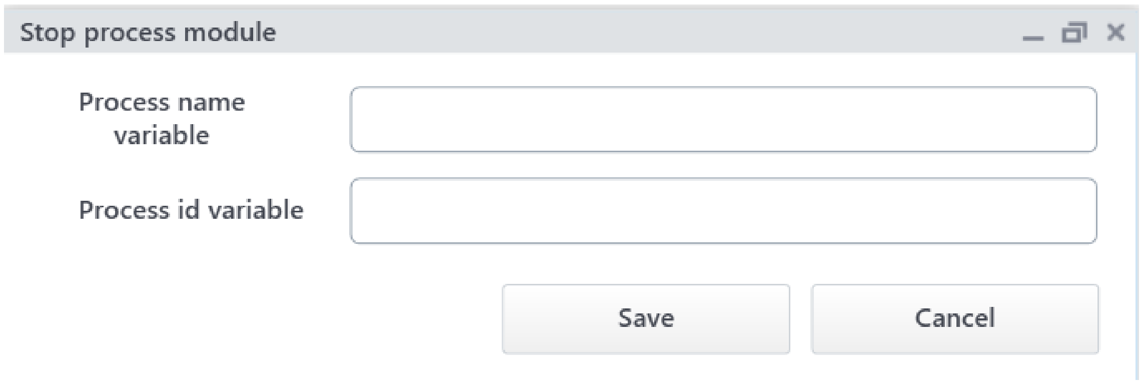


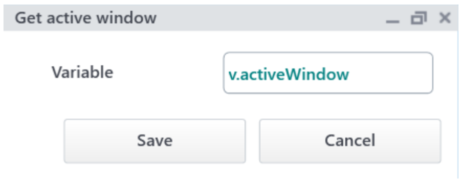
Fig. 5. Module window

If the terminated process was started via "Start Process" module these fields contain variables with the same names in the Process Startup Window fields. Otherwise if you know the name of the process to be terminated just enter it, but (!) in this case all processes with the same name will be terminated. For example, if you want to close an excel file and just write "Excel" then all excel files that have been opened will be closed.

GET ACTIVE WINDOW module

GET ACTIVE WINDOW module allows to get the name of the active window and writes it into a variable. It helps to determine which program is active at the moment, which simplifies working with the "Clicker" module. The module allows you to eliminate a lot of errors in the clicker robots.

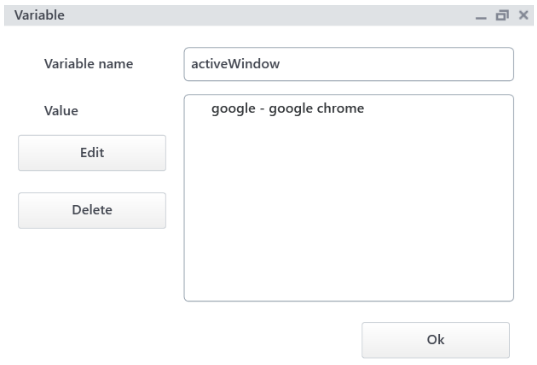
The module consists of one setting field - Variable, in which you must enter the name of the variable. As a result, the module writes the name of the active window in lower-case letters to the specified variable.


Get active window

Variable: v.activeWindow

Fig. 15. Example of filling the "Get Active Window" module

Example of a variable, filled in by the module:


Variable

Variable name: activeWindow

Value: google - google chrome

Fig. 16. Result of the module's work

COMMAND PROMPT module

COMMAND PROMPT module allows you to work with console applications by sending commands and getting a response to commands (cmd, python and other console applications).

Module interface

The module settings window consists of two sections - "Console settings" and "Commands". The console settings contain data about the application to be run, and the "Commands" section contains commands themselves, variables where the result will be written and command execution timeout.

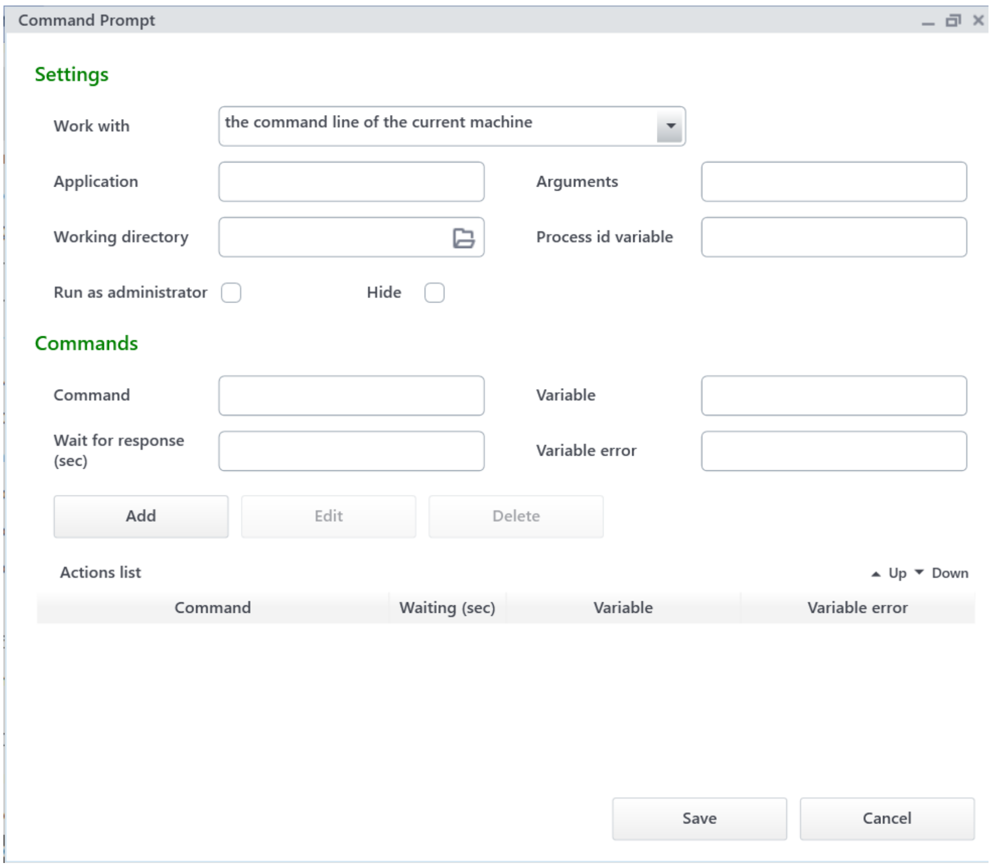


Fig. 5. Module window

Console settings

The list of fields of the section:

- "Working with" - drop-down list of two items - "working with the command line of the current machine" and "working with the server connected through ssh". If you choose to work with the current machine, the console settings fields will be as follows:
- "Application" - the name or path to the console application to be launched, e.g. "cmd";
- "Arguments" - the list of arguments to run the console with;
- "Working directory" - the path to the folder from which the console application should be called;
- "Variable with process id" - the field similar to the field of the same name in the "Start process" module - allows you to save process id to a variable for further reference or closing it;
- "As administrator" - launching the application with administrator privileges;
- "Hide" - launches the application in the hidden mode.

If connection to SSH server is selected, the fields will be as follows:

- "Host"- IP address of the machine to which the connection is made;
- "User" - the user under which you want to log in;
- "Password" - the password to access the system;
- "Process id variable" - the field similar to the field of the same name in the "Start process" module - allows you to store the process id in a variable for future reference or closing it.

Commands

This section consists of fields for creating commands, buttons to control commands and a list of already created commands. List of fields for command creation:

- "Command" - the command itself, which should be entered in the console. It is allowed to use variables;
- "Waiting for a response (sec)" - field for entering the number of seconds to wait for a response from the application to the command. The default is 1 second;
- "Variable" - the field for entering the name of the variable, in which the result of the command will be written in case of its success;
- "Variable error" - the field for entering the name of the variable, in which the error of the command in case of its failure will be written. This field is not available for ssh server, as all responses from it are written to the main variable.

Creation of the command starts with filling in the above fields. Once you have filled them out, you should press the "Add" button and the command will be added to the list of commands executed by the application, you will see it in the "Command List" table below the buttons.

CLIPBOARD module

The module allows you to work with the clipboard of the device.

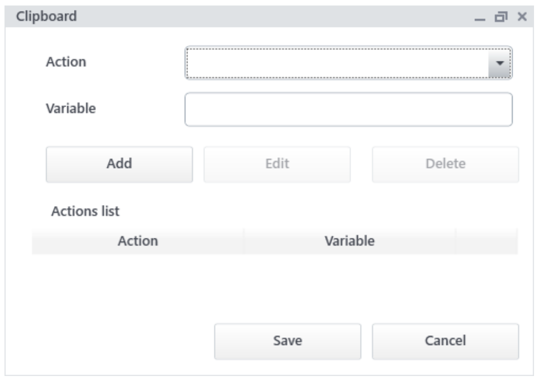


Fig. 26. The Clipboard module interface

This module offers the following actions to work with the clipboard:

1. Write to clipboard;

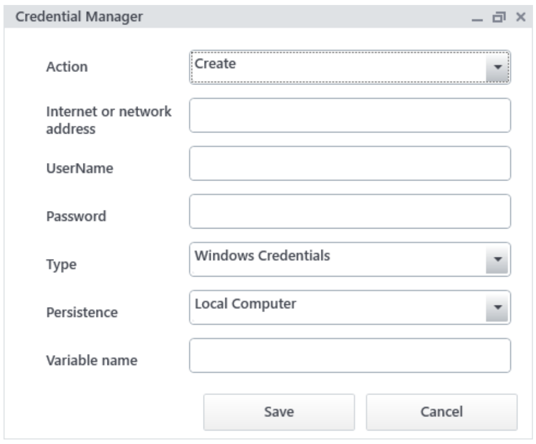
2. Read from clipboard;

3. Clear the clipboard;

4. Check if a string is presented in the clipboard.

CREDENTIAL MANAGER module

CREDENTIAL MANAGER module allows you to work with Windows accounts - create new accounts, get logins and passwords of existing accounts or delete them.



Credential Manager

Action	Create
Internet or network address	
UserName	
Password	
Type	Windows Credentials
Persistence	Local Computer
Variable name	

Fig. 23. The interface of the "Account manager" module

The Action field is a drop-down list of possible actions with accounts - create a new account, fetch an existing one, delete an existing one;

In the "Internet or network address" field, enter the address of the network or site;

"User", "Password", "Persistence" and "Type" fields are required only to create a new account.

In the "User" field enters the login of the account, in the "Password" field, respectively, the password of the account.

The "Type" field offers three account types to choose from - "Windows credentials", "Certificate-based credentials" and "General credentials". Windows Credentials are names and passwords that you use to access network shares, Web sites that use integrated authentication, and when connecting to a remote desktop. Certificate-based credentials - These are used for smart card authentication. Shared credentials - used by third-party applications that require separate authentication with credentials other than those used for login.

The Persistence field allows you to select one of three options for storing credentials - "Local Computer", "Login Session" and "Enterprise".

In the "Variable name" field, enter a variable, to which the result of getting an account or creating a new one will be returned. The variable will be an object of two fields - Username and Password.

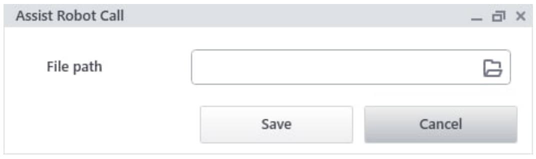
ROBOTS MANAGEMENT modules

LOCAL ROBOT module

LOCAL ROBOT module allows you to load into the script an already written robot that is located on the machine from which the robot will run.

The connected robot will use the same variables that are used by the current robot.

To configure the action, you only need to specify the path to the connected robot.


Assist Robot Call

File path

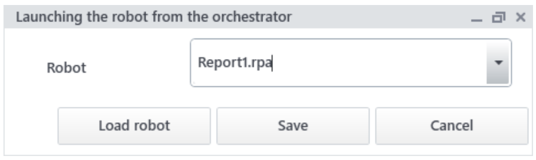
Fig. 19: Window of the Local Robot module

CLOUD ROBOT module

With CLOUD ROBOT module, it is possible to use an existing, published in the orchestrator, so called «robot in the robot» feature.

This version of the orchestrator robot has no input and output variables - it modifies and uses the variables of the main robot, as if being part of it.

To customize the action, select one of the robots offered by the program, i.e. those published by the current studio user.


Launching the robot from the orchestrator

Robot: Report1.rpa

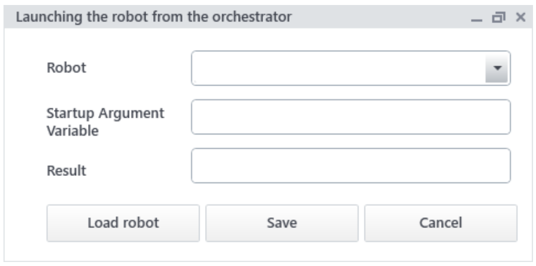
Fig. 22. Cloud Robot module window

Press the Load Robot button to load the selected robot in a separate tab of the studio.

ROBOT FUNCTION module

With ROBOT FUNCTION module it is possible to use an existing robot, published in the orchestrator, in the robot. The robot-function will work in its own isolated environment - it will have its own variables that do not overlap with the variables of the main robot.

To set up the action, select one of the published robots, specify the name of the input variable and the variable where the result of the robot's work will be written.


Launching the robot from the orchestrator

Robot

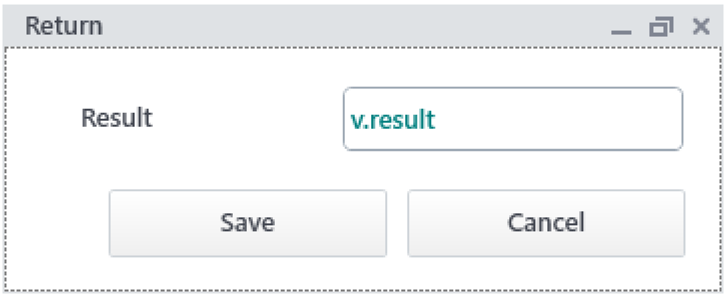
Startup Argument Variable

Result

Fig. 20. Robot Function module window

Variable-argument trigger. If your robot function needs several values, then create an object variable that has as many fields as you need. For example, a function that calculates some index over multiple values, n1 and n2, should take as input an object variable that has both of these fields (v.number.n1 and v.number.n2, as an example). Note that inside the robot function, you will have to treat the variable v.args as input, regardless of which variable you specify as input in the robot function module. That is, if you specified the variable "v. number" in the "Variable trigger argument" field of the module, then inside the robot function itself, the value of v.number will be in the v.args variable. This is done so that you don't have to remember the variable used in the function to use it.

Result. To fill in the variable-result inside the robot-function, you need to use the Return module and specify the variable in it whose value should be placed in the specified variable result.



Result: v.result

Fig. 21. Example of returning a variable from a robot function.

Press the Load Robot button to load the selected robot in a separate tab of the studio.

EMAILS MANAGEMENT modules

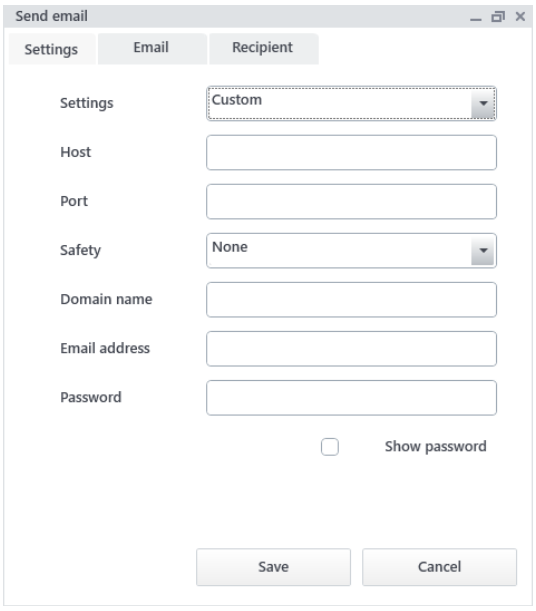
SEND EMAIL module

The Send Mail module is designed to send an email to one or more recipients.

Module interface

The module window consists of several tabs: 'Settings', 'Recipient', 'Letter'. Let's take a look on all of the tabs in order. Let's start with the "Settings" tab.

Settings tab

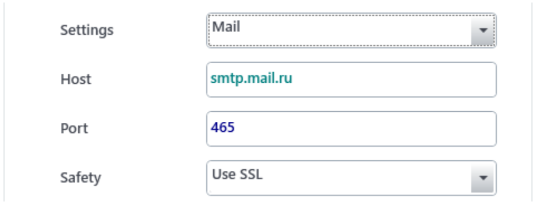


Send email

Settings	Custom
Host	
Port	
Safety	None
Domain name	
Email address	
Password	

Fig. 1. Module window

The "Settings" field is a drop-down list that offers several mail services. If you choose a mail service, the "Host", "Port", and "Security" fields will be filled out according to its settings. The "Custom" item provides the possibility to customize the access parameters of the mail service.



Settings	Mail
Host	smtp.mail.ru
Port	465
Safety	Use SSL

Pic. 2. Example of the "Mail" service settings

You should fill in the fields "Host", "Port", "Security" only if you choose the custom options. They represent a set of mandatory fields for connecting to mail services.

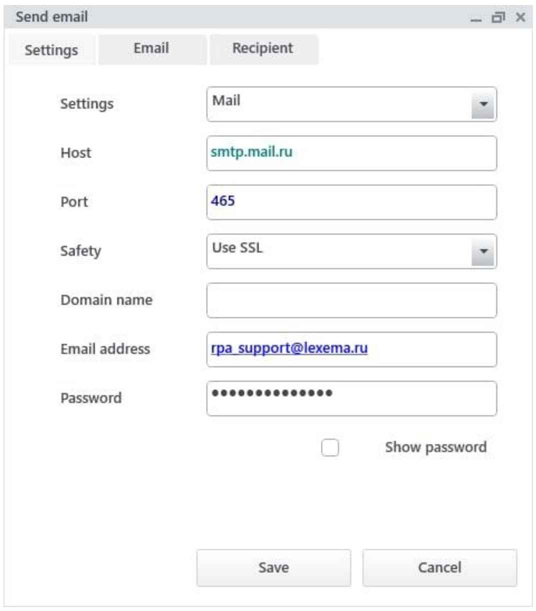
You can disable certificate security check if you are using a trusted host and your device has no access to external internet for certificate checks. To do this, set the "CheckCertificate" flag in the Studio configuration file to "false" (the default value is "true").

The "Domain Name" field is required for some mail services (mostly corporate ones) to specify the domain name, required for authorization.

"E-mail" field is intended for indicating the e-mail address, from which the messages will be sent, in the format name@mail.ru.

"Password" field is intended for entering the password from the e-mail for authorization.

"Show password" checkbox provides the possibility to show the password to check its correctness.


Send email

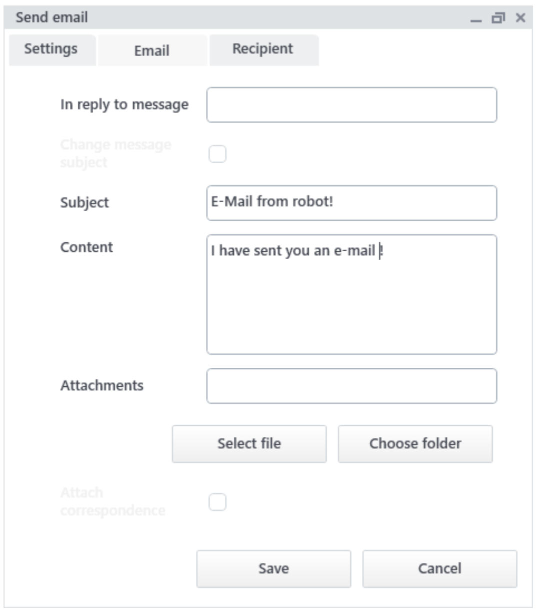
Settings	Mail
Host	smtp.mail.ru
Port	465
Safety	Use SSL
Domain name	
Email address	rpa_support@lexema.ru
Password	••••••••••••••
	☐ Show password

Fig. 3. Example of the filled in "Settings" tab.

E-Mail tab

This tab is used for setting up an e-mail that will be sent to the recipients specified on the next "Recipient" tab. All fields are optional, if you don't fill out any of them, an empty letter with the subject "<Unsubject>" will be sent.

You should fill out the "In reply to message" field only if you want to send an e-mail not as a separate one, but as a reply to some read message.


Send email

In reply to message	
Change message subject	☐
Subject	E-Mail from robot!
Content	I have sent you an e-mail !
Attachments	
Attach correspondence	☐

Fig. 4. Tab "Email".

To attach one file as an attachment, you should press the "Select File" button and choose the file you need in the dialog window that will open.

To attach several files to an attachment, press the "Select Folder" button and select the folder with all the required files in the dialog box that appears.

Each field of the form can be filled with a variable.

The "In reply to message" field must be filled with a variable e-mail, i.e. received with the help of the "Read e-mail" module, and it must be one e-mail, not a list of e-mails. When filling out this field, the "Change subject of message" and "Attach correspondence" checkboxes become available for choosing - when replying to the message, the subject can be left the same with "Re:" added to it in the beginning, or you can change the subject yourself by checking the box and entering your own subject; if you need to attach the correspondence, you should check "Attach correspondence", otherwise only the current message text will be sent. Also the "Reply to all" checkbox appears on the "Recipient" tab, if it is checked, the e-mail will be sent to all who participated in the correspondence, otherwise you can specify the list of e-mail recipients yourself.

Recipient tab

This tab is intended for filling out the information about the recipients of your mails.

The module provides you with different ways of specifying recipients, namely:

- manual input of the list of recipients (names and e-mail's);
- from a string-variable, as a single e-mail or a comma-separated list;
- from a variable-list, as several emails.
- check the "Reply all" box if the e-mail is a reply to another message.
-

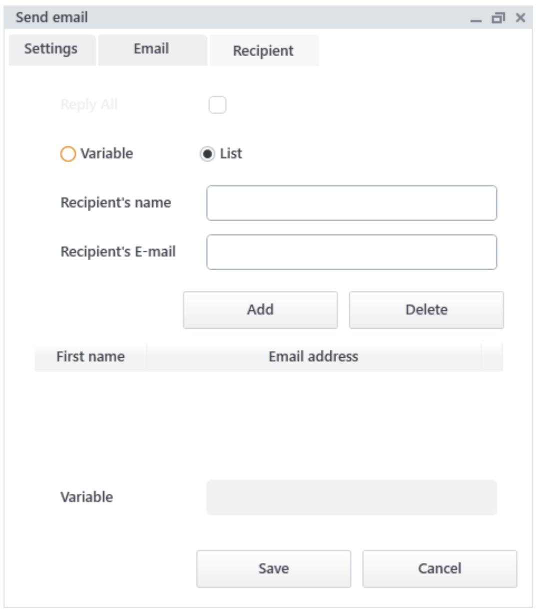
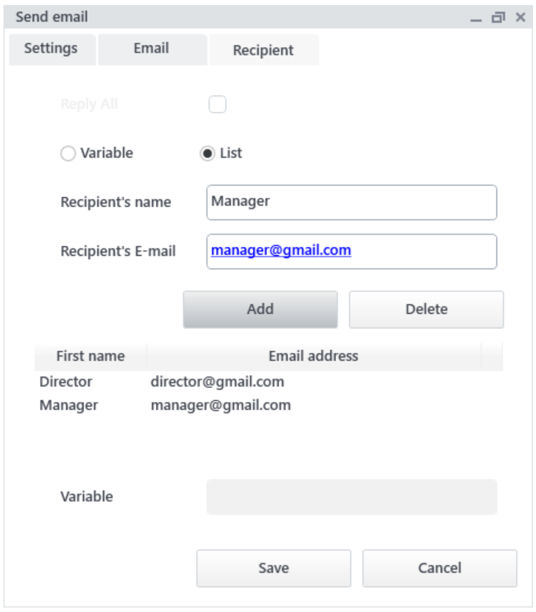


Fig. 5. "Recipient" tab

If you choose the "List" checkbox, then the "Variable" field is unavailable for entering. Otherwise, on the contrary, the "Variable" field is available, but other fields are unavailable.

To enter the recipients using the "List" method, you should fill out the "Recipient's name" and "Recipient's e-mail" fields, then click on the button under the "Add" fields. After that the entered values will appear in the table in the center of the window. If you want to remove any of the recipients from the table, you should choose the line with it and press the "Remove" button.


Send email

Settings | Email | Recipient

Reply All ☐

○ Variable ● List

Recipient's name: Manager

Recipient's E-mail: manager@gmail.com

Add | Delete

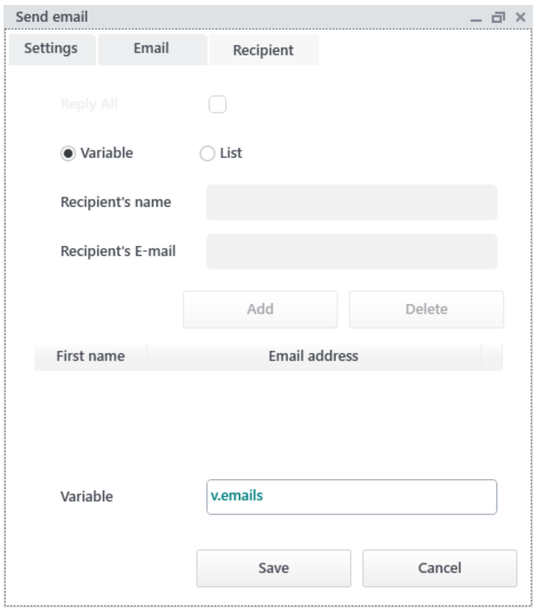
First name	Email address
Director	director@gmail.com
Manager	manager@gmail.com

Variable:

Save | Cancel

Fig. 6. Example of filling in the table of recipients

If you use the method of selecting recipients through a variable, the fields "Recipient's name", "Recipient's e-mail" and the table become inaccessible. In the "Variable" field you should enter the name of variable starting with "v." from which the recipient's emails will be taken. The variable can be a list of emails, or a string listing them separated by commas.


Send email

Settings | Email | Recipient

Reply All ☐

● Variable ○ List

Recipient's name:

Recipient's E-mail:

Add | Delete

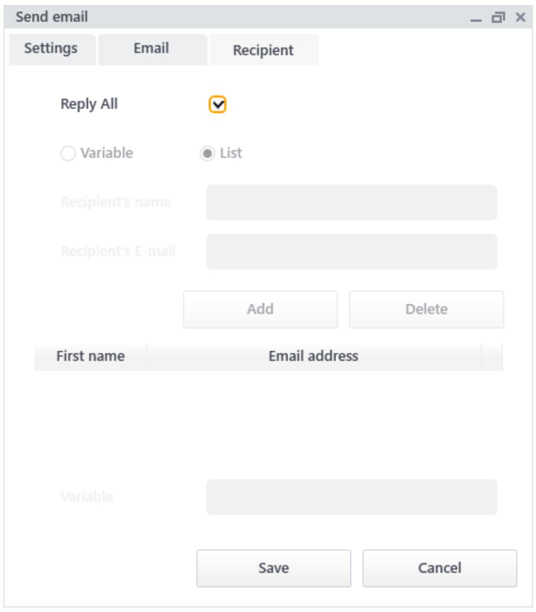
First name	Email address

Variable: v.emails

Save | Cancel

Fig. 7. Recipient tab

If the e-mail is a reply to a message, you have the possibility to check the "Reply to all" box. In this case the e-mail will be sent to all the correspondents, you will not need to specify the addresses manually.



Send email

Settings | Email | Recipient

Reply All ✓

Variable [✓ List]

Recipient's name

Recipient's E-mail

Fig. 8. Checkbox "Reply to all".

READ EMAIL module

Read e-mail module is intended for getting the list of e-mails for their further processing or just for downloading attachments. To enable the module, you must allow the connection via IMAP or POP3 protocol on the mail you are using (details about enabling the protocols are described in this chapter, point "Enabling IMAP and POP3 protocols").

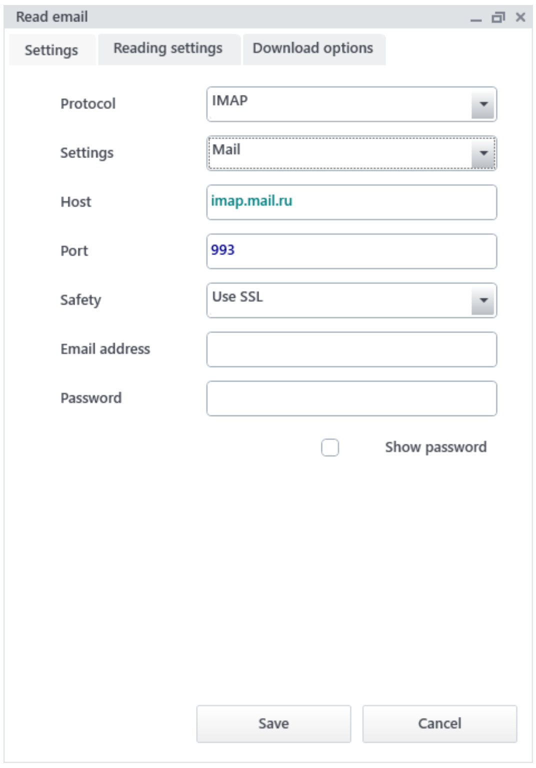
Module interface

The module consists of three tabs: "Settings", "Reading Settings" and "Download Settings".

Settings tab

Settings tab is similar to the tab of the same name in the "Send mail" module, with one major difference: this module gives you the choice of sending protocol: IMAP or POP3. The "Domain name" field is also missing, since it is not required for reading e-mails.

If you choose the corresponding protocol and its settings, the fields "Host", "Port" and "Security" are filled out automatically (except the "Custom settings" mode).

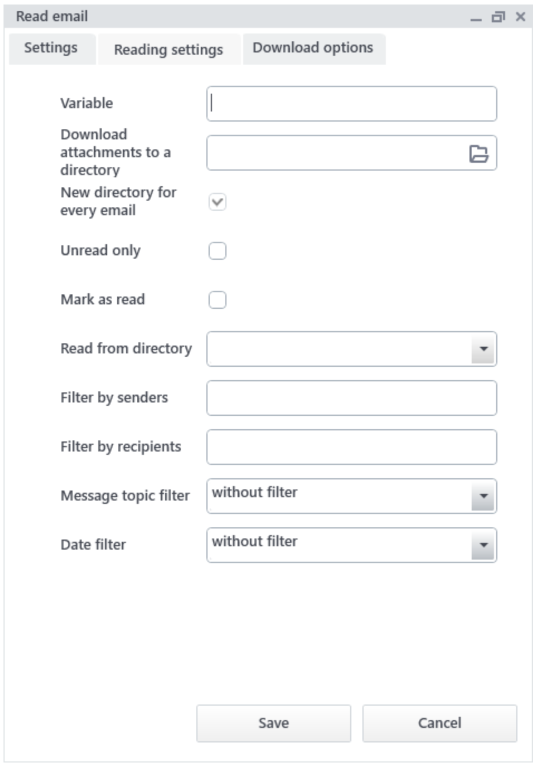


Protocol	IMAP
Settings	Mail
Host	imap.mail.ru
Port	993
Safety	Use SSL
Email address	
Password	

Fig. 9. Module window with an example of autofilling fields.

Reading settings tab

The "Reading Settings" tab is a set of fields that are filled out depending on what e-mails should be read.



Read email

Settings | Reading settings | Download options

Variable	
Download attachments to a directory	
New directory for every email	✓
Unread only	☐
Mark as read	☐
Read from directory	
Filter by senders	
Filter by recipients	
Message topic filter	without filter
Date filter	without filter

Save | Cancel

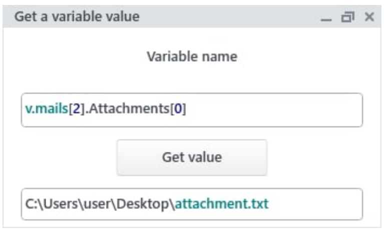
Figure 10. The "Reading Settings" tab.

The "Variable" field. You are expected to enter the name of a variable, in which the information about the read letters will be entered. In the program, this field will be represented as a list of objects with seven fields:

- Subject - subject of the e-mail;
- Body - body of the message, its body text;
- Sender - sender;
- Receivers - recipients listed separated by commas;
- Attachments - list of paths to the downloaded attachments;
- Date - date of receiving the mail;
- Id - unique identifier of the e-mail;
- References - references to other e-mails that can be set by the mail system itself. This field is necessary for correctly sending responses to messages;
- ForReplyId - message identifier, which is necessary to send a response message to this message.

Examples of operations with variables

v.mails[2].Attachments[0] - getting the path of the first attachment of the third read message:



Get a variable value

Variable name

v.mails[2].Attachments[0]

Get value

C:\Users\user\Desktop\attachment.txt

Figure 11. Example of getting the path to the first attachment of the second mails

v.mails[3].Subject - getting the subject of the fourth downloaded message:

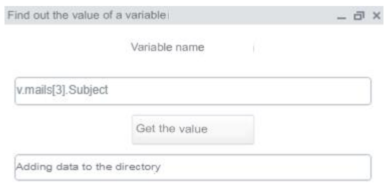

Find out the value of a variable
Variable name
v.mails[3].Subject
Get the value
Adding data to the directory

Fig. 12. Example of getting the subject of the first email

"Download attachments to folder" field is intended for entering the path to the folder, where all the attachments of the read mails will be downloaded. If this field is left empty, the files will be downloaded to a temporary directory of your computer.

"New folder for each e-mail" field allows saving attachments from each e-mail to a separate folder, so that files with the same names, but contained in different e-mails, do not interfere with each other. When downloading attachments to a temporary system folder on your computer, the checkbox is mandatory.

"Unread only" checkbox allows you to set only new mails to be read. It is available only for the IMAP protocol.

"Mark as read" checkbox allows you to set a mark as read after reading a message. It is available only for the IMAP protocol.

"Read from folder" field is intended for setting up a specific folder in your mailbox from which you want to read mails. By default, e-mails are read from the "Inbox" folder. This field is available only for the IMAP protocol.

"Filter by senders" field allows you to search for messages sent from one of the specified senders. You can list all of them separated by commas, or you can feed this field with a variable list containing senders.

"Filter by recipient" field allows you to search for messages that contain the specified recipients. You can list all of them separated by commas or by feeding a variable list containing recipients into this field. It is available for IMAP only.

The "Filter by subject" field is intended for setting the filter by the subject of messages. If you want to download e-mails with a subject that contains or does not contain a certain string, you should select the operation of the same name in the drop-down list and type the necessary string in the field that appears (Figure 13). It is available only for the IMAP protocol.

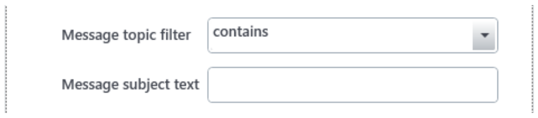


Figure 13: Filter by subject

"Filter by date" field is intended for adjusting the filtration of messages by date. If you specify this field, the robot will read only those messages which satisfy the specified condition.

The following filtering methods are available:

- for a certain date - "On date";
- starting from some date, including it - "From date";
- all messages up to and including any date - "Before date"; "Between dates".

Once you choose the filtering method, one or two fields will appear under the "Date filter" field to enter the necessary values in the format DD.MM.YYYY. This is available only for the IMAP protocol.

Download options tab

Using this tab, you can limit the information downloaded from the mail server to reduce the consumed resources. By default, all available information is downloaded.

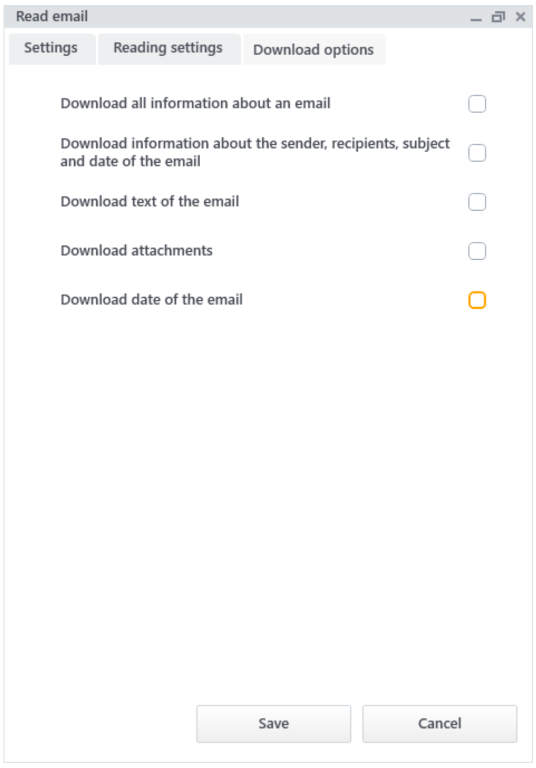
The following restriction for downloading is available:

Downloading information about the sender, recipients, subject, and date of the message (email header);

Downloading only the text of the message;

Downloading attachments (text is also downloaded by default);

Download date of the message only.



Download all information about an email ☐

Download information about the sender, recipients, subject and date of the email ☐

Download text of the email ☐

Download attachments ☐

Download date of the email ☐

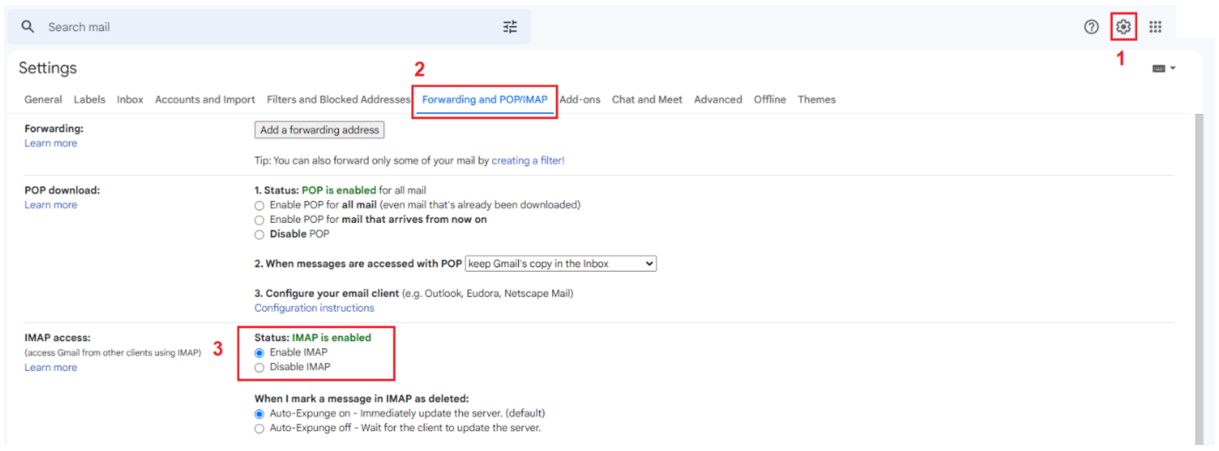
Fig. 14. The "Download Options" tab

Using this function, you can load e.g. "headers" of e-mails, select from them the necessary ones and then load the whole information about them with the help of the "E-mails Actions" module that will be described in the next chapter.

Enabling IMAP and POP3 protocols for Gmail (example)

IMAP

To enable the IMAP protocol in GMail mail service, go to mail settings (1), then select "Forwarding and POP/IMAP" (2) - "IMAP access" and then in "State" enable IMAP (3):


Search mail
1
2
3
Settings
General Labels Inbox Accounts and Import Filters and Blocked Addresses Forwarding and POP/IMAP Add-ons Chat and Meet Advanced Offline Themes

Forwarding: Learn more

Add a forwarding address

Tip: You can also forward only some of your mail by creating a filter!

POP download: Learn more

1. Status: **POP is enabled** for all mail
- Enable POP for **all mail** (even mail that's already been downloaded)
- Enable POP for **mail that arrives from now on**
- **Disable** POP

2. When messages are accessed with POP keep Gmail's copy in the Inbox

3. Configure your email client (e.g. Outlook, Eudora, Netscape Mail)
Configuration instructions

IMAP access: (access Gmail from other clients using IMAP) Learn more

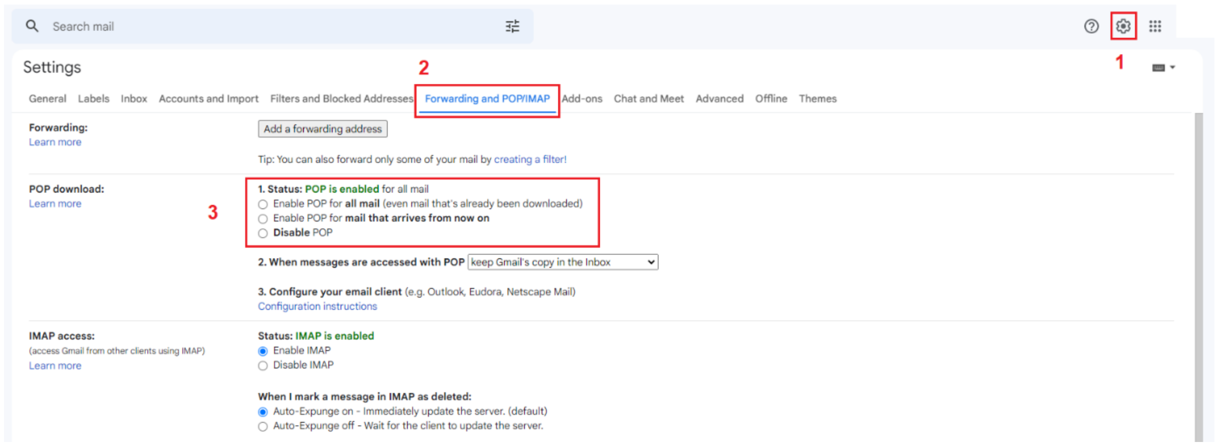
Status: **IMAP is enabled**
- [✓ Enable IMAP]
- Disable IMAP

When I mark a message in IMAP as deleted:
- [✓ Auto-Expunge on - Immediately update the server. (default)]
- Auto-Expunge off - Wait for the client to update the server.

Figure 19: Enabling IMAP in Gmail

POP3

The same way you enable the POP3 protocol: Settings (1) - "Forwarding and POP/IMAP" (2) - "POP Access" - "Enable POP for all e-mails" (3):


Search mail
1
2
3
Settings
General Labels Inbox Accounts and Import Filters and Blocked Addresses Forwarding and POP/IMAP Add-ons Chat and Meet Advanced Offline Themes

Forwarding: Learn more

Add a forwarding address

Tip: You can also forward only some of your mail by creating a filter!

POP download: Learn more

1. Status: **POP is enabled** for all mail
- Enable POP for **all mail** (even mail that's already been downloaded)
- Enable POP for **mail that arrives from now on**
- **Disable** POP

2. When messages are accessed with POP keep Gmail's copy in the Inbox

3. Configure your email client (e.g. Outlook, Eudora, Netscape Mail)
Configuration instructions

IMAP access: (access Gmail from other clients using IMAP) Learn more

Status: **IMAP is enabled**
- [✓ Enable IMAP]
- Disable IMAP

When I mark a message in IMAP as deleted:
- [✓ Auto-Expunge on - Immediately update the server. (default)]
- Auto-Expunge off - Wait for the client to update the server.

Figure 20: Enabling the POP3 protocol in Gmail

EMAIL ACTIONS module

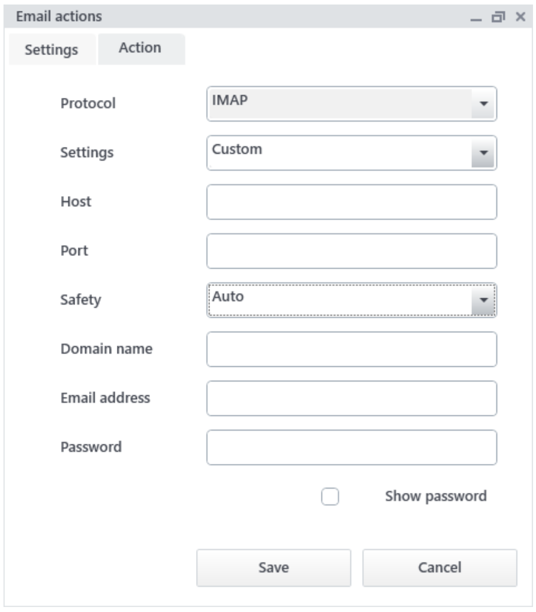
Using EMAIL ACTIONS module, you can transfer already read messages to other folders in the mail, delete messages from the server and download additional information. To use the module, it is necessary to get e-mails using the "Read e-mails" module beforehand.

Module interface

The interface of the module consists of two tabs: "Settings" and "Action".

Settings tab

The "Settings" tab does not differ from the one in the READ EMAILS module, but to this moment only one protocol is available: IMAP. If you choose the necessary mail service in the "Settings" field, the "Host", "Port" and "Safety" fields will be filled out automatically.

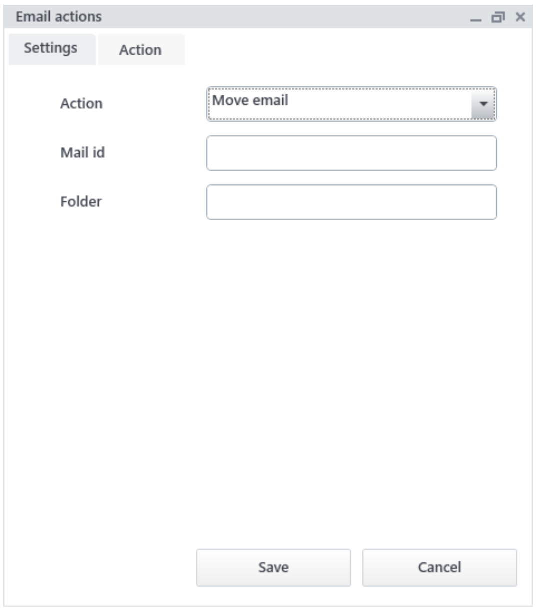


Protocol	IMAP
Settings	Custom
Host	
Port	
Safety	Auto
Domain name	
Email address	
Password	
	☐ Show password

Fig. 21. Settings tab of the "E Mail actions" module

Action tab

You can set up the actions performed on the message on this tab.



Action	Move email
Mail id	
Folder	

Fig. 22. Action tab

The "Action" field allows you to choose one of these available actions:

- Move e-mail
- Delete e-mail
- Load all information from e-mail
- Find folder
- Download message

The "Id of e-mail" field contains the id of the e-mail, over which the action is performed. It can be received using the "Read e-mails" module. Several identifiers separated by commas, or a variable-list containing them can be passed.

The "Folder" field is available only for the "Move e-mail" action. You should enter the name of the folder in the mail, to which you want to move the mail.

The "Variable" field is available only for some actions, where you should enter the name of the variable that will be the result of the action.

OTHER MODULES

CALCULATION module

The "Calculation" module is a linking module or a module for pre-and post-processing of data from other modules. This module is needed to create, calculate and change the values of variables. The module allows you to perform calculations and process information using the JavaScript programming language.

Module interface

The module window is very simple and consists only of a field for input and the buttons "Save" and "Cancel".

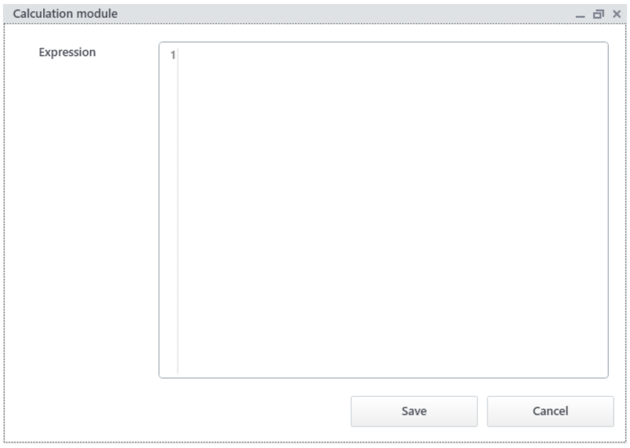


Fig. 1. Module window

"Expression" field. You can enter the code in the JavaScript programming language in this field. It is possible to use local module variables (with their declaration through let, var or const, but they will not be visible in other modules of the robot script), or studio variables that must start with "v." characters.

The module completely obeys the rules of JavaScript language, i.e. all the standard JS functions are supported, the lines are separated by typing ";" or line jumping (the Enter key).

An example of working with module

Suppose you want to get the last day of the previous month. You can do it by entering the following code:

let date = new Date();

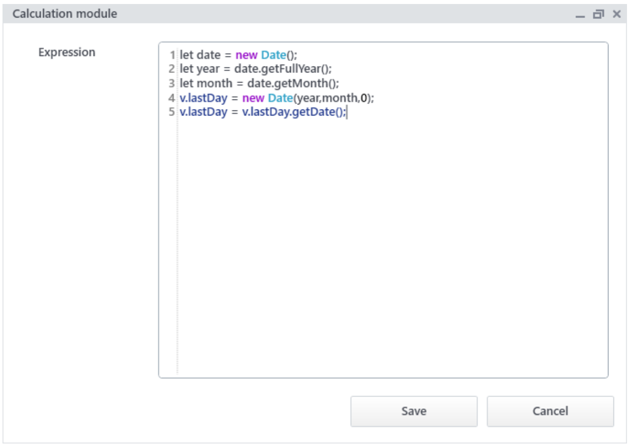
let year = date.getFullYear();

let month = date.getMonth();

v.lastDay = new Date(year,month,0);

v.lastDay = v.lastDay.getDate();

Listing 1. Getting the last day of the previous month



```
1 let date = new Date();
2 let year = date.getFullYear();
3 let month = date.getMonth();
4 v.lastDay = new Date(year,month,0);
5 v.lastDay = v.lastDay.getDate();
```

Fig. 2. Example of module filling

What does the code above do? Here's what - we get today's date, for example, today is March 25th, from this date we take information about current year and month, then we create new date instance, setting current year and month as year and day as zero, as a result JS language will create date with last day of previous month. In more detail:

Line 1 - creating a new date instance without passing parameters, which will create today's date (date = 25.03.2020);

Line 2 - using the getFullYear method, which returns the full year of the date to which it is applied (year = 2020);

Line 3 - using the getMonth method, which returns the month of the date to which it is applied (month = 2, since months in JS are counted from 0);

Line 4 - creating a new date instance, with current year, current month and 0 as the day (v.lastDay = 29.02.2020) as parameters;

Line 5 - getting the number of the last day of the previous month (v.lastDay = 29).

As an introduction to the JavaScript programming language we recommend to study the Internet resource "Modern JavaScript Tutorial", available at https://learn.javascript.ru/.

PAUSE module

The PAUSE module is designed to create a delay between modules execution.

The whole setting of the module is to fill in the "Pause" field - the number of milliseconds to wait. It is allowed to use a variable.

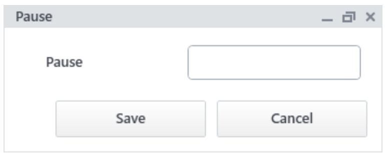


Fig. 16. window of the "Pause" module.

LOG module

LOG module is used to output some messages or variable values into the "Debug output" while the robot is working.

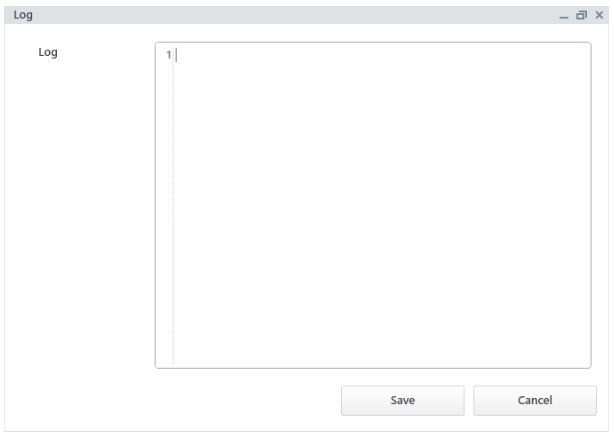


Fig. 18. Window of the "Log" module

INTERFACE module

This module is designed to create dialog boxes for communicating with the robot user. These can be windows allowing you to attach a file, enter some data - dates, names, email addresses, etc., with which your robot will subsequently work, or, conversely, windows displaying the results of your work.

Creating a window consists of adding to it controls - interface elements such as labels, text fields, buttons, drop-down lists.

Main window

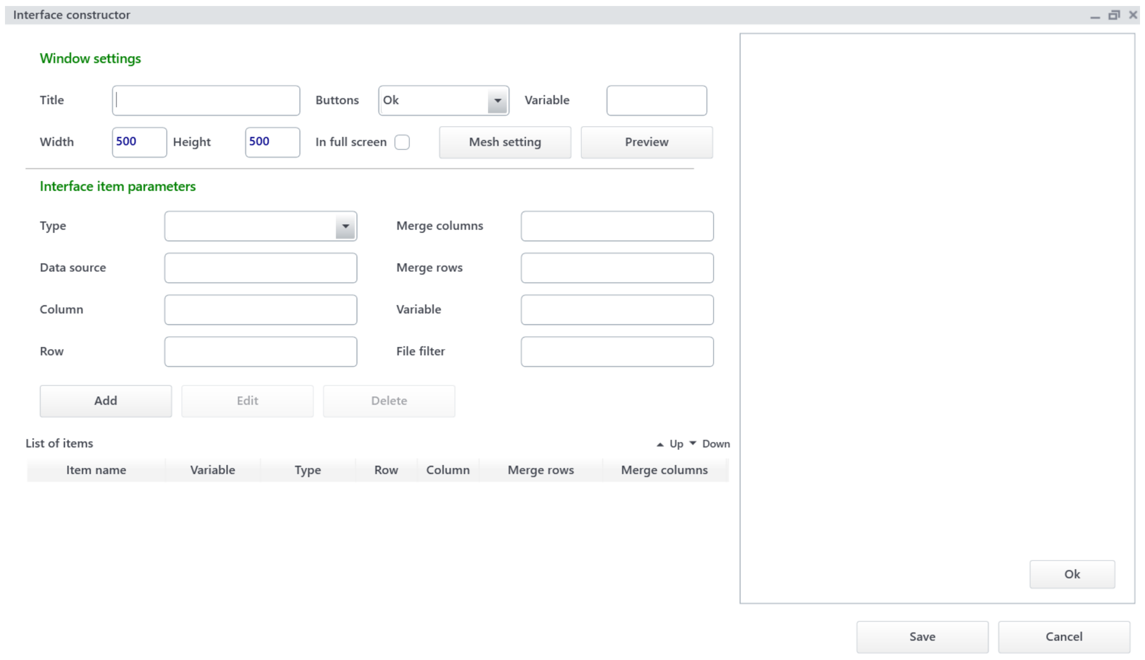
A module window consists of several parts:

Window settings - general settings of the dialog box;

Interface item settings - settings of the element to be added;

Elements list - a table that contains all added > elements;

Window with the current > dialog window preview (right part of the window).



Buttons: Ok
Width: 500
Height: 500

Fig. 1. The main window of the module

You should start working with the dialog window with the general settings, namely the grid settings, which can be opened using the "Grid settings" button.

Window settings

The window settings include the name of the window - "Title", its width and height, and the button and grid settings. If the checkbox "Full screen" is checked, the fields "Height" and "Width" will not be taken into account.

The field "Buttons" allows you to choose a set of buttons that will be placed on the created window. The list of available sets:

- "Ok";
- "Ok" and "Cancel";
- "Yes" and "No";
- "Yes" "No" and "Cancel".

After this field there is a field "Variable", in which you enter the value of the variable, in which the result of selecting the button will be entered:

- When the "Yes" button is clicked, the value "yes" will be written;
- When you press the "No" button, the value "no" will be written;
- When you press the "Cancel" button, the value "cancel" will be entered;
- When you press the "OK" button, the value "ok" will be saved.
- After pressing the "Grid settings" button, the window of grid rows and columns settings will appear.

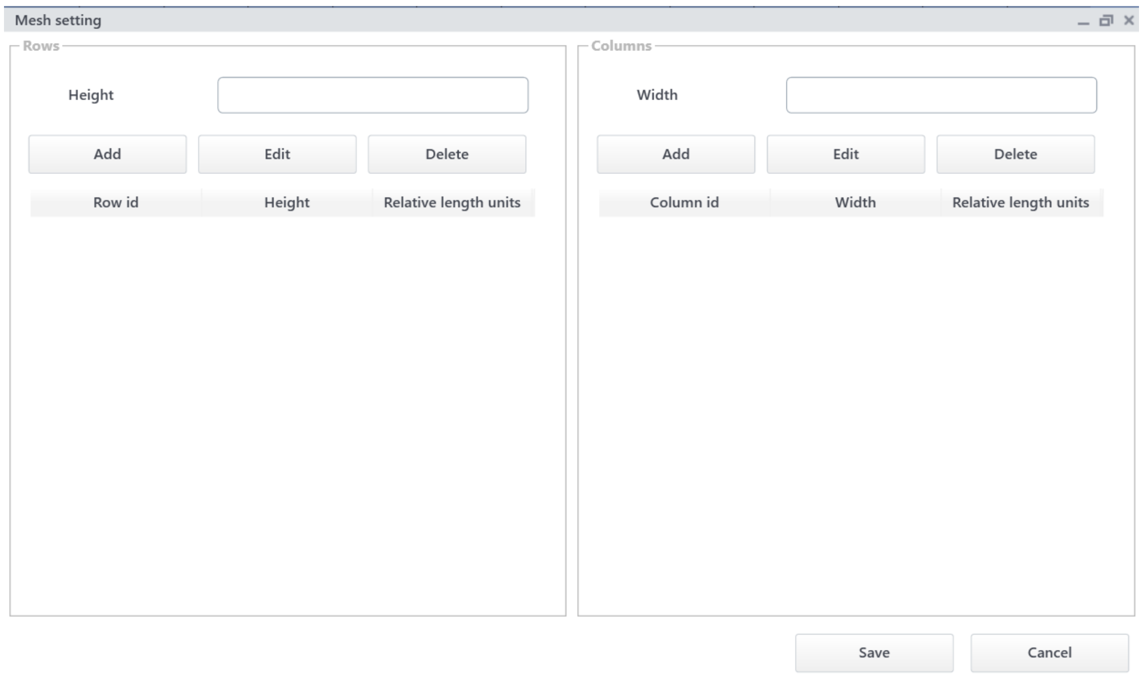


Fig. 2. Grid settings

The grid is a conventional partitioning of interface elements, a kind of table in which controllers will be placed. For example, if you want to create a dialog box where the user would enter two parameters - an end date and a start date for downloading mail - you can configure the dialog box to enter these dates in several ways:

The first way

- in two lines, 4 interface elements - on the first line the signature and the field to enter the start date, on the second line the signature and the field to enter the end date

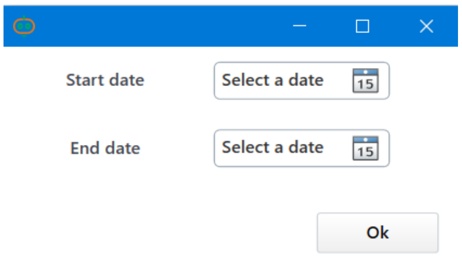


Fig. 3. Example of window #1

For this grid setup, you should set two columns - the first one for caption, the second one for input fields, and two lines - the first one for start date, the second one for end date.

Second way.

- one line, 3 interface elements - signature, field for entering the first date, field for entering the second date.

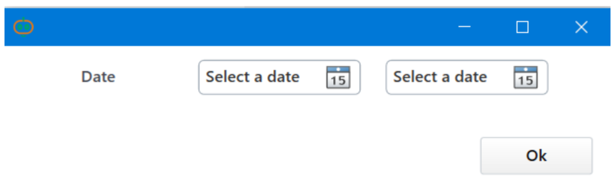


Fig. 4. Example of window #2.

In this case three columns and one line are used.

Third way

- one line, 4 interface elements - two fields to enter and a signature for each of them.

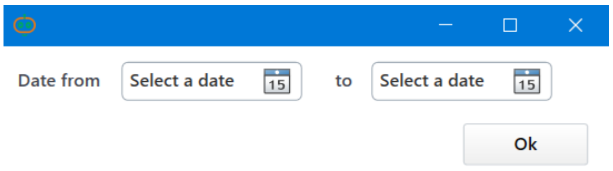
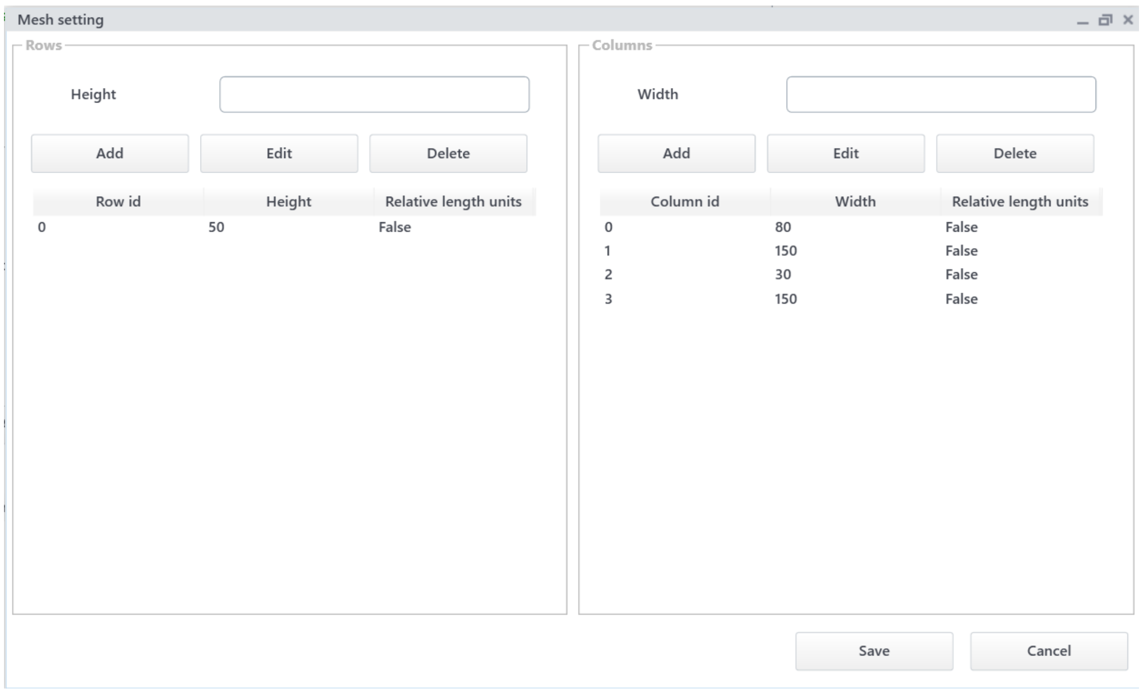


Fig. 5. Example of window #3.

Here we use 4 columns and 1 row.

You can set a different width for each column and height for each line:

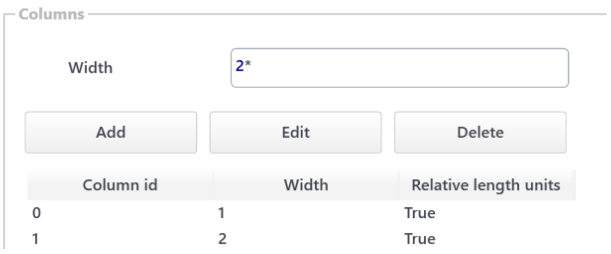


Row id	Height	Relative length units
0	50	False

Column id	Width	Relative length units
0	80	False
1	150	False
2	30	False
3	150	False

Fig. 6. The example of grid settings

The height and width are entered in pixels, but you can use relative window sizes, for example, if one column should be twice as big as the other and there are only two of them, you can enter "1*" for the first column and "2*" for the second in the "Width" field:



Width: 2*

Column id	Width	Relative length units
0	1	True
1	2	True

Figure 7. Example of using relative dimensions

The result of markup:

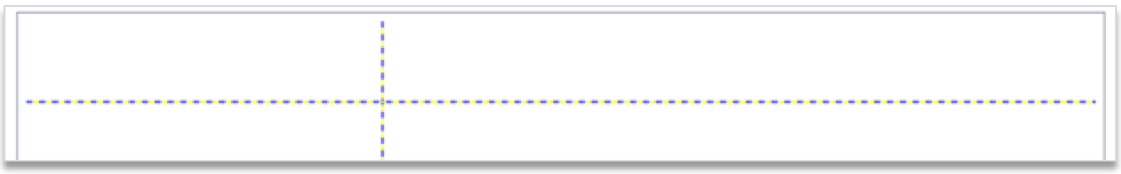


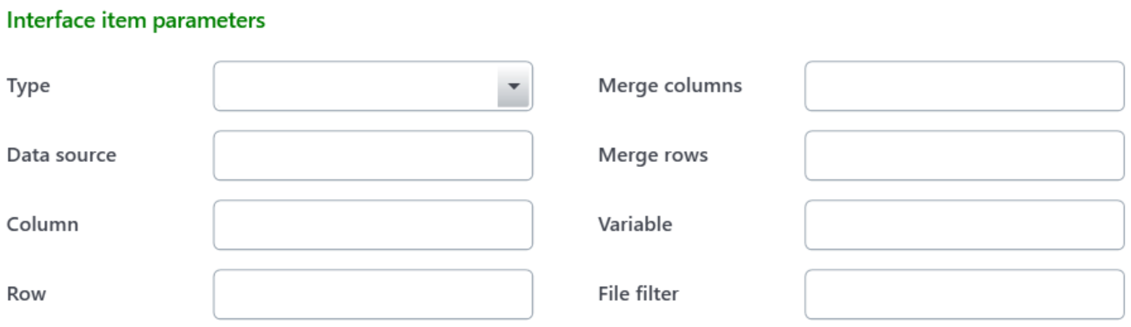
Fig. 8. Example of the layout

The numbering of rows and columns starts from 0. With the "Delete" button you can delete an unnecessary row or column, with the "Edit" button you can change the height or width.

Once the grid is set, it will be displayed with a dotted line in the preview window on the right side of the window, and then you can proceed to add controls.

Adding a new element to interface

To add a new element to the window, you should fill in the fields shown in Figure 9:



Interface item parameters

Type		Merge columns	
Data source		Merge rows	
Column		Variable	
Row		File filter	

Figure 9. Fields for adding a new element

The mandatory fields are all the fields, except the "Merge columns/rows". Field "Filter by file" is available only for the element "Select file".

Type field

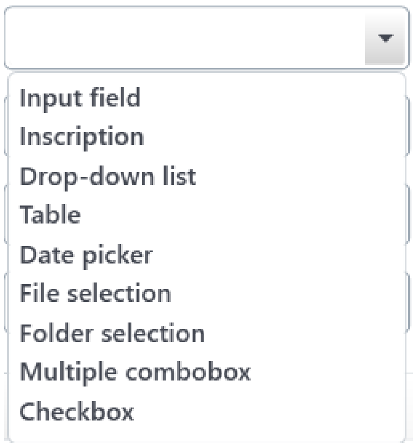


Fig. 10. Drop-down list with element type selection

First you choose a type of interface element to be added from the offered ones:

Input field - the field, where you can input any string;

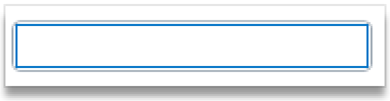


Fig. 11. "Entry field" element.

Inscription - static line of text, which cannot be edited by the user;

Fig. 12. "Inscription" element.

Drop-down list - a field with an arrow on the side, when you click it, the list of predefined values opens;

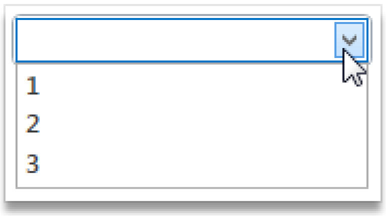
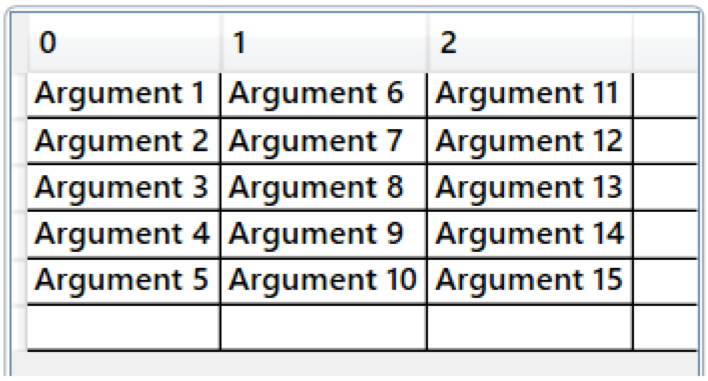


Fig. 13. "Drop-down list" element.

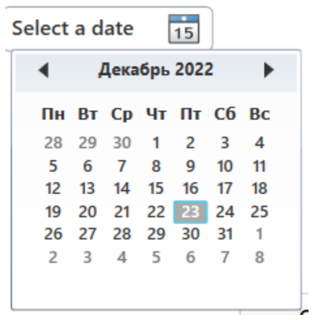
Table - displaying as a table of a pre-created variable. You can edit the table fields, but not create new columns in the table;



0	1	2
Argument 1	Argument 6	Argument 11
Argument 2	Argument 7	Argument 12
Argument 3	Argument 8	Argument 13
Argument 4	Argument 9	Argument 14
Argument 5	Argument 10	Argument 15

Fig. 14. "Table" element.

Select date - a field with a calendar icon, at pressing on which the user will be prompted to select a certain date. You can enter the date value manually. If you enter other lines, the field will automatically select the most suitable date to the entered data. The format is DD;



Пн	Вт	Ср	Чт	Пт	Сб	Вс
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Fig. 15. "Date selection" item.

Select file - a field with an icon of three dots, at pressing on which the user will be prompted to select the file. The path to the selected file will be recorded in this field.

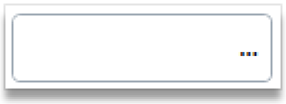
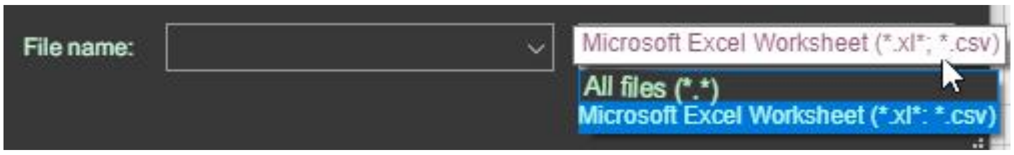


Fig. 16. "File selection" element.

To filter the files displayed to the user, you can use the field "Filter by file", in which the string of the form is entered: "File type name (available extensions)| available extensions". You can specify several filtering settings; the separator for that is the "|" symbol. Example:

In this case, two filter settings are displayed:

1. All files (*.*), which displays files with any extension;
2. Microsoft Excel Worksheet (*.xl*; *.csv), which displays files with extensions .csv and beginning with .xl (i.e. .xlsx, xls and others).


File name:
Microsoft Excel Worksheet (*.xl*; *.csv)
All files (*.*)
Microsoft Excel Worksheet (*.xl*; *.csv)

- Select folder - a field with an icon in the form of three dots, at pressing on which the user will be offered to select a folder. The path to the selected folder is written in this field. It looks the same as "Select File";
- Drop-down list with multiple choices - a field with an arrow on the side, by clicking on which a list of predefined values opens. The user can select multiple values from this list. All selected values are displayed at the beginning of the list. A list of all selected items will be written to the result variable (i.e. the result variable type is a list);

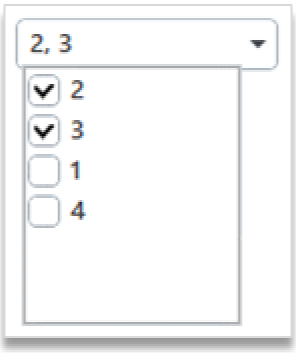


Fig. 17. the "Drop-down list with multiple choice" element

A checkbox is a field, which can have only two values - true and false, that is, a checkbox is checked or not.

Fig. 18. "Checkbox" element

Data Source field

In this field, you enter the name of the variable or string, based on which the control will be filled. A little more about the field when selecting the various elements of the interface:

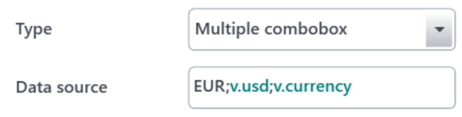
For the "Inscription" element type, the value to be displayed in the displayed inscription is entered there;

For "Entry field", "Date selection", "File selection" and "File selection", you enter the value that will be immediately indicated in the field, if necessary. If the field is to be blank, you do not need to fill in the "Data Source";

The "Checkbox" field supports values such as "true"/"false" and 1/0. If true or 1 is submitted in the data source, the checkbox will be checked;

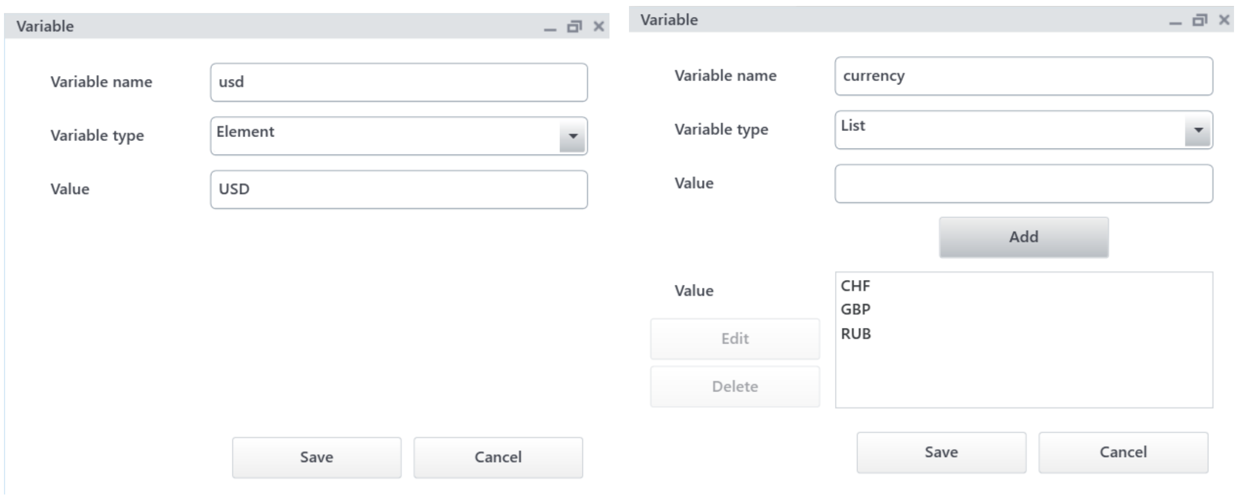
For "Table" you need to supply a populated table variable to display it;

For the interface elements "Drop-down list" and "Drop-down list with multiple choice" you should pass the set of values that the user will be asked to select. This is done with the ";" symbol. These can be strings, element variables, variable lists or table variables consisting of a single column. For example, the following value is entered, in the following examples of filling variables:



Type	Multiple combobox
Data source	EUR;v.usd;v.currency

Fig. 19. Example of filling in the "Data Source" field



Variable

Variable name	usd
Variable type	Element
Value	USD

Variable

Variable name	currency
Variable type	List
Value	
Value	CHF GBP RUB

Fig. 20. examples of filling in the variables type

The result will look like the following:

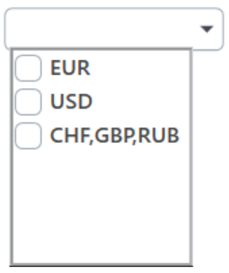


Fig. 21. Example of resulting data

Column and Row fields

You should enter in these fields the column and row numbers of the grid (table) where the control to be added should be located. For example, in the example with dates (Fig. 9.30.2) the inscription control - "Start Date" is located in column 0 and row 0, and the inscription "End Date" - in column 0 and row 1.

Merge Columns and Merge Rows fields

In some cases, you need to output fields not in a specific cell of the grid, but in a union of several cells. For example - you need to prompt the user to select an Excel file to write results to and enter a start and end date on which to collect some data. I would like to combine this into two lines, but you need at least 2 fields to enter the date, and the field for selecting the file is one.

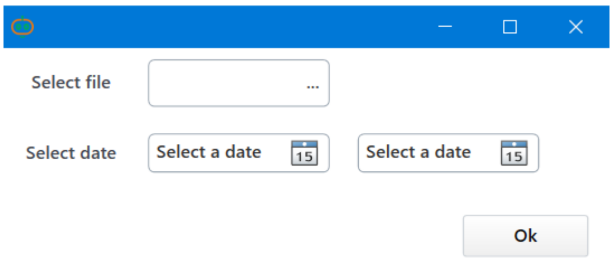
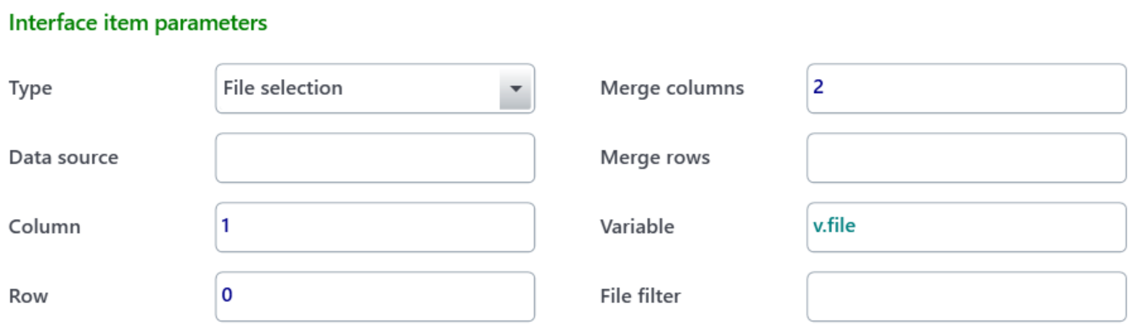


Fig. 22. Example of a window without combining columns

To stretch the file selection field by the length of the date entry fields, we need to place it not just in 1 column and 0 line, but combine two columns for it so that it occupies both at once:



Interface item parameters

Type	File selection	Merge columns	2
Data source		Merge rows	
Column	1	Variable	v.file
Row	0	File filter	

Fig. 23. Example of filling the interface element fields to customize the merging of columns

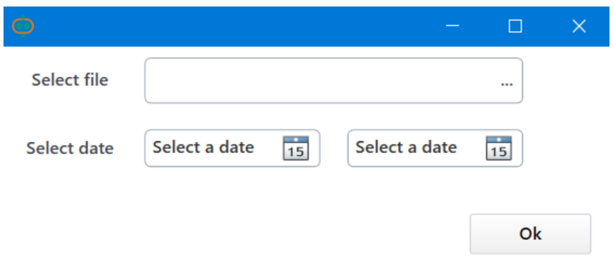


Fig. 24. The example of the window with joined columns in the first line of the grid

That is when you fill the "Merge Columns" field the columns starting from the one specified in the "Column" field will be merged horizontally into one.

Similarly with the "Merge rows" field - the specified number of rows, starting from the one specified in the "Row" field, will be merged into one:

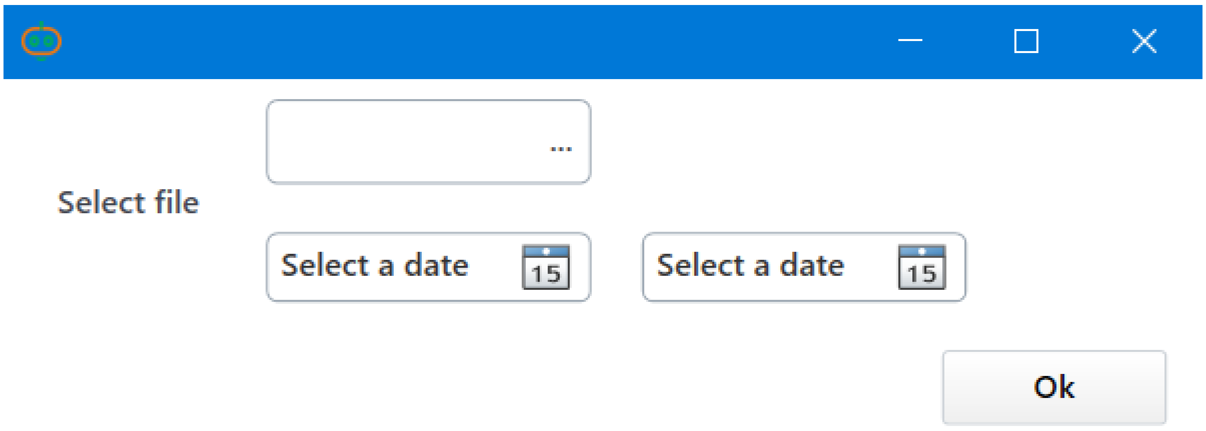
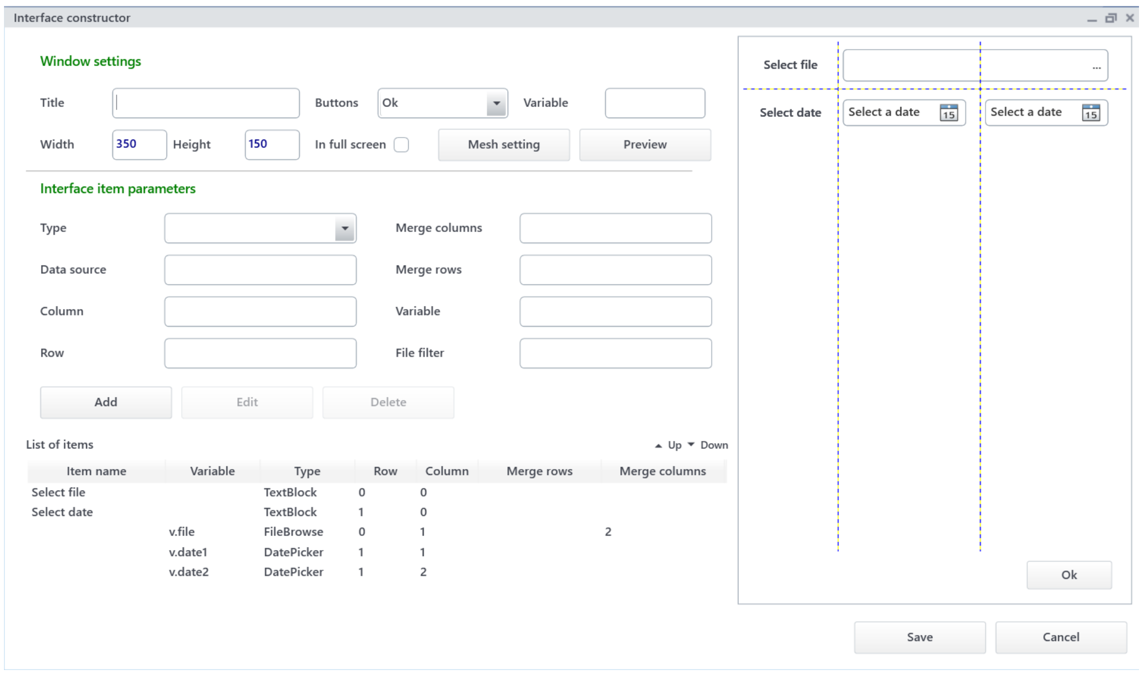


Fig. 25. Example of the window with merged rows in the first grid column

Variable field

This field is where you enter the name of the variable where the value entered (for the drop-down list, the selected value) by the user will be written. It is a mandatory field for all types of controls, except for "Inscriptions".

After filling in all the fields you should click on the "Add" button. After clicking on it the element will be immediately displayed in the preview window on the right side of the window.



Interface constructor

Window settings

Title:
Buttons: Ok
Variable:
Width: 350
Height: 150
In full screen: ☐

Interface item parameters

Type, Data source, Column, Row, Merge columns, Merge rows, Variable, File filter

List of items

Item name	Variable	Type	Row	Column	Merge rows	Merge columns
Select file		TextBlock	0	0		
Select date		TextBlock	1	0		
	v.file	FileBrowse	0	1		2
	v.date1	DatePicker	1	1		
	v.date2	DatePicker	1	2		

Fig. 26. Example of the filled window of the "Interface" module.

Clicking the "Preview" button will open the dialog box in the form, in which it will be presented to the user.

You can delete or edit the element by selecting it in the "Elements list" area, and then click on the button, located above the list of elements.

After adding all the necessary controls you must click the "Save" button at the right bottom of the window.