



Guide to “Parametrization” plugin

Contents

General information

Configuration setup

Processing order of strategies, rules and expressions

Committing changes

Managing strategies and rules

Prefilters and filters

Step-by-step calculation

Strategies

Group 1. Independent elements

1. Self

2. Families

3. Types

16. Materials

Working with source elements

Group 2. Containment

4. In Host

5. In Group

6. In Room

7. In Space

8. In Mass

9. In Solid

15. Rebar In System

17. In MEP System

21. In Assembly

Group 3. Components

12. Connection SubElements

13. Nested Families

18. Curtain Wall Components

20. Parts

23. Stair Components

Group 4. Rooms

11. Room Boundaries

24. Room Openings

26. Room Finishing

Group 5. Adjacent elements

10. Solid Touch

14. MEP Insulation

28. Rebar Couplers

Group 6. Levels and views

22. ViewSpecific

27. On Level

29. Level Views

Group 7. Arbitrary relationships

19. One To Many

25. Many To Many

Element matching

Element filter

Element grouping

Formula Editor

Parameters in formulas

Formula syntax

Writing to a source element

Fields

Functions

Mathematical functions

Logical functions

Modifying functions

Value retrieval functions

Aggregate functions

Write-back functions

Checking parameter values

Check results

Palette

General information

The “Parametrization” plugin processes element parameter values using strategies, filters and formulas.

Processing is configured at four levels:

- 1. Configuration
combines the settings required for one or more tasks
- 2. Strategy
defines how elements are processed
- 3. Rule
specifies the processing conditions within a strategy
- 4. Formula (expressions)
defines the operations performed on parameters

PARAMETRIZATION

Configuration: ModPlus Examples by strategy ModPlus

Strategies

2 Add strategy

Import Export

☐ Self

1/18 1. Self

☒ In Room

1/1 6. In Room

☒ Families

2/6 2. Families

☐ Types

2/2 3. Types

☐ Materials

4/4 16. Materials

☒ In Mass

2/2 8. In Mass

☐ In Host

2/6 4. In Host

Process elements in Document

☐ Processing elements from linked files

☐ Select linked files

Execute

Check

Settings

Strategy processes elements located in Rooms

Name and description: Name: In Room Description: Copy the room number and name

Data source: File: Sheet:

Pre-filter: Target element Filter by category And Filter by parameters

Pre-filter: Room Filter by parameters

Strategy filter: Target element Filter by category And Filter by parameters

Strategy filter: Room Filter by parameters

Rules

#	On	Room	Target element	Grouping: Target element	Formula	Options
Example 1 – Copy the room number and name						
3	1	☒	Parameters:	Categories: AND Parameters:	4 \${Comments}=@[Number]...	T3 Aa

Work in the plugin is organized into configurations. A configuration combines the strategies, rules, filters and formulas needed for one or more related tasks.

To configure element processing:

1. Create a configuration and give it a clear name that describes its purpose. You can export and import a configuration together with all its strategies.
2. Add one or more strategies. A strategy defines how elements and their relationships are processed. You can name strategies and add descriptions. Each strategy can be exported and imported separately with all its rules.
3. Add rules to the strategies. Depending on the strategy, a rule contains filters and other conditions that determine which elements are processed.
4. Enter the formulas. Each rule can contain one or more expressions that define the values to retrieve or write. Enter each expression on a separate line.
5. Set the processing order. Add other strategies and rules as needed and arrange them in the required order. Strategies, rules and expressions run sequentially, so their order matters.
6. Select the processing scope: selected elements, elements on the current or selected views, or elements in the entire document.
7. Click Execute. When processing is complete, a window displays the configuration results.

The plugin processes strategies, rules and expressions sequentially, in their order in the configuration:

- 1.Processing starts with the first strategy. Its rules are processed in sequence, starting with the first rule.
- 2.Within each rule, expressions are evaluated in sequence. The first expression is applied to the elements, followed by the second, and so on.
- 3.After all rules in the first strategy have been processed, the plugin moves to the next strategy. Processing continues until all enabled strategies have been completed.
- 4.The Parametrization Results window then displays information about the strategies, rules and expressions that were processed.

The order matters when later operations use the results of earlier ones.

PARAMETERIZATION RESULTS

Configuration: Examples by strategy. Total number of strategies: 1. Total number of rules: 1. Total time spent: 00:00:00.7927631

Searching for relationships between element instances: 00:00:00.5658850

Total number of processed elements: 327

Expand all

Collapse all

Export to txt

Export to Excel

6. In Room - In Room

136

125

66

00:00:00.2268781

Rules executed: 1 of 1

Rule 1

136

125

66

00:00:00.2240633

Expression 1. \$[Comments]=@[Number] @

136

125

66

! Parameter "Comments" is not found

66

! Value setting is skipped, because the parameter "Comments" has the same value

125

✓ Processed

136

Processed elements

Select all

Isolate all

←

1 / 2

→

972953

970378

567916

568023

931195

931201

931280

970834

971595

975080

975081

975082

976237

976238

976239

976240

415681

626060

700296

857192

857195

v.5.0 ModPlus™

The plugin settings specify when parameter changes are committed to the document. This determines whether subsequent expressions and rules can use the updated values.

1. When setting a parameter value

A change is committed immediately after a value is written to an element parameter. Subsequent expressions can use values updated by earlier expressions.

This option provides the most sequential calculation behavior, but may increase processing time.

2. When processing a rule

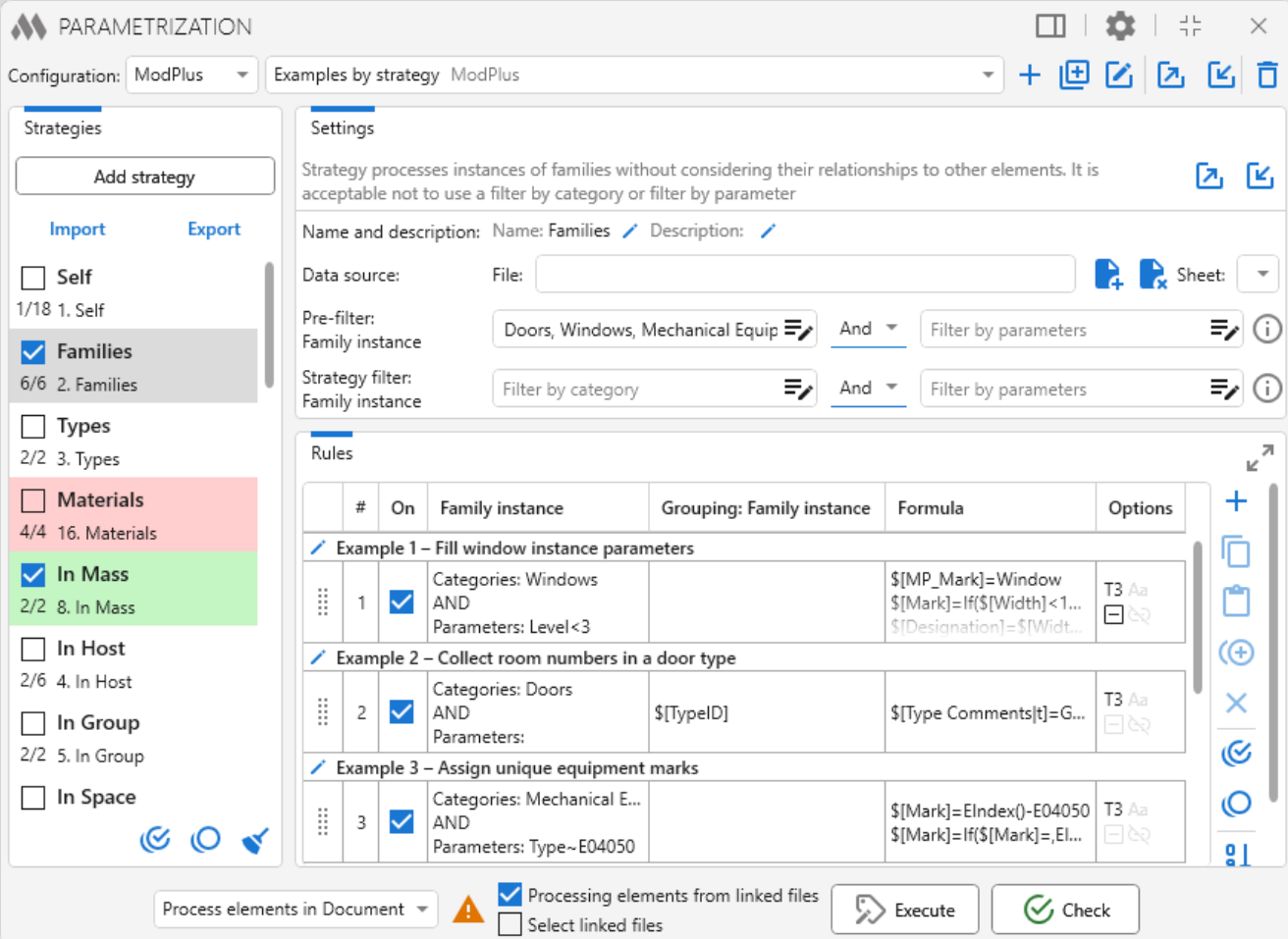
Changes are committed after all expressions in the rule have been evaluated. A value updated by one expression is unavailable to another expression in the same rule.

The updated value is available to subsequent rules in the strategy.

3. When processing a strategy

Changes are committed after all rules in the strategy have been processed. Updated values cannot be reused within the same strategy.

The updated values are available to subsequent strategies.



The checkbox beside each strategy and rule enables or disables it. Disabled strategies and rules are not processed when you click Execute. Move strategies and rules in the list to change their processing order. Rules can also be dragged between strategies.

Strategies can be duplicated or deleted. Rules can be copied to the clipboard and pasted into the same or another strategy, including one in another configuration. Rules can be renumbered to match their current order.

You can change the background color of a strategy and the first cell of a rule to distinguish them visually.

A strategy can use an Excel file as its Data source. Functions such as CV, CVIf and CVOr read values from that file.

Strategy prefilters and element filters limit the set of elements to be processed.

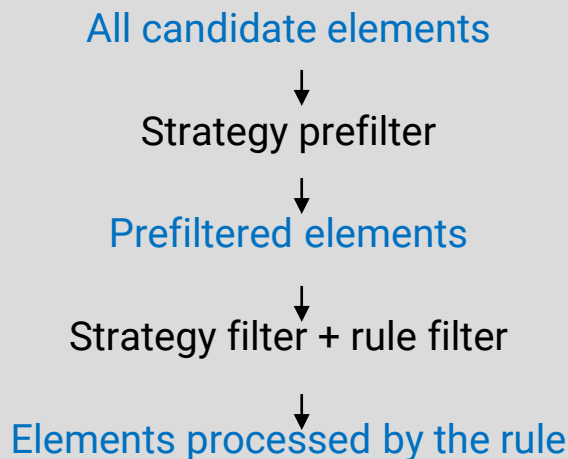
A prefilter operates at strategy level and excludes elements before processing. Depending on the strategy, prefilters may be available for source elements, target elements or both.

The remaining elements are evaluated against the filters defined in the strategy and its rules.

Element filters can be defined for a strategy or individual rules. A strategy filter is used when the corresponding filter is not defined in the rule.

If the strategy and rule define different parts of a filter, those parts are combined. For example, a strategy category filter can be combined with the parameter conditions of a rule.

If a rule defines both category and parameter filters, the corresponding strategy filter is ignored.



Step-by-step calculation

Step-by-step calculation evaluates a formula for one element and shows each stage of the calculation.

Hover over a Formula cell to display the button that opens the Step-by-step Calculation window.

	1	☒	Categories: Walls AND Parameters:	$\$[Comments]=\{ \{ 1+2 \} + Round(Max(12.55, 12.23, 13.41), 0.1) + \{ \$[Length] - 1000 \} \}$	 	T3 Aa  
---	---	-------------------------------------	---	---	---	--

Pick a target element and, where required by the strategy, a source element. Then click Execute.

The window shows the original expression, parameter substitution, the normalized expression, intermediate calculations and the final result. This helps you locate calculation errors.

 STEP-BY-STEP CALCULATION

Target element: Basic Wall. CL_W1

Pick

Execute

1 --> Original expression:

2 $\$[Comments]=\{ \{ 1+2 \} + Round(Max(12.55, 12.23, 13.41), 0.1) + \{ \$[Length] - 1000 \} \}$

3

4 --> Replacing parameters with unique keys:

5 - Length -> ~T_E_P1~

6

7 --> Normalized formula:

8 $\{ \{ 1+2 \} + Round(Max(12.55, 12.23, 13.41), 0.1) + \{ \sim T_E_P1 \sim - 1000 \} \}$

9

10 --> Step-by-step calculation:

11 $\{ \{ 1+2 \} + Round(Max(12.55, 12.23, 13.41), 0.1) + \{ 21026.016024 - 1000 \} \}$

12 $\{ \{ 1+2 \} + Round(Max(12.55, 12.23, 13.41), 0.1) + 20026.016024 \}$

13 $\{ \{ 1+2 \} + Round(13.41, 0.1) + 20026.016024 \}$

14 $\{ \{ 1+2 \} + 13.4 + 20026.016024 \}$

15 $\{ 3 + 13.4 + 20026.016024 \}$

16 20042.416024

17

18 --> Expression calculation result:

19 20042.416024

Open in Notepad

v.5.0 ModPlus™

Strategies

A strategy defines the element-processing algorithm. The strategies are organized into seven groups:

1. Independent elements – processing without considering relationships to other elements.
2. Containment – processing elements based on their relationship to a source element that contains them.
3. Components – processing component parts and nested elements.
4. Rooms – processing room boundaries, openings and finishes.
5. Adjacent elements – processing elements based on adjacency or connections.
6. Levels and views – processing elements and views based on their associated level or view.
7. Arbitrary relationships – processing relative to a specified source element, or matching two sets of elements by a calculated value.

Identify the elements to process and the relationship to other elements that matters for the task. Select a strategy with the appropriate algorithm.

Group 1. Independent elements

The strategies in this group process elements independently of their relationships to other elements.

1. Self – processing element instances
2. Families – processing loadable family instances
3. Types – processing element types
16. Materials – processing every material in the current document, regardless of selection mode

You can write the following to a target parameter:

- an arbitrary value;
- the current target value as part of an expression;
- a value from another target parameter;
- a value composed from several target parameters;
- an empty value: a text parameter is cleared, while a numeric parameter is set to 0.

1. Self – processing independent elements

The Target element column must contain a category filter, a parameter filter, or both.

Example 1

	#	On	Target element	Grouping: Target	Formula	Options
Example 1 – Fill and clear wall parameters						
	1	☒	Categories: Walls AND Parameters:		<code>\$(Comments)=wall</code> <code>\$(Mark)=\$(Mark)_S</code> <code>\$(MP_Level)=Format(\$(Base Constraint),d2).level</code> <code>\$(MP_Note)=\$(Family)_\$(Width).1</code> <code>\$(MP_Designation)=</code>	T3 Aa 
Formula: <code>\$(Comments)=wall</code> <code>\$(Mark)=\$(Mark)_S</code> <code>\$(MP_Level)=Format(\$(Base Constraint),d2).level</code> <code>\$(MP_Note)=\$(Family)_\$(Width).1</code> <code>\$(MP_Designation)=</code>						

Target element – elements in the “Walls” category.
No parameter filter is defined, so any walls placed in the model can be processed.

The target parameters receive the following values:

- An arbitrary value: “wall” is written to Comments.
- The current value as part of an expression: Mark retains its previous value and appends the suffix “_S”.
- A value from another parameter: “MP_Level” receives the floor number extracted from Base Constraint, formatted as two digits with Format(x,f), followed by “level”. The result is “03 level”.
- A composite value: “MP_Note” combines Family, “_”, Width and the suffix “.1”. The result is “Basic Wall_300.1”.
- The value of “MP_Designation” is cleared.

Example 2 – fill “MP_Position” for rebar.

Example 2 – Fill rebar positions

...	2	☒	Categories: Structural Rebar AND Parameters:		$[MP_Position]=If(Or([MP_dimension...]$	T3 Aa □ ↺
-----	---	-------------------------------------	--	--	---	--------------

Formula:
 $[MP_Position]=If(Or([MP_dimension\ in\ linear\ meters]=Yes,[Bar\ Length]>11700),[Bar\ Diameter]-1m,[Bar\ Diameter]-Round([Bar\ Length]*0.1,!5))$

Target element – elements in the “Structural Rebar” category

If “MP_dimension in linear meters” is checked or Bar Length exceeds 11700 mm, “MP_Position” receives Bar Diameter followed by “-1m”. Otherwise it receives Bar Diameter, a hyphen and Bar Length converted to centimeters and rounded to a multiple of 5 using Round(a1,a2).

If the checkbox is off, Bar Diameter is 12 mm and Bar Length is 1580 mm, the result is 12-160.

Example 3 – determine whether a wall is a pylon.

Example 3 – Identify pylon walls

...	3	☒	Categories: Walls AND Parameters:		$[MP_Note]=If(Or([Length]/[Width]<4,([Unconnected\ Height]-250)/[Length]>4),pylon,wall)$	T3 Aa □ ↺
-----	---	-------------------------------------	---	--	---	--------------

Formula:
 $[MP_Note]=If(Or([Length]/[Width]<4,([Unconnected\ Height]-250)/[Length]>4),pylon,wall)$

Target element – elements in the “Walls” category

Arithmetic calculations in the conditions are enclosed in braces. Parentheses used for arithmetic rather than function arguments are escaped with a backslash (\).

If at least one condition in Or(c1,c2,...,cn) is true, “pylon” is written to “MP_Note”. If none is true, the result is “wall”.

Example 4 – write a level elevation in meters, with three decimal places, to the level name.

Example 4 – Include elevation in level names					
⋮	4	☒	Categories: Levels AND Parameters:		<div>\$[Name]=Level Format(\$[Elevation]/1...</div> <div>T3 Aa</div> <div>□ ↺</div>
Formula: \$[Name]=Level Format (\$[Elevation]/1000,f3)					

Target element – elements in the “Levels” category

Name contains “Level” followed by Elevation divided by 1000 and formatted to three decimal places using Format(x,f).

Example 5 – count ducts with the same length.

	#	On	Target element	Grouping: Target	Formula	Options
Example 5 – Count ducts of equal length						
⋮	5	☒	Categories: Ducts AND Parameters:	<div>\$[Length]</div>	<div>\$[MP_Count]=GCount()</div>	<div>T3 Aa</div> <div>□ ↺</div>
Formula: \$[MP_Count]= GCount ()						
Grouping: Target element: \$[Length]						

Target element – elements in the “Ducts” category

Group the elements by Length.

Ducts of equal length form a group. Each duct receives the group element count in “MP_Count”.

For three ducts 1500 mm long the result is 3; for two ducts 2000 mm long it is 2.

Example 6 – write system totals to pipes and number the pipes within each system.

Example 6 – Calculate pipe system totals

⋮	6	☒	Categories: Pipes AND Parameters:	[\$System Name] ascending [\$Mark]	[\$MP_Pipe count]=GCount() [\$MP_Total length]=GSum(\$[Length]) [\$MP_Average length]=GAvg(\$[Length])	T3 Aa ☐
---	---	-------------------------------------	---	--	--	-----------------------------------

Formula:
\$[MP_Pipe count]=GCount()
\$[MP_Total length]=GSum(\$[Length])
\$[MP_Average length]=GAvg(\$[Length])
\$[MP_Minimum length]=GMin(\$[Length])
\$[MP_Maximum length]=GMax(\$[Length])
\$[MP_Mark list]=GJoin(\$[Mark],\,,d)
\$[MP_System index]=GIndex()

Grouping: Target element: [\$System Name]
ascending
[\$Mark]

Target element – elements in the “Pipes” category

Group the elements by System Name.

Enable sorting within each group by Mark.

The element count, length statistics and list of marks are the same for all pipes in a group. GIndex() numbers each pipe in the sorted group. GJoin() preserves that order and removes duplicates.

Pipes with an empty grouping key are skipped.

Example 7 – determine pipe orientation.

Example 7 – Identify pipe orientation

⋮	7	☒	Categories: Pipes AND Parameters:		[\$MP_Pipe orientation]=If(IsVertical(),...	T3 Aa ☐
---	---	-------------------------------------	---	--	---	-----------------------------------

Formula:
\$[MP_Pipe orientation]=If(IsVertical(),Riser,If(IsHorizontal(3),Branch,Sloped segment))

Target element – elements in the “Pipes” category.

Only straight pipes are processed.

Vertical pipes receive “Riser”. Pipes whose axes are within 3° of the horizontal plane receive “Branch”. The remaining pipes receive “Sloped segment”.

Example 8 – check whether a mark is assigned and follows the required format.

Example 8 – Check equipment mark format

⋮	8	☒	Categories: Mechanical Equipment AND Parameters:		<code>\$(MP_Check result)=If(HasValue(\$[Mar...</code>	T3 Aa □ ↺
---	---	-------------------------------------	--	--	--	--------------

Formula:
`$(MP_Check result)=If(HasValue($[Mark]),If(RgxMatch($[Mark],"^K[0-9]+$"),Valid,Invalid),Value not assigned)`

Target element – elements in the “Mechanical Equipment” category

A mark must consist of “K” followed by one or more digits.

“K12” matches the pattern; “K12A” does not. The ^ and \$ anchors test the entire value.

HasValue() checks whether a parameter has ever been assigned a value. A parameter assigned an empty string passes this check but does not match the pattern.

Enable Case sensitive to check for an uppercase “K”.

Example 9 – retrieve information from a list of systems.

Example 9 – Read a list of systems

⋮	9	☒	Categories: Mechanical Equipment AND Parameters:		<code>\$(MP_First system)=Item(\$(MP_Syste...</code> <code>\$(MP_Found system)=FindItem(\$(MP_...</code> <code>\$(MP_System item index)=ItemIndex(...</code>	T3 Aa □ ↺
---	---	-------------------------------------	--	--	--	--------------

Formula:
`$(MP_First system)=Item($(MP_Systems),\,,0)`
`$(MP_Found system)=FindItem($(MP_Systems),\,,PV1)`
`$(MP_System item index)=ItemIndex($(MP_Systems),\,,PV1)`

Target element – elements in the “Mechanical Equipment” category

Item() splits the string at commas and returns item 0, “EX3.1”, to “MP_First system”.

FindItem() returns the first item containing “PV1”. Both “PV1(s)” and “PV1(e)” match; the first, “PV1(s)”, is written to “MP_Found system”.

ItemIndex() returns the index of the first match: 0 for “EX3.1”, 1 for “PV1(s)” and 2 for “PV1(e)”. “MP_System item index” receives 1.

If no match is found, ItemIndex() returns -1. All three functions use the current element's list.

Example 10 – write information about the project, current view and document.

Example 10 – Read project, view and document information				
...	10	☒	Categories: Mechanical Equipment AND Parameters:	<div> <div>\$[MP_Project Number]=#[Project Nu...</div> <div>\$[MP_Section code]=Item(GetCurrent...</div> <div>\$[MP_Model file]=DocTitle()</div> </div> <div>T3 Aa</div> <div> </div>

Formula:

```

$[MP_Project Number]=#[Project Number]
$[MP_Section code]=Item(GetCurrentViewParameter(VIEW_NAME),_,0)
$[MP_Model file]=DocTitle()

```

Target element – elements in the “Mechanical Equipment” category

“MP_Project Number” receives Project Number from Project Information, for example “2026-015”.

"MP_Section code" receives "S1", the first part of a view name such as "S1 AR Level 02".

“MP_Model file” receives the document title without its extension.

The rule runs on the current view. GetCurrentViewParameter() is unavailable when processing the entire document.

Example 11 – find an equipment name by its primary or alternative code and construct its mark.

Example 11 – Look up equipment names and build marks					
⋮	11	☒	Categories: Mechanical Equipment AND Parameters:	<div> <div></div> <div> <div></div> <div></div> </div> </div> <div> <div></div> <div></div> </div>	<div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div>

Target element – elements in the “Mechanical Equipment” category

The Excel data source contains the primary code in column A, the alternative code in B and the name in C.

The first row whose column A or B matches the equipment code is used. The value in C is returned.

A row contains primary code “EQ01”, alternative code “ALT01” and name “Pump”. Either code returns “Pump”. With prefix “MEP” and code “EQ01”, the mark is “MEP-EQ01”.

The mark combines the global parameter “MP_Equipment mark prefix” with the equipment code.

Example 12 – retrieve numeric limits from Excel.

Example 12 – Read numeric limits from Excel					
⋮	12	☒	Categories: Mechanical Equipment AND Parameters:	<div>\$[MP_Column minimum]=CCMin(D,2.... \$[MP_Column maximum]=CCMax(D,2.... \$[MP_Row minimum]=CRMin(2,F..H)</div>	T3 Aa □ ↺
Formula: \$[MP_Column minimum]=CCMin(D,2..20) \$[MP_Column maximum]=CCMax(D,2..20) \$[MP_Row minimum]=CRMin(2,F..H) \$[MP_Row maximum]=CRMax(2,F..H)					

Target element – elements in the “Mechanical Equipment” category
Column D, rows 2–20, contains power values. Cells F2:H2 contain a separate set.

CCMin(D,2..20) returns the smallest numeric value in D2:D20;
CCMax(D,2..20) returns the largest. For 2.5, 4, 6, 8 and 10, “MP_Column minimum” and “MP_Column maximum” receive 2.5 and 10.
CRMin(2,F..H) compares cells F2:H2. CRMax(2,F..H) returns the largest value in that range. For F2 = 4, G2 = 6 and H2 = 8, “MP_Row minimum” and “MP_Row maximum” receive 4 and 8.

Each function returns one number from its range. It neither searches by equipment code nor compares model elements, so all equipment processed by the rule receives the same four results.

Use consistent power units in the spreadsheet and result parameters.
Headers, empty cells and nonnumeric values are ignored.
Check compares calculated formula results with current parameter values.
When the check is complete, Check Results opens, similar to the execution results window.

Example 13 – find rebar with an empty Mark parameter. Check expression: `$(Mark)=`.

13

☒

Categories: Structural Rebar
AND
Parameters:

`$(Mark)=`

T3 Aa

Formula:
`$(Mark)=`

Target element – elements in the “Structural Rebar” category.
The check produces the following results window:

CHECK RESULTS

Configuration: Examples by strategy. Total number of strategies: 1. Total number of rules: 1. Total time spent: 00:00:00.2541707
Searching for relationships between element instances: 00:00:00.0712350
Total number of processed elements: 346

Expand all

Collapse all

Export to txt

Export to Excel

1. Self - Self

80 0 266 00:00:00.1829357

Rules executed: 1 of 1

Rule 13

80 0 266 00:00:00.1715246

Expression 1. `$(Mark)=`

80 0 266

✓

The value "" of parameter "Mark" matches the calculated value of the expression

80

!

The value "29" of parameter "Mark" does not match the calculated value "" of the expression

1

!

The value "30" of parameter "Mark" does not match the calculated value "" of the expression

1

!

The value "31" of parameter "Mark" does not match the calculated value "" of the expression

1

!

The value "32" of parameter "Mark" does not match the calculated value "" of the expression

1

!

The value "33" of parameter "Mark" does not match the calculated value "" of the expression

1

Processed elements

Select all

Isolate all

←

1 / 2

→

[1316567](#) [1316568](#) [1316569](#)

[1316570](#) [1316696](#) [1316697](#)

[1316698](#) [1316699](#) [1316774](#)

[1316775](#) [1316776](#) [1316777](#)

[1316890](#) [1316891](#) [1316892](#)

[1316893](#) [1316960](#) [1316961](#)

[1316962](#) [1316963](#) [1317123](#)

[1317124](#) [1317125](#) [1317126](#)

[1348258](#) [1348259](#) [1348260](#)

[1348261](#) [1348653](#) [1348654](#)

[1348655](#) [1348656](#) [1348674](#)

Eight elements have an empty Mark.

v.5.0 ModPlus™

Example 14 – check whether “MP_Position” is correctly assigned to rebar.

Example 14 – Check rebar positions					
	14		Categories: Structural Rebar AND Parameters:	\$[MP_Position]=If(Or(\$[MP_dimension...]	T3 Aa

Formula:

```
[MP_Position]=If(Or([MP_dimension.in.linear.meters]=Yes,[Bar.Length]>11700),[Bar.Diameter]-1m,[Bar.Diameter]-Round([Bar.Length]*0.1,!5))
```

Target element – elements in the “Structural Rebar” category.

The check produces the following results window:

CHECK RESULTS

Configuration: Examples by strategy. Total number of strategies: 2. Total number of rules: 2. Total time spent: 00:00:00.3282620
Searching for relationships between element instances: 00:00:00.0895562
Total number of processed elements: 346

Expand all Collapse all Export to txt Export to Excel

matches the calculated value of the expression

Icon	Message	Count
!	The value "" of parameter "MP_Position" does not match the calculated value "8-230" of the expression	4
!	The value "" of parameter "MP_Position" does not match the calculated value "8-1m" of the expression	12
!	The value "" of parameter "MP_Position" does not match the calculated value "8-855" of the expression	2
!	The value "" of parameter "MP_Position" does not match the calculated value "8-820" of the expression	2
!	The value "" of parameter "MP_Position" does not match the calculated value "8-775" of the expression	1
!	The value "" of parameter "MP_Position" does not match the calculated value "8-780" of the expression	1
!	The value "" of parameter "MP_Position" does not match the calculated value "8-135" of the expression	3
!	The value "" of parameter "MP_Position" does not match the calculated value "8-990" of the expression	6
!	The value "" of parameter "MP_Position" does not match the calculated value "8-130" of the expression	1
✓	The value "8-1m" of parameter "MP_Position" matches the calculated value of the expression	30

Processed elements

Select all Isolate all

1 / 1

[1372293](#) [1372716](#) [1373610](#)
[1375623](#)

Elements whose “MP_Position” values differ from the calculated results appear as mismatches.

Example 15 – check values against lists.

Example 15 – Check marks and stages against Excel lists

<div><div></div><div></div><div></div></div>	15	<div><div></div><div></div><div></div></div>	Categories: Mechanical Equipment AND Parameters:		<div><div></div><div></div><div></div></div> <div><div>\$[Mark]= CCList(A,2..100) \$[MP_Stage]= CRLList(2,C..E)</div></div>	T3 Aa <div><div></div><div></div></div>
--	----	--	--	--	---	--

Formula:
\$[Mark]=|CCList(A,2..100)
\$[MP_Stage]=|CRLList(2,C..E)

Target element – elements in the “Mechanical Equipment” category

The Excel data source contains marks in column A, rows 2–100, and permitted stages in row 2, columns C–E.

Both expressions are Check only. They do not change parameter values.

If the permitted stages are listed directly in the formula, replace the second expression with:

Example 15 – Check stages with List()

<div><div></div><div></div><div></div></div>	16	<div><div></div><div></div><div></div></div>	Categories: Mechanical Equipment AND Parameters:		<div><div></div><div></div><div></div></div> <div><div>\$[MP_Stage]= List("P;R;WD",;)</div></div>	T3 Aa <div><div></div><div></div></div>
--	----	--	--	--	---	--

Formula:
\$[MP_Stage]=|List("P;R;WD",;)

2. Families – processing loadable family instances

Families is a special case of Self for loadable families. The Family instance column does not require an element filter.

Example 1

	#	On	Family instance	Grouping: Family	Formula	Options
Example 1 – Fill window instance parameters						
	1	☒	Categories: Windows AND Parameters: Level<3		<code>\$(MP_Mark)=Window</code> <code>\$(Mark)=If(\$(Width)<1200,W-1,W-2)</code> <code>\$(Designation)=\$(Width)x\$(Height) mm</code>	T3 Aa
Formula: <code>\$(MP_Mark)=Window</code> <code>\$(Mark)=If(\$(Width)<1200,W-1,W-2)</code> <code>\$(Designation)=\$(Width)x\$(Height) mm</code> <code>\$(Comments)=</code>						

Family instance – elements in the “Windows” category.
The strategy can write the following to a target parameter:

- an arbitrary value:
`$(MP_Mark)=Window`
Result: Window
- a value determined by another target parameter in a formula condition:
`$(Mark)=If($(Width)<1200,W-1,W-2)`
Result: W-1
- a value composed from target parameters:
`$(Designation)=$(Width)x$(Height) mm`
Result: 900x2100 mm
- clear a target parameter value:
`$(Comments)=`

For details on writing expressions, see the [Formula Editor](#) section.

Example 2 – write room numbers to a door type parameter.

Example 2 – Collect room numbers in a door type

<div><div></div><div></div><div></div></div>	2	☒	Categories: Doors AND Parameters:	[\$TypeID]	[\$Type Comments t]=GJoin(\$[FromRo...	T3 Aa <div></div>
--	---	-------------------------------------	---	------------	--	----------------------

Formula:
\$[Type Comments|t]=GJoin(\$[FromRoom|Number],\,,ds)
Grouping: Family instance: \$[TypeID]

Family instance – elements in the “Doors” category.
Collect room numbers from the source side of each door and write them as a comma-separated list to a type parameter.
Group the elements by TypeID.

TypeID groups instances of the same type. FromRoom|Number retrieves the number of the room on the source side of a door. The |t suffix writes to a type parameter.
GJoin() removes duplicate numbers and sorts them. For “101”, “103” and “101”, the result is “101,103”.

If there is no room on the source side of a door, no value is added.
The list includes only door instances selected by the rule. Use Process elements in document to include every instance of the type.

Example 3 – assign a unique Mark to each equipment element.

Example 3 – Assign unique equipment marks

<div><div></div><div></div><div></div></div>	3	☒	Categories: Mechanical Equipment AND Parameters: Type~E04050		[\$Mark]=EIndex()-E04050 [\$Mark]=If(\$[Mark]=,EIndex(86)-E0405...	T3 Aa <div></div>
--	---	-------------------------------------	--	--	---	----------------------

Formula:
\$[Mark]=EIndex()-E04050
\$[Mark]=If(\$[Mark]=,EIndex(86)-E04050,)

Family instance – elements in the “Mechanical Equipment” category.
Type must contain “E04050”.
Mark receives a sequential number from EIndex(i), followed by “E04050”.
The selection mode determines whether numbering applies to a view or the model.
Elements are processed in their creation order.

To number new instances, supply an offset to EIndex(i). To start at 87-E04050:

```
$[Mark]=If($[Mark]=,EIndex(86)-E04050,)
```

If Mark is empty, numbering starts at 87-E04050. Elements with a filled Mark are skipped.

Example 4 – find mirrored elements.

4

Categories:
AND
Parameters:

\$[Comments]=If(TMirrored(),\$[ID],)

T3 Aa

Formula:
\$[Comments]=If(TMirrored(),\$[ID],)

Family instance – elements in all categories.
No filter is defined, so all placed loadable family instances are processed.
TMirrored() writes the element ID to Comments for mirrored elements.
Otherwise nothing is written because Write empty result is disabled.

3. Types – processing element types

This strategy writes only to type parameters.

Unlike Self, Types can process every type in the document, including types with no placed instances. Select Process elements in document.

To process the types of selected elements or elements on a view, select Process selected elements or Process elements on current view.

The Type column must contain a category filter, a parameter filter using type parameters, or both.

Example 1

	#	On	Type	Grouping: Type	Formula	Options
Example 1 – Set URL for MP types						
	1	☒	Categories: AND Parameters: Type Name~MP		\${URL}=MP	T3 Aa □ □
Formula: \${URL}=MP						

Type – elements in all categories.
Type Name must contain “MP”. “MP” is written to URL.

Example 2

Example 2 – Fill window type name, designation and material						
	2	☒	Categories: Windows AND Parameters:		\${MP_Name}=\${Type Name} \${MP_Designation}=\${Rough Height}x\${Rough Width} \${Glass}=MP_Glass_Transparent Blue	T3 Aa □ □
Formula: \${MP_Name}=\${Type Name} \${MP_Designation}=\${Rough Height}x\${Rough Width} \${Glass}=MP_Glass_Transparent Blue						

Type – elements in the “Windows” category
“MP_Name” receives Type Name.
“MP_Designation” combines Rough Width, “x” and Rough Height. For example: 1470x1460.
Assign “MP_Glass_Transparent Blue” to “Glass”.

For details on writing expressions, see the [Formula Editor](#) section.

16. Materials – processing every material in the current document, regardless of selection mode

Example 1 – identify materials whose names contain “Concrete”.

	#	On	Material	Grouping: Material	Formula	Options
✎ Example 1 – Identify concrete materials						
⋮	1	☒	Parameters: Name~Concrete		[\$[Comments]=Concrete	T3 Aa □ ↺
Formula: \$[Comments]=Concrete						

Material – materials in the current document.

Name must contain “Concrete”.

“Concrete” is written to the material's Comments.

For details on writing expressions, see the [Formula Editor](#) section.

Working with source elements

This information applies to all strategies with a source element, regardless of their group.

Working with geometry

Before running geometry-based strategies, close schedules and unnecessary views. Use Fine detail level for current-view processing. Geometry is always processed at Fine detail level when the entire document is selected.

Linked files

When writing to a target parameter, the source may be in the current document or a linked file. For example, an In Mass source mass may be in a linked file.

When writing to a source parameter, the targets may be in the current document or linked files. In One To Many, the source may be in the current document and the targets in linked files.

How elements interact

1. Filtering source elements limits the related targets. A target parameter can receive:

- an arbitrary value;
- the current target value as part of an expression;
- a value from another target parameter;
- a value composed from several target parameters;
- an empty value.

2. Copy a source value to a target parameter, including within a composite expression:

`$[target element parameter]=@[source element parameter]`

3. Use source parameter values in conditions:

`$(target element parameter)=If(@[source element parameter]>10,A,B)`

4. Combine conditions and source values:

`$(target element parameter)=If(@[source element parameter
1]>10,@[source element parameter 2],B)`

5. Accumulated write-back. Write-back functions collect target results and write the final value to the source parameter. For example:

`Join(@[source element parameter],)=$(target element parameter]`

6. Direct writing. The expression starts with `@[parameter]=...` . The value is written as each element pair is processed. For example:

`@[Number count]=GCount(t)`

Different values from different pairs overwrite the source parameter. Processing order is undefined, so the final value cannot be predicted.

Repeating the same value with the same expression for the same element is skipped and omitted from the report.

Aggregate functions cannot be combined with write-back functions. This restriction does not apply to direct writing.

For details, see Writing to a source element.

4. In Host – processing hosted elements

These include rebar, area and path reinforcement, rebar containers, openings, stacked walls and loadable families, including curtain wall system elements.

The Host column must contain a category filter, a parameter filter, or both.

The category filter lists only categories that can host other elements.

Example 1

	#	On	Host	Target element	Grouping: Tar	Formula	Options
Example 1 – Fill parameters of wall-hosted rebar							
⋮	1	☒	Categories: Walls AND Parameters:	Categories: Structural... AND Parameters:		<code>\$(MP_product mark)=\$[...]</code> <code>\$(Comments)=If(\$[REB...</code>	T3 Aa 
Formula: <code>\$(MP_product mark)=\$[Mark]</code> <code>\$(Comments)=If(\$[REBAR_ELEM_LENGTH]<2000,t1,t2)</code>							

Host – elements in the “Walls” category

Target element – elements in the “Structural Rebar” category

The filters select rebar hosted by walls. Only target parameters are used in these formulas.

Example 2

Example 2 – Copy floor notes to area reinforcement							
⋮	2	☒	Categories: Floors AND Parameters:	Categories: Structural... AND Parameters:		<code>\$(Comments)=@[MP_N...</code> <code>\$(MP_Grouping)=If(@[...</code>	T3 Aa 
Formula: <code>\$(Comments)=@[MP_Note]_rebar</code> <code>\$(MP_Grouping)=If(@[Thickness]<300,F1,F2)</code>							

Host – elements in the “Floors” category

Target element – elements in the “Structural Area Reinforcement” category

The host's “MP_Note” is copied to the target's Comments with the suffix “_rebar”.

The target's “MP_Grouping” depends on the host floor's Thickness, but does not copy that value.

Example 3 – record the material of a door's host wall.

Example 3 – Record the host wall material in a door						
⋮	3	☒	Categories: Walls AND Parameters: Thickness...	Categories: Doors AND Parameters:		<div>\$[MP_Note]=ToLower(If...</div> <div>T3 Aa</div>
Formula: \$[MP_Note]=ToLower(If(StrLen(@[Type])=24,SStr(@[Type],12,8),SStr(@[Type],12,5)))·wall						

Host – elements in the “Walls” category.
Width must be between 200 and 250 mm.

Target element – elements in the “Doors” category

The value of “MP_Note” depends on the wall Type's character count, using StrLen(s). For 24 characters, SStr(s,i,l) extracts eight characters starting at index 12; otherwise it extracts five. ToLower(s) converts the substring to lowercase. The suffix “ wall” follows it.

For “MP_Internal_Concrete_200”, the string has 24 characters. Eight are extracted from index 12 and converted to lowercase. The result is “concrete wall”.

Example 4 – write the count of equally marked floors to embedded parts.

Example 4 – Count equally marked floors for embedded rebars						
⋮	4	☒	Categories: Floors AND Parameters:	Categories: Structural... AND Parameters: Family~E...	@[Mark]	<div>\$[MP_Floor count]=GC...</div> <div>T3 Aa</div>
Formula: \$[MP_Floor count]=GCount(s)						
Grouping: Target element: @[Mark]						

Host – elements in the “Floors” category
Target element – elements in the “Structural Rebar” category.
Embedded parts hosted by the selected floors are processed.
Family must contain “EmbeddedPart”.

Group by the floor's Mark.

Floors with the same mark form a group. GCount(s) counts distinct source floors, regardless of their embedded component count.

For three floors marked “F1”, each embedded part receives 3 in “MP_Floor count”. For two marked “F2”, each receives 2.

Use Process elements in document. Each counted floor must host at least one embedded component matching the rule filter.

An empty floor mark produces an empty key, so its embedded components are skipped.

5. In Group – processing group members

The Group column does not require a parameter filter; Target element does not require category or parameter filters.

The target parameter must be allowed to vary by group instance.

Example 1 – identify the group containing a wall.

	#	On	Group	Target element	Grouping: Tar	Formula	Options
Example 1 – Record the wall group							
	1	☒	Parameters:	Categories: Walls AND Parameters:		[\$[Comments]=in group...	T3 Aa
Formula: [\$[Comments]=in group \"@[Type]\"							

Group – elements in the “Model Groups” category.

No group parameter filter is defined.

Target element – elements in the “Walls” category

Comments receives “in group” followed by the source group's Type in quotation marks. Escape quotation marks with a backslash.

Example 2 – find elements included in groups.

Example 2 – Mark group members							
	2	☒	Parameters:	Categories: AND Parameters:		[\$[Comments]=in group	T3 Aa
Formula: [\$[Comments]=in group							

Group – elements in the “Model Groups” category.

Target element – elements in all categories.

No filters are defined.

“in group” is written to each group member's Comments.

6. In Room – processing elements in rooms

The Room column does not require a parameter filter; Target element does not require category or parameter filters.

Example 1 – record the name and number of the room containing an element.

	#	On	Room	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy the room number and name							
	1	☒	Parameters:	Categories: AND Parameters:		\$(Comments)=@[Number...	T3 Aa  
Formula: \$(Comments)=@[Number] .@[Name]							

Room – elements in the “Rooms” category.

No room parameter filter is defined.

Target element – elements in all categories.

No filter is defined, so all elements inside rooms are processed.

Comments receives the Number and Name of the containing room. If an element overlaps several rooms, only the room containing its centroid is used.

For details on writing expressions, see the Formula Editor section.

7. In Space – processing elements in spaces

The Space column does not require a parameter filter; Target element does not require category or parameter filters.

Example 1 – record the space number for electrical fixtures.

	#	On	Space	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy the space number to electrical fixtures							
	1	☒	Parameters:	Categories: Electrica... AND Parameters:		\$_[Location]=@[_Number]	T3 Aa  
Formula: \$_[Location]=@[_Number]							

Space – elements in the “Spaces” category.

No space parameter filter is defined.

Target element – elements in the “Electrical Fixtures” category

“Location” receives the space Number.

Example 2 – count lighting fixtures and write the count to their space.

Example 2 – Count lighting fixtures in a space							
	2	☒	Parameters: Number>30	Categories: Lighting... AND Parameters:		Sum(@[_MP_Note])=1	T3 Aa  
Formula: Sum(@[_MP_Note])=1							

Space – elements in the “Spaces” category.

Number must be greater than 30.

Target element – elements in the “Lighting Fixtures” category

“MP_Note” receives the lighting fixture count for each qualifying space, using Sum(p).

8. In Mass – processing elements in masses

Masses must have solid geometry. Run this strategy in a 3D view.

The Mass column does not require a parameter filter; Target element does not require category or parameter filters.

Example 1 – assign section numbers using masses that represent the sections.

	#	On	Mass	Target element	Grouping: Tar	Formula	Options
Example 1 – Copy the section number from a mass							
	1	☒	Parameters:	Categories: AND Parameters:		\$(MP_Section Number)...	T3 Aa  
Formula: \$(MP_Section Number)=@[Mark]							

Mass – elements in the “Mass” category.

No mass parameter filter is defined.

Target element – elements in all categories.

No filter is defined, so all elements inside masses are processed.

“MP_Section Number” receives the Mark of the mass containing the element.

Only elements with solid geometry are processed. Elements without solid geometry are excluded. For openings, use In Host to transfer values from their host walls:

In Mass – Example 1 – Copy the section number to hosted doors							
	5	☒	Categories: Walls AND Parameters:	Categories: Doors AND Parameters:		\$(MP_Section Number)...	T3 Aa  
Formula: \$(MP_Section Number)=@[MP_Section Number]							

Host – elements in the “Walls” category.

Target element – elements in the “Doors” category.

9. In Solid – processing elements inside solid geometry

The Parent element column must contain a category filter.

Example 1 – record the structural material of walls containing the targets.

	#	On	Parent element	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy wall material to contained elements							
<div><div></div><div></div><div></div><div></div><div></div></div>	1	☒	Categories: Walls AND Parameters:	Categories: AND Parameters:		\$(Comments)=in wall - SS...	T3 Aa <div><div></div><div></div><div></div></div>
Formula: \$(Comments)=in wall - - SStr(@[Structural Material Name],5)							

Parent element – elements in the “Walls” category

Target element – elements in all categories.

No filter is defined, so elements with solid geometry inside walls are processed.

Comments receives “in wall -” followed by the Name of the wall's Structural Material reference, starting at character index 5, using SStr(s,i).

Select Auxiliary rays. This algorithm identifies the single parent containing the target solid's centroid.

Example 2 – copy a parameter from a linked room to a space.

Example 2 – Copy linked room data to spaces							
<div><div></div><div></div><div></div><div></div><div></div></div>	2	☒	Categories: Rooms AND Parameters:	Categories: Spaces AND Parameters:		\$(Comments)=@[Depart...	T3 Aa <div><div></div><div></div><div></div></div>
Formula: \$(Comments)=@[Department]							

Parent element – elements in the “Rooms” category

Target element – elements in the “Spaces” category

The space's Comments receives Department from the room in the linked file.

Example 3 – write intersected floor marks to a pipe.

Example 3 – Collect intersected floor marks in a pipe						
	3	☒	Categories: Floors AND Parameters:	Categories: Pipes AND Parameters:	$[\text{Mark}] = [\text{Mark}] @ [\text{Mark}]$	T3 Aa  

Parent element – elements in the “Floors” category.

Target element – elements in the “Pipes” category.

Grouping by pipe ID combines its pairs with intersected floors. `GJoin()` collects comma-separated floor marks, removes duplicates (d) and sorts values (s).

For “ Π_2 ”, “ Π_1 ” and “ Π_2 ”, the result is “ Π_1, Π_2 ”.

Select Boolean operations to include intersections with multiple floors.

No prior clearing is needed. Each run builds a new list and replaces the previous mark.

A pipe with no qualifying floor intersections is not processed; its previous mark is retained.

For details on writing expressions, see the [Formula Editor](#) section.

For details on using functions, see the [Functions](#) section.

15. Rebar In System – processing rebar in a reinforcement system

These systems include structural area reinforcement and structural path reinforcement.

The System column must contain a category filter, a parameter filter, or both.

Example 1 – copy an area reinforcement parameter to its rebars.

	#	On	System	Rebar	Grouping: Rebe	Formula	Options
Example 1 – Copy the reinforcement system mark							
	1	☒	Categories: Structural A... AND Parameters:	Parameters:		\$(MP_Note)=@[Mark]	T3 Aa 
Formula: \$(MP_Note)=@[Mark]							

System – elements in the “Structural Area Reinforcement” category

Rebar – elements in the “Structural Rebar” category.

No parameter filter is defined.

The rebars' “MP_Note” receives the area reinforcement Mark.

Example 2 – set a rebar parameter based on its length within an area reinforcement system.

Example 2 – Assign a product mark by bar length							
	2	☒	Categories: Structural A... AND Parameters:	Parameters:		\$(MP_Mark)=FirstTrue(InR...	T3 Aa 
Formula: \$(MP_Mark)=FirstTrue(InRange(\$[Bar·Length],0..2000):t1;InRange(\$[Bar·Length],2001..4000):t2;InRange(\$[Bar·Length],4001..6000):t3;t4)							

System – elements in the “Structural Area Reinforcement” category

Rebar – elements in the “Structural Rebar” category.

No parameter filter is defined.

“MP_Mark” depends on the range containing Bar Length, using InRange(x,min..max).

A 2550 mm bar receives “t2”; a 6900 mm bar receives “t4”.

Example 3 – set a rebar key parameter from the area reinforcement type name.

✎ Example 3 – Set a rebar key from the system type

⋮	3	☒	Categories: Structural A... AND Parameters:	Parameters:		\$(MP_Rebar type)=FirstTru...	T3 Aa □ ↺
---	---	-------------------------------------	---	-------------	--	-------------------------------	--------------

Formula:
\$(MP_Rebar type)=FirstTrue(@[Type Name]=TX:2. Top X;@[Type Name]=TY:3. Top Y;@[Type Name]=BX:4. Bottom X;@[Type Name]=BY:5. Bottom Y;@[Type Name]=Starters:6. Starters;@[Type Name]=Details:7. Details;@[Type Name]=Main:1. Main;@[Type Name]=Meshes:13. Mesh;)

System – elements in the “Structural Area Reinforcement” category

Rebar – elements in the “Structural Rebar” category.

No parameter filter is defined.

“MP_Rebar type” depends on the system's Type Name, using FirstTrue(c1:v1;...;cN:vN;v).

For details on writing expressions, see the Formula Editor section.

For details on using functions, see the Functions section.

17. In MEP System – processing elements in MEP systems

These include electrical circuits, piping systems and duct systems.
The MEP system column must contain a category filter, a parameter filter, or both.

Example 1 – assign a position number to piping system elements.

	#	On	MEP system	Target element	Grouping: Targ	Formula	Options
Example 1 – Set positions in a polyethylene system							
	1	☒	Categories: Piping Syst... AND Parameters: Type~Poly...	Categories: AND Parameters:		\$(MP_Position)=3	T3 Aa 
Formula: \$(MP_Position)=3							

MEP system – elements in the “Piping Systems” category
Target element – elements in all categories.
No filter is defined. Elements in piping systems whose Type contains “Polyethylene” are processed.
“MP_Position” receives 3 for target elements that match the filters.

Example 2 – calculate duct cross-sectional area.

Example 2 – Calculate duct cross-sectional area							
	2	☒	Categories: Duct Syst... AND Parameters: Type~Sup...	Categories: Ducts AND Parameters:		\$(MP_Cross-sectional area...	T3 Aa 
Formula: \$(MP_Cross-sectional area)=If(\$[Size]~x,FirstTrue(StrIndex(\$[Size],x)=3:{{SStr(\$[Size],0,3)}*{{SStr(\$[Size],4)}/1000000}};StrIndex(\$[Size],x)=4:{{SStr(\$[Size],0,4)}*{{SStr(\$[Size],5)}/1000000}};0),{\$[Size]*\$[Size]/1000000*Pi()/4})							

MEP system – elements in the “Duct Systems” category
Target element – elements in the “Ducts” category
“MP_Cross-sectional area” receives the calculated duct area. If Size contains “x”, StrIndex(s1,s2) finds its position and SStr(s,i,l) extracts the two dimensions. Their product is divided by 1000000. Otherwise, the circular area is Size squared, multiplied by π and divided by 1000000 and 4.

For Size = “300x500”, “x” is at index 3; indices start at 0. The first three characters, “300”, are multiplied by the substring starting at index 4, “500”. Dividing by 1000000 gives 0.15 m².

Example 3 – write connected plumbing fixture names to a system, without duplicates and in alphabetical order.

✎ Example 3 – Collect equipment names in the system

⋮	3	☒	Categories: Piping Syst... AND Parameters:	Categories: Plumbin... AND Parameters:		Join(@[Comments],,,ds)=...	T3 Aa □ ↺
---	---	-------------------------------------	--	--	--	----------------------------	--------------

Formula:
Join(@[Comments],,,ds)=\$[MP_Name]

MEP system – elements in the “Piping Systems” category

Target element – elements in the “Plumbing Fixtures” category.

Join(p,d,f) writes a comma-separated list of the fixtures' “MP_Name” values to the MEP system's Comments. The third argument, ds, removes duplicate names and sorts them alphabetically.

For details on using functions, see the [Functions](#) section.

21. In Assembly – processing assembly members

The Assembly column does not require a parameter filter; Target element does not require category or parameter filters.

Example 1 – copy an assembly name to the marks of its reinforcing bars.

	#	On	Assembly	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy the assembly name to rebar marks							
	1	☒	Parameters:	Categories: Structur... AND Parameters:		\$(Mark)=@[Type]	T3 Aa □ ↺
Formula: \$(Mark)=@[Type]							

Assembly – elements in the “Assemblies” category.

No assembly parameter filter is defined.

Target element – elements in the “Structural Rebar” category

Rebars belonging to assemblies are processed. Each target's Mark receives the assembly Type.

Example 2 – write the first and last element marks to their assembly.

Example 2 – Write the first and last marks to the assembly							
	2	☒	Parameters:	Categories: AND Parameters: Mark<>		First(@[MP_First mark],s)=... Last(@[MP_Last mark],s)=...	T3 Aa □ ↺
Formula: First(@[MP_First mark],s)=\$(Mark) Last(@[MP_Last mark],s)=\$(Mark)							

Assembly – elements in the “Assemblies” category.

Target element – elements in all categories.

Assembly members with nonempty marks are processed.

The s argument sorts the accumulated values before selecting one. Without it, the result depends on an undefined processing order.

For marks “A03”, “A01” and “A02”, the assembly receives “A01” and “A03”.

12. Connection SubElements – processing structural connection subelements

These include plates, bolts, anchors and other connection subelements.

The Subelement category filter lists only categories that can belong to structural connections. A parameter filter and a parameter list in the Formula Editor are not available.

The Subelement column must contain a category filter.

Example 1 – fill the profile name for plates.

	#	On	Connection	SubElement	Grouping: SubE	Formula	Options
Example 1 – Build a plate profile name							
	1	☒	Parameters:	Categories: Plates		\$(Profile Name)= - \$[Thick...	T3 Aa  
Formula: \$[Profile Name]=--\$[Thickness]xROUND(\$[Width])L=Round(\$[Length])/\$[Structural Material]							

Connection – elements in the “Structural Connections” category.
No parameter filter is defined, so all connections are processed.

Subelement – elements in the “Plates” category

“Profile Name” receives a composite value built from other target parameters, including values rounded using Round(a1). For example: 10x236L=100/Aluminium.

Example 2 – identify the connection type for subelements.

Example 2 – Copy the connection type to subelements							
	2	☒	Parameters: Type~Link	Categories:		\$(MP_Node)=@[Type]	T3 Aa  
Formula: \$[MP_Node]=@[Type]							

Connection – elements in the “Structural Connections” category.
Type must contain “Link”.

Subelement – elements in the “Anchors”, “Plates”, “Bolts” and “Structural Framing” categories.

“MP_Node” receives the Type of the structural connection containing the subelement.

13. Nested Families – processing shared nested families

Element filters are optional in the Parent family and Nested family columns.

Example 1 – transfer a system name from a parent family to its nested family.

	#	On	Parent family	Nested family	Grouping: Nest	Formula	Options
Example 1 – Copy the parent system name							
<div><div></div><div></div><div></div></div>	1	☒	Categories: Mechanical... AND Parameters:	Categories: Mechan... AND Parameters:		\$(System Name)=@[Syste...	T3 Aa <div><div></div><div></div></div>
Formula: \$(System Name)=@[System Name]							

Parent family – elements in the “Mechanical Equipment” category
Nested family – elements in the “Mechanical Equipment” category
The nested family's System Name receives the parent family's System Name.

Example 2 – record parent family information in every nested family.

Example 2 – Record parent family information							
<div><div></div><div></div><div></div></div>	2	☒	Categories: AND Parameters:	Categories: AND Parameters:		\$(Comments)=nested\, ID...	T3 Aa <div><div></div><div></div></div>
Formula: \$(Comments)=nested\, ID-@[ID] @[Family and Type]							

Parent family – elements in all categories.
Nested family – elements in all categories.
No filters are defined. All shared nested family instances in the model are processed.
The nested family's Comments receives “nested, ID”, the parent element ID, and the parent Family and Type names.

Example 3 – transfer a system name from a nested family to its parent.

Example 3 – Copy the trap system to the sink

<div><div></div><div></div><div></div><div></div><div></div></div>	3	<div><div></div></div>	<div>Categories: Plumbing F...</div> <div>AND</div> <div>Parameters: Family~Sink</div>	<div>Categories: Plumbin...</div> <div>AND</div> <div>Parameters: Family~...</div>		<div>Join(@[MP_System],)=\${M...</div>	<div>T3 Aa</div> <div><div></div><div></div></div>
--	---	------------------------	--	--	--	--	--

Formula:

Join(@[MP_System],)=\${MP_System]

Parent family – elements in the “Plumbing Fixtures” category.
Family must contain “Sink”.

Nested family – elements in the “Plumbing Fixtures” category.
Family must contain “Trap”.

The parent's “MP_System” receives the nested family's “MP_System”, using Join(p,d,f).

For details on writing expressions, see the [Formula Editor](#) section.

18. Curtain Wall Components – processing curtain wall components

These include curtain wall mullions and panels, as well as windows and doors. The source element is a wall.

The Curtain wall column does not require a parameter filter; Target element does not require category or parameter filters.

Example 1 – transfer a curtain wall Mark to its panels and mullions.

	#	On	Curtain Wall	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy curtain wall marks to components							
	1	☒	Parameters:	Categories: AND Parameters:		\$(Mark)=@[Mark]	T3 Aa 
Formula: \$(Mark)=@[Mark]							

Curtain wall – elements in the “Walls” category.

No curtain wall parameter filter is defined.

Target element – elements in all categories.

No filters are defined, so all curtain wall components are processed.

Each target's Mark receives the Mark of its parent curtain wall.

Example 2 – collect curtain wall door marks and transfer them to adjacent walls.

Two strategies run in sequence: Curtain Wall Components collects door marks in curtain walls; Solid Touch transfers the lists to ordinary walls.

First stage – Curtain Wall Components.

Example 2 – Collect door marks in curtain walls							
	2	☒	Parameters:	Categories: Doors AND Parameters:		Join(@[MP_Door marks],,,ds)	T3 Aa 
Formula: Join(@[MP_Door marks],,,ds)=\$(Mark)							

Curtain wall – elements in the “Walls” category

Target element – elements in the “Doors” category

Each curtain wall receives a comma-separated list of its door marks, without duplicates and in alphabetical order.

Marks “D2”, “D1” and “D2” produce “D1,D2”. For a second curtain wall with “D2” and “D3”, the list is “D2,D3”.

No grouping is defined. Join() accumulates values separately for each curtain wall.

Second stage – Solid Touch.

Curtain Wall Components – Example 2 – Collect curtain wall lists in adjacent walls						
	4	☒	Categories: Walls AND Parameters: Type Nam...	Categories: Walls AND Parameters: Type N...	Join(@[MP_Door marks],\...	T3 Aa 
Formula: Join(@[MP_Door marks],\n,ds)=\$[MP_Door marks]						

Touching element – elements in the “Walls” category.

Target element – elements in the “Walls” category

Parameter filters distinguish ordinary walls from curtain walls.

The wall receives the door-mark lists from its adjacent curtain walls. Each list occupies a separate line. Identical lists are removed and the remaining lists are sorted.

Result for two curtain walls:

“D1,D2
D2,D3”

Join() receives complete strings. The d flag removes identical strings, not individual marks within different strings. “D2” therefore remains in both lists.

20. Parts – processing parts

If a part was created from several elements, the first is treated as its source. Source element category and parameter filters are optional. The Part parameter filter is also optional.

Example 1 – transfer wall marks to parts created from the walls.

	#	On	Source element	Part	Grouping: Part	Formula	Options
✎ Example 1 – Copy the source wall mark to parts							
⋮	1	☒	Categories: Walls AND Parameters:	Parameters:		\$(Comments)=@[Mark]	T3 Aa  
Formula: \$(Comments)=@[Mark]							

Source element – elements in the “Walls” category
Part – elements in the “Parts” category.
No parameter filter is defined, so all parts are processed.
A part's Comments receives the Mark of its source wall.

For details on writing expressions, see the [Formula Editor](#) section.

23. Stair Components – processing stair components

These include stair runs, landings, supports and associated railings. The source element is the stair.

The Stair parameter filter is optional.

Example 1 – transfer a stair Mark to its runs and landings.

	#	On	Stair	Stair component	Grouping: Stair	Formula	Options
✎ Example 1 – Copy the stair mark to components							
⋮	1	☒	Parameters:	Categories: Runs, La... AND Parameters:		\$(Mark)=@[Mark]	T3 Aa □ ↺
Formula: \$(Mark)=@[Mark]							

Stair – elements in the “Stairs” category

Component – elements in the “Stair Runs” and “Stair Landings” categories.

Each target's Mark receives the Mark of its parent stair.

For details on writing expressions, see the [Formula Editor](#) section.

11. Room Boundaries – processing room boundary elements

These include walls, columns, floors and room separation lines.

Element filters are optional in the Room and Target element columns.

Example 1 – write the numbers of all rooms bounded by a wall to that wall.

	#	On	Room	Target element	Grouping: Target	Formula	Options
Example 1 – Collect room numbers in boundary walls							
	1	☒	Parameters:	Categories: Walls AND Parameters:	\$(ID)	\$(Mark)=GJoin(@[Number],\,...	T3 Aa 
Formula: \$(Mark)=GJoin(@[Number],\,...,ds) Grouping: Target element: \$(ID)							

Target element – elements in the “Walls” category

Group the elements by ID.

Grouping combines relationships for the same wall. GJoin() collects comma-separated room numbers, removes duplicates and sorts the values.

A wall bounding rooms “101” and “102” receives “101,102”.

For details on writing expressions, see the Formula Editor section.

24. Room Openings – processing room openings

These include windows and doors.

Element filters are optional in the Room and Target element columns.

Example 1 – calculate the total area of rectangular room openings.

	#	On	Room	Opening	Grouping: Openi	Formula	Options
Example 1 – Calculate total room opening area							
	1	☒	Parameters:	Categories: Windows,... AND Parameters:		Sum(@[MP_Opening area])=...	T3 Aa  
Formula: Sum(@[MP_Opening area])={@[Height]*@[Width]/1000000}							

Opening – elements in the “Doors” and “Windows” categories.

Dimensions are read in millimeters; the result is written in square meters.

Each opening's Height is multiplied by Width. After processing, the sum of the opening areas is written to the room.

A 2100 × 900 mm door and a 1500 × 1200 mm window give 1.89 + 1.80 = 3.69 m².

For details on writing expressions, see the [Formula Editor](#) section.

26. Room Finishing – processing room finishes

The strategy processes room finishes created by the Room Finishing plugin. The room is the source; its finishes are the targets, including finishes manually linked using that plugin. Finishes created by other tools are not processed. Filters are optional in the Room and Target element columns.

Example 1 – transfer room information to finishes and write finish information to the room.

	#	On	Room	Finishing element	Grouping: Finishi	Formula	Options
Example 1 – Copy room data and collect ceiling finishes							
<div><div></div><div></div><div></div><div></div></div>	1	☒	Parameters:	Categories: Ceilings AND Parameters:		<code>\$(MP_Room Number)=@[Nu...</code> <code>\$(MP_Room Name)=@[Name]</code> <code>Sum(@[MP_Ceiling finish are...</code>	T3 Aa <div><div></div><div></div></div>
Formula: <code>\$(MP_Room Number)=@[Number]</code> <code>\$(MP_Room Name)=@[Name]</code> <code>Sum(@[MP_Ceiling finish area])=[\$Area]</code> <code>Join(@[Ceiling Finish],,ds)=\$[Type]</code>							

Target element – elements in the “Ceilings” category.
Ceilings created or linked by the Room Finishing plugin are processed.
The room Number and Name are copied to the ceilings.
The room receives the total ceiling area and a sorted list of ceiling types without duplicates.

Example 2 – calculate the height of the ceiling void.

Example 2 – Calculate ceiling void height							
<div><div></div><div></div><div></div><div></div></div>	2	☒	Parameters:	Categories: Ceilings AND Parameters:		<code>\$(MP_Ceiling void height)={S...</code>	T3 Aa <div><div></div><div></div></div>
Formula: <code>\$(MP_Ceiling void height)={SourceTopZ()-TargetBottomZ()}</code>							

Target element – elements in the “Ceilings” category.
Ceilings created by the Room Finishing plugin are processed.
SourceTopZ() returns the room's highest elevation.
TargetBottomZ() returns the ceiling's lowest elevation. Both elevations are in millimeters.

“MP_Ceiling void height” receives the difference between these elevations. For a room top elevation of 3300 mm and a ceiling bottom elevation of 3000 mm, the result is 300 mm.

Example 3 – write wall finish types and their areas to a multiline room parameter.

First stage – sum the areas by finish type.

Example 3 – Calculate wall finish area by type

<div><div></div><div></div><div></div><div></div></div>	3	☒	Parameters:	Categories: Walls AND Parameters:	@[ID]_\${TypeID}	\${Comments}=\${Type} - GSum...	T3 Aa <div><div></div><div></div></div>
---	---	-------------------------------------	-------------	---	------------------	---------------------------------	--

Formula:
\${Comments}=\${Type} - - GSum(\${Area})
Grouping: Finishing element: @[ID]_\${TypeID}

Target element – elements in the “Walls” category. Separate finish walls created or linked by the Room Finishing plugin are processed.

Group the elements by: @[ID]_\${TypeID}

The composite key groups walls of the same type within one room. Walls of that type in other rooms form separate groups.

Each wall receives its type name and its group's total area. Three “Paint” walls totalling 35.8 m² each receive “Paint - 35.8 m²”. “Tile” walls totalling 8.3 m² receive “Tile - 8.3 m²”.

Example 3 – Collect wall finish lines in rooms

<div><div></div><div></div><div></div><div></div></div>	4	☒	Parameters:	Categories: Walls AND Parameters:		Join(@[Wall Finish],\n,d)=\$[Co...	T3 Aa <div><div></div><div></div></div>
---	---	-------------------------------------	-------------	---	--	------------------------------------	--

Formula:
Join(@[Wall Finish],\n,d)=\$[Comments]

Second stage – write the finish list to the room

Target element – elements in the “Walls” category

No grouping is defined. The same rooms and finish walls are used as in the first strategy.

Join() collects strings from the walls’ Comments and writes them to the room. The d argument removes duplicate strings; the \n separator starts a new line.

Result in “Wall Finish”:

Plaster - 12.4 m²

Primer - 35.8 m²

Putty - 35.8 m²

Paint - 35.8 m²

Tile - 8.3 m²

With d, line order is undefined. Use ds to sort lines alphabetically.

The two stages first calculate areas using GSum(), then collect unique strings using Join(). Aggregate and write-back functions cannot be combined in one expression.

Each finish type must be represented by separate walls of the corresponding type. This example does not extract individual layer names or areas from a compound wall.

For details on writing expressions, see the Formula Editor section.

10. Solid Touch – processing elements with touching solid faces

The Touching element column must contain a category filter.

Example 1

	#	On	Touched element	Target element	Grouping: Targ	Formula	Options
Example 1 – Copy room data to floors							
⋮	1	☒	Categories: Rooms AND Parameters:	Categories: Floors AND Parameters:		\$(MP_Number of the apar...	T3 Aa □ 🔍
Formula: \$(MP_Number of the apartment room)=@[Number]_@[Name]							
Example 1 – Copy floor finish data to rooms							
⋮	2	☒	Categories: Floors AND Parameters:	Categories: Rooms AND Parameters:		\$(Floor Finish)=@[Floor fi...	T3 Aa □ 🔍
Formula: \$(Floor Finish)=@[Floor finish type]							

Rule 1

Touching element – elements in the “Rooms” category.

Target element – elements in the “Floors” category

The target floor's “MP_Number of the apartment room” receives a value composed of the touching room's Number and Name.

Rule 2

Touching element – elements in the “Floors” category.

Target element – elements in the “Rooms” category

The target room's Floor Finish receives the floor's “Floor finish type”.

* Set a face-to-face distance tolerance between 0 and 20 mm in the plugin settings.

Example 2 – calculate the total length of vertical joints between interior and exterior walls.

Example 2 – Calculate vertical joint lengths

<div><div></div><div></div><div></div><div></div></div>	3	☒	Categories: Walls AND Parameters: Function~...	Categories: Walls AND Parameters: Functio...	\$(ID)	\$(MP_Joint count)=GCoun... \$(MP_joint length)={{Targ...	T3 Aa <div></div>
---	---	-------------------------------------	--	--	--------	--	----------------------

Formula:
\$(MP_Joint count)=GCount(s)
\$(MP_joint length)={{TargetTopZ()-TargetBottomZ()}}*GCount(s)*2}
Grouping: Target element: \$(ID)

Touching element – elements in the “Walls” category.

Target element – elements in the “Walls” category

The touching walls' type Function must contain “Exterior”; the targets' must contain “Interior”.

Group the elements by ID.

GCount(s) counts distinct exterior walls touching the current interior wall.

The difference between elevations gives the target wall's geometric height.

For an interior wall 3000 mm high touching two exterior walls, the joint length is $3000 \times 2 \times 2 = 12000$ mm.

For details on writing expressions, see the [Formula Editor](#) section.

14. MEP Insulation – processing pipe and duct insulation

The Pipe or duct column must contain a category filter, a parameter filter, or both.

Example 1 – write a duct's area to an insulation parameter.

	#	On	Pipe or Duct	Insulation	Grouping: Insulat	Formula	Options
Example 1 – Copy duct area to insulation							
	1	☒	Categories: Ducts AND Parameters:	Categories: Duct Insul... AND Parameters:		<code>\$(MP_Quantity)=@[Area]</code>	T3 Aa  
Formula: <code>\$(MP_Quantity)=@[Area]</code>							

Pipe or duct – elements in the “Ducts” category
Insulation – elements in the “Duct Insulations” category.
The insulation's “MP_Quantity” receives the duct Area.

Example 2 – add 6 mm pipe insulation.

1. Use Self to create the insulation:

MEP Insulation – Example 2 – Create pipe insulation							
	18	☒	Categories: Pipes AND Parameters: Type Name~PN10			<code>\$(Insulation Type)=MP_insulation tubes</code>	T3 Aa  
Formula: <code>\$(Insulation Type)=MP_insulation tubes</code>							

Target element – elements in the “Pipes” category.
Type Name must contain “PN10”.

2. Use MEP Insulation to set the required thickness:

Example 2 – Set pipe insulation thickness to 6 mm							
	2	☒	Categories: Pipes AND Parameters: Typ...	Categories: Pipe Insul... AND Parameters:		<code>\$(Insulation Thickness)=6</code>	T3 Aa  
Formula: <code>\$(Insulation Thickness)=6</code>							

28. Rebar Couplers – processing rebar couplers

The source is a structural rebar coupler; the targets are the one or two bars connected to its ends. A coupler on one bar forms a one-to-one relationship, so write-back functions transfer that bar's value to the coupler. For two connected bars, the write-back expression is evaluated for both.

Parameter filters are optional in the Coupler and Rebar columns.

Example 1 – transfer a coupler's mark to its connected rebars.

	#	On	Coupler	Rebar	Grouping: Rebar	Formula	Options
✎ Example 1 – Copy coupler marks to rebars							
⋮	1	☒	Parameters:	Parameters:		\$(MP_Coupler mark)=@[Mark]	T3 Aa  
Formula: \$(MP_Coupler mark)=@[Mark]							

Coupler – elements in the “Rebar Couplers” category

Rebar – elements in the “Structural Rebar” category

The rebars connected by the selected coupler are processed.

Each bar's “MP_Coupler mark” receives the coupler's Mark.

If a bar has multiple couplers, different values may overwrite each other. This example assumes one coupler per bar.

For details on writing expressions, see the [Formula Editor](#) section.

22. ViewSpecific – processing view-specific elements

The strategy processes elements that belong to a single view rather than the 3D model: dimensions, text notes, tags, detail lines and regions, and sheet content. They appear only in the view where they were created. Model elements such as walls, floors and equipment are unaffected.

Element filters are optional in the View and Target element columns.

Example 1 – hide the upper corner graphic of title blocks on sheets at stage “P”.

	#	On	View	Target element	Grouping: Target	Formula	Options
Example 1 – Hide the title block corner graphic							
	1	☒	Parameters: MP...	Categories: Title Blocks AND Parameters:		\$(MP_Page number)=0 \$(MP_Sheet count)=0	T3 Aa  
Formula: \$(MP_Page number)=0 \$(MP_Sheet count)=0							

View – elements in the “Sheets” category

Target element – elements in the “Title Blocks” category.

Clear the title block's Yes/No parameters that control the upper corner graphic.

Example 2 – transfer the sheet number to revision clouds.

Example 2 – Copy sheet numbers to revision clouds							
	2	☒	Parameters:	Categories: Revision C... AND Parameters:		\$(Sheet Number)=[Sheet...]	T3 Aa  
Formula: \$(Sheet Number)=[Sheet Number]							

View – elements in the “Sheets” category

Target element – elements in the “Revision Clouds” category

Only clouds placed directly on a sheet are processed.

The sheet number is written to revision clouds belonging to that sheet.

Example 3 – collect revision cloud numbers in a sheet parameter.

Example 3 – Collect revision numbers in sheets							
⋮	3	☒	Parameters:	Categories: Revision Cl... AND Parameters:		Join(@[MP_Revision numbers]...	T3 Aa □ ↺
Formula: Join(@[MP_Revision numbers],,,d)=\$[MP_Revision number]							
Example 3 – Compress revision numbers into ranges							
⋮	4	☒	Parameters:	Categories: Revision Cl... AND Parameters:		@[MP_Revision numbers]=Ra...	T3 Aa □ ↺
Formula: @[MP_Revision numbers]=Ranges(@[MP_Revision numbers],\,,)							

View – elements in the “Sheets” category

Target element – elements in the “Revision Clouds” category.
Only clouds placed directly on the sheet are processed.

Two rules run in sequence.

The first collects unique numbers. The second converts the list to ranges using Ranges(s,d).

For example, “1,2,3,5,6,7,10” becomes “1-3,5-7,10”.

For details on writing expressions, see the Formula Editor section.

27. On Level – processing elements associated with a level

The level is the source; elements associated with it are the targets. The strategy uses the element's level property, or a level parameter such as Level, Reference Level or Base Constraint if that property is empty. Elements spanning several levels belong to their lowest level. Source levels are included automatically; they need not be selected.

Element filters are optional in the Level and Target element columns.

Example 1 – transfer a level code to equipment.

	#	On	Level	Target element	Grouping: Target	Formula	Options
✎ Example 1 – Copy the level code to equipment							
⋮	1	☒	Parameters:	Categories: Mechanical... AND Parameters:		\$(MP_Level)=@[MP_Level co...	T3 Aa □ ↺
Formula: \$(MP_Level)=@[MP_Level · code]							

Target element – elements in the “Mechanical Equipment” category.
Instances associated with the selected level are processed.
“MP_Level” receives the level’s “MP_Level code”.

For details on writing expressions, see the [Formula Editor](#) section.

29. Level Views – processing views associated with a level

The level is the source; floor plans, ceiling plans, structural plans and area plans based on it are the targets. A view's level is fixed when the view is created: a duplicated plan retains the original level. Sections, elevations, 3D views, legends, schedules and view templates are excluded.

Document processing uses all levels; selected-element processing uses selected levels; view processing uses visible levels. Levels can be picked in sections and elevations. A plan has no level line, so processing a plan uses the level on which it is based.

Element filters are optional in the Level and View columns.

Example 1 – construct plan names from their level names.

	#	On	Level	View	Grouping: View	Formula	Options
Example 1 – Name plans after their levels							
	1	☒	Parameters:	Parameters:		\$(View Name)=AR_@[Name]	T3 Aa ☐ ☐
Formula: \$(View Name)=AR_@[Name]							

View – plan views associated with the processed levels.

For the level “Level 02”, the plan name is “AR_Level 02”.

Example 2 – extract a section code from a level name.

Example 2 – Extract a section code from level names							
	2	☒	Parameters:	Parameters:		\$(MP_Section code)=LStr(@[...]	T3 Aa ☐ ☐
Formula: \$(MP_Section code)=LStr(@[Name],LStrIndex(@[Name],_))							

Level names follow “Section_1_Level02”; the final part after the underscore identifies the floor.

LStrIndex() finds the last underscore. LStr() extracts the preceding text.

“Section_1_Level02” produces “Section_1”. Only levels following this naming pattern are processed.

19. One To Many – processing elements relative to specified source elements

If the source filters find several elements, the last element found is used.

Both the Source element and Target element columns must contain category filters.

Use the pick button to select a source in the model. Its filters are filled automatically. 

Example 1 – copy several parameters from one duct to all other ducts on the view.

	#	On	Source element	Target element	Grouping: Target	Formula	Options
Example 1 – Copy duct parameters to other ducts							
	1	☒	Categories: Ducts AND Parameters: ID=11...	Categories: Ducts AND Parameters:		<code>\$(MP_Kit)=@[MP_Kit]</code> <code>\$(MP_Position)=@[MP_Posit...</code> <code>\$(MP_Note)=@[MP_Note]</code>	T3 Aa 
Formula: <code>\$(MP_Kit)=@[MP_Kit]</code> <code>\$(MP_Position)=@[MP_Position]</code> <code>\$(MP_Note)=@[MP_Note]</code>							

Source element – elements in the “Ducts” category.
The source is picked on the view; its category and parameter filters are filled automatically.

Target element – elements in the “Ducts” category
The source values of “MP_Kit”, “MP_Position” and “MP_Note” are copied to the targets.

Example 2 – collect the total number of pipe supports in a selected accounting element.

Example 2 – Count pipe supports in an accounting element

<div><div></div><div></div><div></div></div>	2	☒	Categories: Generi... AND Parameters: ID=26...	Categories: Pipes AND Parameters: Type Na...		@[MP_Support count]=GSu...	T3 Aa <div></div>
--	---	-------------------------------------	--	--	--	----------------------------	----------------------

Formula:

@[MP_Support · count]=GSum({RoundUp(\$[Length]/2000)+1})

Source element – elements in the “Generic Models” category.
Select one family instance, such as a small accounting cube.

Target element – elements in the “Pipes” category whose Type Name contains “PN-10”.

Pick the source in the model; its filters are filled automatically.

Supports are required at both ends of each pipe, with no more than 2000 mm between supports.

No grouping is defined: all pipes selected by the rule form one group.
RoundUp() rounds the number of intervals upward; adding 1 gives the support count. GSum() adds the counts.

Calculations for individual pipes:

1500 mm → 2 supports; 2000 mm → 2; 2500 mm → 3; 4500 mm → 4. The total for these four pipes is 11.

The result is written directly to the source. GSum() returns the same total for every pair; repeated writes of the same result are skipped.

25. Many To Many – matching independent sets of elements by calculated values

The strategy processes independent source and target sets. Pairs are formed by matching calculated values, rather than model relationships. Matching elements form a group used by aggregate functions.

The Source elements and Target element columns must contain category filters.

Example 1 – transfer data between equipment from different disciplines using a code.

	#	On	Source elements	Target element	Matching	Formula	Options
Example 1 – Match equipment across disciplines							
⋮	1	☒	Categories: Mechan... AND Parameters:	Categories: Mechanical... AND Parameters:	@[MP_Equip... ↔ [MP_Mat...	\$(Mark)=@[Mark] \$(MP_Note)=@[MP_Note]	T3 Aa □ ↺
Formula: \$(Mark)=@[Mark] \$(MP_Note)=@[MP_Note] Matching: @[MP_Equipment code] ↔ [MP_Matching code]							

Source elements – elements in the “Mechanical Equipment” category

Target element – elements in the “Mechanical Equipment” category

Matching uses “MP_Equipment code” on sources and “MP_Matching code” on targets.

Data is transferred between matching codes. Check the matching preview to ensure that each code has exactly one source element.

If codes differ in formatting, such as “AB-001”, “AB 001” and “ AB-001 ”, replace the matching expressions with:

Example 1 – Normalize matching codes							
⋮	2	☒	Categories: Mechan... AND Parameters:	Categories: Mechanical... AND Parameters:	RgxReplace(... ↔ RgxReplac...	\$(Mark)=@[Mark] \$(MP_Note)=@[MP_Note]	T3 Aa □ ↺
Formula: \$(Mark)=@[Mark] \$(MP_Note)=@[MP_Note] Matching: RgxReplace(Trim(@[MP_Equipment code]),"[-]","") ↔ RgxReplace(Trim(\$[MP_Matching code]),"[-]","")							

Match source elements by:

```
RgxReplace(Trim(@[MP_Equipment code]),"[- ]","")
```

Match target elements by:

```
RgxReplace(Trim($[MP_Matching code]),"[- ]","")
```

Trim() removes leading and trailing spaces. RgxReplace() removes internal hyphens and ordinary spaces. All these variants produce "AB001".

Verify that each matching key has exactly one source element.

Example 2 – count each apartment's bathrooms and balconies and write the result to its room marked "1".

Example 2 – Count apartment bathrooms and balconies

<div><div></div><div></div><div></div></div>	3	☒	Categories: Rooms AND Parameters: AR_Apa...	Categories: Rooms AND Parameters: Name=Bat...	@[AR_Apart... ↔ \$[AR_Apa...	@[Comments]=GCount(t)	T3 Aa <div></div>
--	---	-------------------------------------	---	---	---------------------------------	-----------------------	----------------------

Formula:
@[Comments]=GCount(t)
Matching: @[AR_Apartment ID]
↔ \$[AR_Apartment ID]

Source elements – elements in the "Rooms" category.
"AR_Apartment marker" must equal 1. Select one such room per apartment.

Target element – elements in the "Rooms" category.
Name must equal "Bathroom" or "Balcony".

Match sources and targets by "AR_Apartment ID".

The source room's Comments receives the total count of bathrooms and balconies with the same apartment identifier.

Two bathrooms and one balcony produce 3.

The result is written directly to the source. GCount(t) returns the same value for every pair in the group.

Element matching

Many To Many uses independent source and target sets. Matching defines the relationship between them.

Each set has its own expression. Elements with equal results form a matching group. Each target in the group is processed relative to each source.

A matching group is the rule's group. GSum(), GCount(), GJoin() and other aggregate functions operate on it. No separate grouping is defined for Many To Many.

Double-click a cell in the Matching column to open its settings.

The window requires two expressions:

Match source elements by – an expression using source parameters, referenced as @[parameter].

Match target elements by – an expression using target parameters, referenced as \$[parameter].

Both expressions are required. Each is evaluated with only an element from its own set available. Source expressions cannot access target parameters, and target expressions cannot access source parameters.

Ignore case matches strings regardless of letter case. Numeric values are detected automatically and rounded to the numeric comparison tolerance. An empty expression result causes the element to be skipped.

The example matches rooms and spaces by number.

 MATCHING

×

[Instruction](#)

Conditionally source and target elements fall into one group when the values calculated on both sides coincide. This group is what aggregate functions are calculated over

Match source elements by:

@[Number]




Match target elements by:

\$_[Number]




☐ Sort

☐ Descending

☐ Ignore case

Sort within the group by:





Matching preview

Accept

Cancel

Source elements – elements in the “Rooms” category

Target elements – elements in the “Spaces” category

@[Number] retrieves the room number; \$_[Number] retrieves the space number. Equal values place the elements in the same matching group.

 MATCHING PREVIEW ×

Incomplete groups come first: their elements are skipped during processing

☐ Processing elements from linked files

Value	Source	Target	Source examples	Target examples	Problem
4	0	1		Spaces: Space 4	No conditionally source elements with this value
5	0	1		Spaces: Space 5	No conditionally source elements with this value
6	0	1		Spaces: Space 6	No conditionally source elements with this value
104	1	0	Rooms: Laundry 104		No target elements with this value
105	1	0	Rooms: Hall 105		No target elements with this value
106	1	0	Rooms: Living 106		No target elements with this value
101	1	1	Rooms: Kitchen & Dini	Spaces: Space 101	
102	1	1	Rooms: Mech. 102	Spaces: Space 102	
103	1	1	Rooms: Bath 103	Spaces: Space 103	

Groups: 9, complete: 3. Conditionally source elements: 6, target elements: 6 Close

The preview shows matching values, counts on both sides, and sample rooms and spaces.

The preview contains 9 groups, including 3 complete groups, with 6 source elements and 6 targets.

Groups 4, 5 and 6 contain only spaces; groups 104, 105 and 106 contain only rooms. These incomplete groups appear first and are skipped during processing.

Groups 101–103 contain one room and one space each.

The preview uses elements in the document or current view. In Process selected elements mode, its selection is broader than the subsequent rule execution.

Element filter

The element filter determines which elements are processed. It combines category and parameter filters. Depending on the strategy, one or both may be available.

The category filter specifies the permitted element categories.

The parameter filter defines one or more selection conditions. Conditions can be grouped and combined using AND or OR.

If both filters are available, an AND or OR operator also combines them:

- AND – the element must match both filters.
- OR – the element must match at least one filter.

Parameter filters can use element parameters, ID, TypeID and special parameters that describe element properties and relationships.

The [website help](#) explains filter settings, conditions and special parameters in detail.

	#	On	Target element	Grouping: Target element	Formula	Options
						
	1	☒	<div>Walls, Floors </div> <div>And </div> <div>Structural=1 AND ! Group </div>			T3 Aa  

Example – process walls and floors outside groups whose Structural parameter is checked.

Element grouping

Expressions normally evaluate each element separately. Grouping combines elements so GSum(), GCount(), GJoin() and other aggregate functions can calculate results over the entire group.

Elements whose grouping expressions return equal values belong to the same group.

For example, group ducts by type to calculate each type's total duct area and write it to every duct in that group.

	#	On	Target element	Grouping: Target elem	Formula	Options
✎ Total duct area by type						
⋮	1	☒	Categories: Ducts AND Parameters:	\$(Type)	\$(Comments)=GSum(\$(Area))	T3 Aa □ ↺

Double-click a cell in the Grouping column to open its settings.

 GROUPING: TARGET ELEMENT

✕

[Instruction](#)

Elements with the same value form one group

Group by:

\$(Type)|

🏷️

f

☐ Sort

☐ Descending

☐ Ignore case

Sort within the group by:

🏷️

f

Accept

Cancel

Group by contains a parameter or expression that determines group membership. Grouping and sorting expressions can use parameters and string or numeric functions.

An empty grouping key causes the element to be skipped; this is recorded in the report.

Grouping settings

GSum(), GCount(), GJoin(), GAvg(), GMin(), GMax() and GIndex() operate on the current group.

If aggregate functions are used without a grouping key, the rule's entire selection is treated as one group.

Ignore case combines string keys without distinguishing uppercase and lowercase.

Numeric keys are compared using the formula tolerance.

Grouping expressions do not support geometry, data-source functions or aggregate functions.

Sorting within a group

Enable Sort to order the elements in each group. Sort within the group by specifies the sorting parameter or expression. Descending reverses the order.

Sorting affects GIndex() values and the value order in GJoin().

Elements with empty sorting values appear at the end of the group, regardless of the sort direction.

If no separate sorting expression is defined, the grouping expression is used. Its value is equal for all members, so define a separate sorting expression to obtain a specific order.

For details, see Aggregate functions.

Formula Editor

FORMULA EDITOR

Instruction

Tolerance: 0.001

Case sensitive

Write empty result

Dissociate

1 \$[MP_Text]=Host · mark: · @[Mark]

2 \$[MP_Code]=#[Project · Number]_%[MP_Mark · prefix]

3 @[MP_Code]=Door · host

4 Join(@[MP_Door · marks] , ; , ds)=\$[Mark]

Fields

Expressions count: 4

Wrap text to new line

Accept

Cancel

A formula processes parameters of the target and the related (conditionally source) element within a rule.

A formula consists of expressions. Each expression is a separate operation entered on its own line.

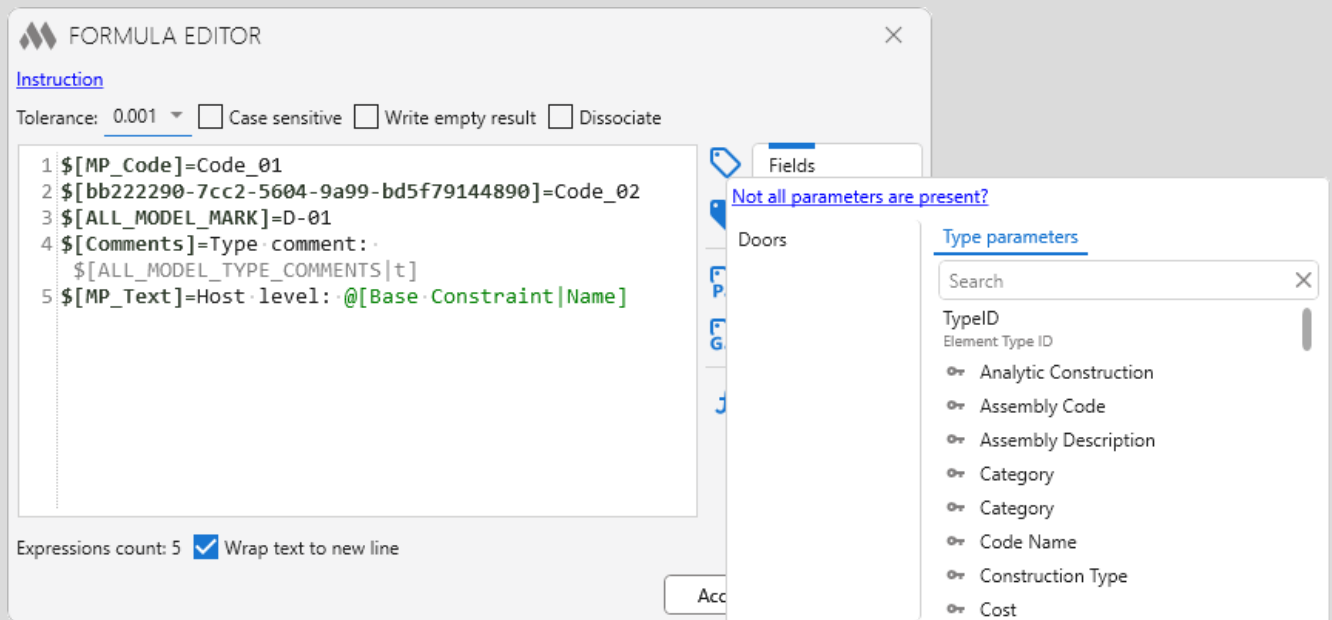
Parameters use identifiers in square brackets. The preceding symbol identifies the parameter's source.

Notation	Element
\$(parameter)	target element parameter
@(parameter)	source element parameter
#[parameter]	current Project Information parameter
%[parameter]	current project's global parameter

An expression can start with:

- a target parameter followed by “=”;
- a source parameter followed by “=”;
- a write-back function for a source element followed by “=”.

v.5.0 ModPlus™



Parameters in formulas

Enter a parameter name manually or select it from the Formula Editor's parameter list.

The list becomes available when a category filter is defined for the relevant element. Instance and type parameters appear on separate tabs.

A parameter can be identified by:

- its name;
- its shared parameter GUID;
- its built-in parameter identifier.

When selecting a parameter, click the key button to use its GUID or built-in identifier instead of its name.

If instance and type parameters share a name, the instance parameter is used by default. Append |t to explicitly access the type parameter.

If a parameter refers to another Revit element, use | to access that element's parameters. For example, Base Constraint refers to a Level.

FORMULA EDITOR

Instruction

Tolerance: 0.001

☐ Case sensitive

☐ Write empty result

☐ Dissociate

1 `$(MP_Text)=10+5`

2 `$(MP_Numeric.result)={10+5}`

3 `$(MP_Code)=Min(10·m³,25)`

4 `$(Comments)=Section·\{1\}\,·\{level`
`\}\:·\"AR\"\\nSecond·line`

Fields

+

★

Expressions count: 4

☒ Wrap text to new line

Accept

Cancel

By default, an expression result is processed as a string.

Enclose arithmetic expressions in braces, {}.

If a parameter is used:

- in a mathematical function, such as Min or Abs;
- in an arithmetic comparison (>, <, >=, <=);
- inside {};

its numeric portion is extracted, regardless of parameter data type. “10 m³” gives 10; “Level 1” gives 1.

Escape control characters with a preceding backslash when they must be treated as text. The separator in Join does not require escaping. Other characters requiring escaping are: () { } < > , ; " .

Use \n or \r\n to insert a line break into the written value.

Yes/No parameters accept these values:

State	Accepted values
On	1, true, yes
Off	0, false, no

v.5.0 ModPlus™

FORMULA EDITOR

×

[Instruction](#)

Tolerance: 0.001 ☐ Case sensitive ☐ Write empty result ☐ Dissociate

1 *// Doors marked SKIP are skipped*

2 `[$MP_Code]=If($[Mark]`

3 *// This line processes the*

4 `[$Comments]=Processed`

P

G

f

Fields

+

★

Expressions count: 2 ☒ Wrap text to new line

Accept

Cancel

SKIP_ELEMENT stops processing the current element when the specified condition is met.

Comments must be on a separate line and start with //.

A comparison operator can replace “=” after a target parameter. The expression is then marked Check only.

FORMULA EDITOR

×

[Instruction](#)

Tolerance: 0.001 ☐ Case sensitive ☐ Write empty result ☐ Dissociate

1 `[$MP_Numeric_result]>=0`

2 `[$MP_Flag]!=0`

P

G

f

Fields

+

★

Expressions count: 2 ☒ Wrap text to new line

Check only

Accept

Cancel

v.5.0 ModPlus™


FORMULA EDITOR
×

[Instruction](#)

Tolerance: 0.001 ☐ Case sensitive ☐ Write empty result ☐ Dissociate

```

1 Join(@[MP_Door·marks],,,ds)=$[Mark]
2 Sum(@[MP_Door·width·sum])=$[Width|t]
3 @[MP_Door·count]=GCount(t)

```







Fields

+
★

Expressions count: 3 ☒ Wrap text to new line

Accept

Cancel

Writing to a source element

For strategies with a source element, such as In Room, In Host and One To Many, target results can be written to source parameters in two ways.

1. Accumulated write-back collects results from multiple targets and writes one final value to a source parameter.

Use write-back functions such as Join(), Sum(), First(), Last(), Avg(), Min() and Max(). Values accumulate as targets are processed and are written after all qualifying targets have been processed.

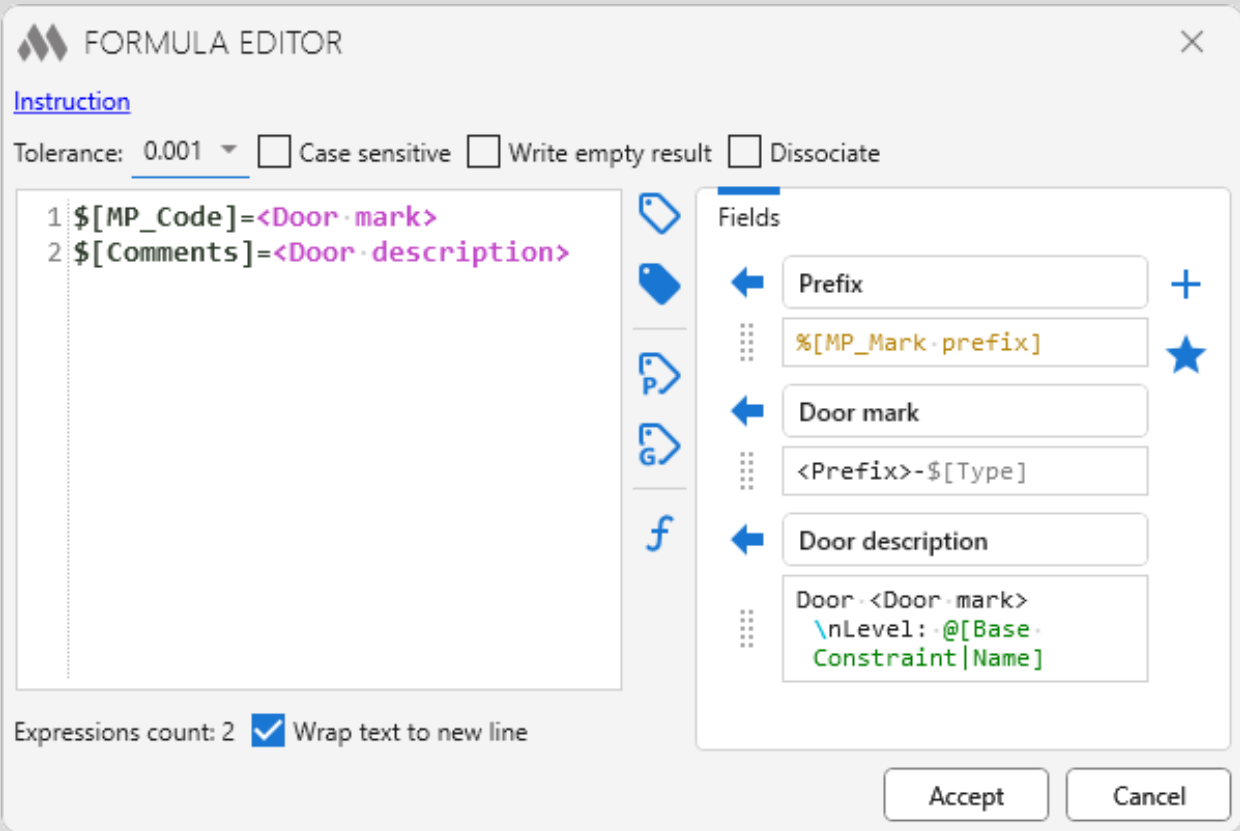
2. Direct writing requires no accumulation. An expression beginning @[parameter]=... writes its value as each element pair is processed.

If pairs produce different results, the source retains the last calculated value. Pair-processing order is undefined, so the final value cannot be predicted.

If the result may vary, use a write-back function such as First().

Repeating the same value with the same expression for the same element is skipped and omitted from the report.

Fields



Fields replace parts of a formula with keys, making long formulas easier to write and repeated parts reusable.

A field has a key and a value. Insert it as a key enclosed in angle brackets, <>. Field keys within a formula must be unique.

Field values can contain other field keys. A field must not refer to itself.

Save frequently used fields as Favorite fields. You can edit them and update their values in every rule where they were inserted.

Open Favorite fields from the plugin settings menu.

Functions

Expressions can use various functions.

The plugin provides these function types:

- mathematical functions
- logical functions
- modifying functions
- value retrieval functions
- aggregate functions
- write-back functions

Enter functions manually or select them using the function button.

Mapping function names in the plugin settings assigns custom names to Formula Editor functions. Write-back functions cannot be renamed.

 MAPPING FUNCTION NAMES

×

In this window, you can define your own names for the functions used in the formula editor. Function names must not contain spaces and control characters (brackets, commas, dots, etc.). Case is not important

Abs	<input type="text"/>	×
Acos	<input type="text"/>	×
And	<input type="text"/>	×
Asin	<input type="text"/>	×
Atan	<input type="text"/>	×
CcList	<input type="text"/>	×
CcMax	<input type="text"/>	×
CcMin	<input type="text"/>	×

Accept

Mathematical functions

Mathematical function arguments are evaluated as arithmetic expressions by default; braces are not required.

Min(x,y,...,n). Returns the smallest number.

`Min($[Volume],@[Volume],100)`

Max(x,y,...,n). Returns the largest number.

`Max($[Volume],@[Volume],100)`

Abs(x). Returns the absolute value.

`Abs(-10/$[Length])`

Sqrt(x). Returns the square root.

`Sqrt($[Length])`

Pow(x,y). Returns **x** raised to the power **y**.

`Pow($[Length],3)`

Sin(α). Returns the sine of an angle specified in degrees.

`Sin(60)`

Cos(α). Returns the cosine of an angle specified in degrees.

`Cos(60)`

Tan(α). Returns the tangent of an angle specified in degrees.

`Tan(60)`

Asin(x). Returns the angle in degrees whose sine equals the specified number.

`Asin($[Length])`

Acos(x). Returns the angle in degrees whose cosine equals the specified number.

`Acos($[Length])`

Atan(x). Returns the angle in degrees whose tangent equals the specified number.

`Atan($[Length])`

Log(x,y). Returns the logarithm of **x** to the specified base **y**.

`Log($[Length], 5)`

Ln(x). Returns the natural logarithm, with base e, of the specified number.

`Ln($[Length])`

Exp(x). Returns e raised to the specified power.

`Exp($[Length])`

Pi(). Returns the ratio of a circle's circumference to its diameter.

`10 * Pi()`

Mathematical functions. Rounding

Round(a1), Round(a1,a2) or Round(a1,a2,a3). Rounds to the nearest integer or the specified precision.

Two midpoint rounding conventions are available:

ToEven – midpoint values round to the nearest even number. Specify the letter e.

AwayFromZero – midpoint values round away from zero. Specify the letter z.

The function accepts one to three arguments:

·One argument (number) – rounds to the nearest integer, using ToEven for midpoint values.

$\text{Round}(\text{\$[Length]}) \longrightarrow \text{Round}(55.1438)=55$

Example in a strategy

·Two arguments (number, number) – rounds to the specified precision. An integer (1, 2, 3...) specifies decimal places. A fraction (0.1, 0.5, 0.01...) or an integer preceded by ! (!10, !50, !100...) specifies the rounding increment.

$\text{Round}(\text{\$[Length]},3) \longrightarrow \text{Round}(55.1438,3)=55.144$

$\text{Round}(\text{\$[Length]},!30) \longrightarrow \text{Round}(55.1438,!30)=60$

$\text{Round}(\text{\$[Length]},0.005) \longrightarrow \text{Round}(55.1438,0.005)=55.145$

Example in a strategy

·Two arguments (number, letter) – rounds to an integer using AwayFromZero (z) or ToEven (e).

Round(\$[Length],z) → Round(14.5,z)=15

Round(\$[Length],e) → Round(14.5,e)=14

·Three arguments (number, number, letter) – rounds to the precision specified by the second argument, using the convention specified by the third. An integer specifies decimal places; a fraction or an integer preceded by ! specifies a rounding increment. Use z for AwayFromZero or e for ToEven.

Round(\$[Length],3,z)

Round(\$[Length],!30,z)

Round(\$[Length],0.005,z)

RoundUp(x). Returns the smallest integer greater than or equal to x.

RoundUp(\$[Length]) → RoundUp(55.1438)=56

Example in a strategy

RoundDown(x). Returns the largest integer less than or equal to x.

RoundDown(\$[Length]) → RoundDown(55.1438)=55

Logical functions

Logical functions evaluate a condition and return a result based on whether it is true.

Conditions can use operator expressions or other logical functions that return TRUE or FALSE.

Operators supported in logical functions:

=	– equal to. Numeric or string values.
<> or !=	– not equal to. Numeric or string values.
>	– greater than. Numeric values only.
>=	– greater than or equal to. Numeric values only.
<	– less than. Numeric values only.
<=	– less than or equal to. Numeric values only.
~	– contains. String values only.
!~	– does not contain. String values only.

Starts with, Ends with, Does not start with and Does not end with conditions are also supported.

For Starts with, use Contains and append * to the second argument.

`$(Level) ~ Ypo*`

For Ends with, use Contains and prepend * to the second argument.

`$(Level) ~ *Ypo`

For Does not start with, use Does not contain and append * to the second argument.

`$(Level) !~ Ypo*`

For Does not end with, use Does not contain and prepend * to the second argument.

`$(Level) !~ *Ypo`

Logical functions

Functions that return a value according to a condition:

If(c,true,false). Returns <true> when <c> is true; otherwise returns <false>.

`If($[Length]>100,Длинное,Короткое)`

Example in a strategy

FirstTrue(c1:v1;c2:v2;...;cN:vN;v). Returns the value for the first true condition. If no condition is true, returns the final fallback value <v>.

`FirstTrue($[Length]>100:A;And($[Length]>50,$[Height]>50):B;C)`

Example in a strategy 1

Example in a strategy 2

Functions usable as conditions in other logical functions:

And(c1,c2,...,cn). Returns TRUE if every condition is true; otherwise returns FALSE.

`If(And($[Length]>100,$[Height]>100),Большое,Малое)`

Or(c1,c2,...,cn). Returns TRUE if any condition is true; otherwise returns FALSE.

`If(Or($[Length]>100,$[Height]>100),Большое,Малое)`

Example in a strategy

Not(c). Returns TRUE if <c> is false, and FALSE if <c> is true. The condition can be any function that returns TRUE or FALSE.

`Not(TMirrored())`

InRange(x,min..max). Returns TRUE if <x> is greater than or equal to <min> and less than or equal to <max>; otherwise returns FALSE.

And(InRange(\$[Length],1000..5000),InRange (\$[Height], 1000..5000))

Example in a strategy

Has(p). Returns TRUE if parameter <p> exists; otherwise returns FALSE. Use it as a condition in other logical functions. It supports source @[], target \$[], Project Information #[] and global %[] parameters.

If(Has(\$[Length]),да,нет)

HasValue(p). Returns TRUE if <p> has ever been assigned a value; otherwise returns FALSE. Use it in other logical functions. It supports source @[], target \$[] and Project Information #[] parameters.

If(HasValue(\$[Length]),да,нет)

Example in a strategy

TMirrored(). Returns TRUE if the target is a mirrored family instance; otherwise returns FALSE.

If(TMirrored(),Отзеркален,Ок)

Example in a strategy

SMirrored(). Returns TRUE if the source is a mirrored family instance; otherwise returns FALSE.

If(SMirrored(),В отзеркаленном,Ок)

IsHorizontal() or **IsHorizontal(a)**. Returns TRUE if the target's location line is horizontal within tolerance <a>; otherwise returns FALSE. The tolerance is an angle from 0° to 90° between the line and the horizontal plane. The default is 0.1°. Specify a larger tolerance to include sloped pipes. Element restrictions are the same as for IsVertical.

The function can be used as a condition in other logical functions.

Example in a strategy

IsVertical() or **IsVertical(a)**. Returns TRUE if the target's location line is vertical within tolerance <a>; otherwise returns FALSE. The tolerance is an angle from 0° to 90° away from vertical. The default, 0.1°, accommodates calculation inaccuracies in an aligned riser while rejecting an intentional slope.

Only elements located by a straight line are supported: pipes, ducts, cable trays, conduits, beams and slanted columns. Point-based elements and curved or flexible location lines produce an error. The function can be used in other logical conditions.

Example in a strategy

Modifying functions

ToUpper(s). Returns the string in uppercase.

ToUpper(\$[Level]) → ToUpper(Level 06)=LEVEL 06

ToLower(s). Returns the string in lowercase.

ToLower(\$[Level]) → ToLower(Level 06)=level 06

Example in a strategy

LStr(s,l). Returns the first <l> characters of the string.

LStr(\$[Level],5) → LStr(Level 06,5)=Level

Example in a strategy

RStr(s,l). Returns the last <l> characters of the string.

RStr (\$[Level],3) → RStr (Level 06,3)= 06

SStr(s,i,l). Returns a substring.

The function accepts two or three arguments:

·Two arguments (string, number) – returns text from index <i> to the end of the string. The first character is at index 0.

SStr (\$[Level],2) → SStr (Level 06,2)=vel 06

Example in a strategy

·Three arguments (string, number, number) – returns <l> characters starting at index <i>. The first character is at index 0.

SStr (\$[Level],2,2) → SStr (Этаж 06,2,2)=el

Example in a strategy

Replace(s,s1,s2). Replaces every occurrence of <s1> in <s> with <s2>. If <s1> is absent, returns <s> unchanged. All arguments are treated as strings. Matching is case-sensitive.

Replace(Test,s,x) = Text
Replace(\$[Comments],1,5)
Replace(level 1,1,5) = level 5

Format(x,f). Converts <x> to a string using format <f>. Supported formats:
Standard formats:

- D or d. Formats an integer as decimal digits (0–9), preceded by a minus sign if negative. Fractional input is truncated to an integer. The precision specifies the minimum digit count; missing digits are padded with zeros. Without a precision, the number uses the minimum required digits with no leading zeros.

Format(123,d) → 123
Format (123,d5) → 00123
Format(123.3,d) → 123

Example in a strategy

- F or f. Formats a number as “-ddd.ddd...”, where d is a digit. Negative numbers start with a minus sign. Precision specifies decimal places; if omitted, the operating system's current numeric precision is used. The plugin settings determine the decimal separator.

Format(15,f2) → 15.00
Format(15.5,f3) → 15.500

Example in a strategy

Custom formats:

- 0. Inserts the corresponding digit, or zero if no digit exists. The number of zeros after the decimal separator determines the rounding precision. The plugin settings determine the decimal separator.

Format(15,0.00) → 15.00
Format(15.455,0.00) → 15.46
Format(15.455, 000.0) → 015.5

Distinct(s1,s2). Returns unique items from list <s1>, using each character of <s2> as a separate separator. Leading and trailing spaces are ignored when comparing items. Matching is case-sensitive.

Distinct(\$[Comments],\,)

FindItem(s,d,t). Returns the first item in list <s> containing <t>. The entire <d> string is the separator. Leading and trailing spaces are removed from the items and search text. If no item matches, returns an empty value. Matching is case-insensitive.

Example in a strategy

Item(s,d,i). Returns item <i> from list <s>. The entire <d> string is the separator: "A; B" with separator ";" produces two items. Indices start at 0. Leading and trailing spaces are removed. If the index is absent, returns an empty value.

Example in a strategy

List(s,d). Converts <s> to a list for membership checks using =| or !=|. The entire <d> string is the separator. Leading and trailing spaces and empty items are removed. The input may be a parameter value or a literal list written directly in the expression.

Example in a strategy

Trim(s). Removes leading and trailing spaces from <s>, preserving internal spaces.

Example in a strategy

RgxReplace(s,"s1",s2)

Replaces every match of regular expression <s1> in <s> with <s2>. If nothing matches, returns <s> unchanged. The pattern must be double-quoted, obtained from a parameter or returned by a nested function. An unquoted pattern is parsed as formula syntax: parentheses delimit functions, braces delimit arithmetic, commas separate arguments and backslashes escape characters.

Replacement <s2> supports captured groups such as "\$1" and "\$2". To insert a dollar sign, use "\$\$". Case sensitivity follows the formula setting.

Example in a strategy

Ranges(s,d)

Converts consecutive integers in <s> into ranges. <d> separates both input and output items. Integers are sorted ascending and deduplicated. A range requires at least three consecutive numbers. Values that are neither integers nor compound numbers remain unchanged and are appended to the result.

Dot-separated compound numbers are also compressed when all parts except the last match and the final parts are consecutive.

"2.1.1.1,2.1.1.2,2.1.1.3" becomes "2.1.1.1-2.1.1.3". Compound numbers are sorted by their parts and follow integers. With separator ",", "1,2,3,5,6,7,10" becomes "1-3,5-7,10".

Example in a strategy

Value retrieval functions

StrLen(s). Returns the string length.

StrLen(\$[Comments]) → StrLen(Этаж 06)=7

Example in a strategy

StrIndex(s1,s2). Returns the zero-based index of the first occurrence of <s2> in <s1>, or -1 if absent. Matching is case-insensitive.

StrIndex(\$[Comments],_) → StrIndex(Тип_A_15,_) = 3

Example in a strategy

LStrIndex(s1,s2). Returns the zero-based index of the last occurrence of <s2> in <s1>, or -1 if absent. Matching is case-insensitive.

LStrIndex(\$[Comments],_) → LStrIndex(Тип_A_15,_) = 5

Example in a strategy

Count(s1,s2). Counts matches of regular expression <s2> in <s1>, returning 0 if none exist. Escape regex metacharacters to count them literally. Matching is always case-insensitive.

Count(\$[Comments],_) → Count(Тип_A_15,_) = 2

ValueStr(p). Returns a parameter value as a string with units. Its single argument must reference a target \$[] or source @[] parameter. This is useful only for parameters whose units appear in the standard Properties palette.

\$[Mark]=ValueStr(\$[Диаметр]) → \$[Mark]=150 mm

EIndex(i). Returns the successful-use counter for the current expression. Each element matching the rule's filters is processed. The counter increases by 1 when a parameter value is successfully set. An optional numeric argument offsets the index: with 5, the first returned value is 6.

\$[Mark]=C-EIndex() → \$[Mark]=C-1

\$[Mark]=C-EIndex(5) → \$[Mark]=C-6

Example in a strategy

CV(cell). Returns a value from the strategy's Excel data source. Specify a cell address such as A1 or \$A\$1. The returned value is not validated: control characters or operators in the cell may cause later expression evaluation errors.

CV(A1)

Example:

Data source:File: Sheet:

Pre-filter:Target element

Filter by category

And

Filter by parameters

Strategy filter:Target element

Filter by category

And

Filter by parameters

Rules

	#	On	Target element	Grouping: Target	Formula	Options
Read specific cells						
	1	☒	Categories: Doors AND Parameters:		\$(Comments)=CV(A2) \$(MP_Code)=CV(B2)	T3 Aa <input type="text"/>

Formula:
\$(Comments)=CV(A2)
\$(MP_Code)=CV(B2)

Values are read from cells A2 and B2 on sheet “2.” of the Excel workbook “Product.xlsx”.

CVIf(c1,c2,...,cN,r). Finds the first Excel row meeting all conditions <cN> and returns column <r>. Use column letters or numbers on the left of each condition and for <r>. Conditions follow the rules of logical functions. Returned values are not validated: control characters or operators may cause evaluation errors.

CVIf(A=5,B=10,C)CVIf(1~текст,2!=15,5)

Example:

Data source:

File:

Product.xlsx

Sheet:

2.

Pre-filter:

Target element

Filter by category

And

Filter by parameters

Strategy filter:

Target element

Filter by category

And

Filter by parameters

Rules

	#	On	Target element	Grouping: Targe	Formula	Options
Match all conditions						
	1	☒	Categories: Doors AND Parameters:		<div><div>[\$[Comments]=CVIf(C=SD,F>=900,A)]</div><div>[\$[MP_Code]=CVIf(C=SD,F>=900,B)]</div></div>	T3 Aa

Formula:
\$[Comments]=CVIf(C=SD,F>=900,A)
\$[MP_Code]=CVIf(C=SD,F>=900,B)

Values come from columns A and B in the first row on sheet “2.” of “Product.xlsx” where both C=SD and F>=900 are true.

CVOr(c1,c2,...,cN,r). Finds the first Excel row meeting at least one condition <cN> and returns column <r>. Use column letters or numbers on the left of each condition and for <r>. Conditions follow the rules of logical functions. Returned values are not validated: control characters or operators may cause evaluation errors.

CVOr(A=5,B=10,C)CVOr(1~текст,2!=15,5)

Example:

Data source:File: Product.xlsxSheet: 2.

Pre-filter:Target elementFilter by categoryAndFilter by parameters

Strategy filter:Target elementFilter by categoryAndFilter by parameters

Rules

	#	On	Target element	Grouping: Targe	Formula	Options
Match any condition						
	1	☒	Categories: Doors AND Parameters:		[\$Comments]=CVOr(B=[\$MP_Code],...	T3 Aa

Formula:
[\$Comments]=CVOr(B=[\$MP_Code],D=[\$MP_Code],A)

The value comes from column A in the first row on sheet “2.” of “Product.xlsx” where the door code matches the primary code in B or the alternative code in D.

Example in a strategy

CCList(c,r1..r2)

Returns column <c> from the Excel data source as a list for =| and !=| membership checks. Specify the column by letter or number. Optional <r1..r2> limits the row range, with row numbers separated by two dots. Empty cells are omitted.

Example in a strategy

CCMin(c,r1..r2)

Returns the minimum numeric value in Excel column <c>, specified by letter or number. Optional <r1..r2> limits the row range. Headers, labels, empty cells and nonnumeric values are ignored. If no numeric cells exist, returns an empty value.

Example in a strategy

CCMax(c,r1..r2)

Returns the maximum numeric value in Excel column <c>, specified by letter or number. Optional <r1..r2> limits the row range. Headers, labels, empty cells and nonnumeric values are ignored. If no numeric cells exist, returns an empty value.

Example in a strategy

CRList(r,c1..c2)

Returns row <r> from the Excel data source as a list for =| and !=| membership checks. Specify the row by number. Optional <c1..c2> limits the columns, using letters or numbers separated by two dots. Empty cells are omitted.

Example in a strategy

CRMin(r,c1..c2). Returns the minimum numeric value in Excel row <r>. Optional <c1..c2> limits the columns, specified by letters or numbers separated by two dots. Headers, labels, empty cells and nonnumeric values are ignored. If no numeric cells exist, returns an empty value.

Example in a strategy

CRMax(r,c1..c2). Returns the maximum numeric value in Excel row <r>. Optional <c1..c2> limits the columns, specified by letters or numbers separated by two dots. Headers, labels, empty cells and nonnumeric values are ignored. If no numeric cells exist, returns an empty value.

Example in a strategy

GetCurrentViewParameter(p). Returns <p> from the view being processed. Identify the parameter by name, shared GUID or built-in identifier, as for \$[] and @[]. Available for current-view, selected-view and selected-element processing. Document-wide processing has no view context and produces an error.

A missing view parameter also produces an error.

Example in a strategy

ItemIndex(s,d,t). Returns the zero-based index of the first list item containing <t>. The entire <d> string is the separator. Leading and trailing spaces are removed from items and search text. If no item matches, returns -1. Matching is case-insensitive.

Example in a strategy

LStrIndex(s1,s2). Returns the zero-based index of the last occurrence of <s2> in <s1>, or -1 if absent. Matching is case-insensitive.

Example in a strategy

SourceBottomZ(). Returns the lowest Z elevation of the source geometry, in millimeters rounded to three decimal places.

SourceTopZ(). Returns the highest Z elevation of the source geometry, in millimeters rounded to three decimal places. Unless the entire document is processed, geometry is taken from the active view and limited by its cut plane. On a floor plan, a wall's result may therefore be the cut-plane elevation rather than its actual top. Use Process elements in document for full geometry.

Linked-file elements always use model geometry.

Example in a strategy

TargetBottomZ(). Returns the lowest Z elevation of the target geometry, in millimeters rounded to three decimal places.

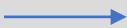
Example in a strategy

TargetTopZ()

Returns the highest Z elevation of the target geometry, in millimeters rounded to three decimal places. Unless the entire document is processed, geometry is taken from the active view and limited by its cut plane. On a floor plan, a wall's result may be the cut-plane elevation rather than its actual top. Use Process elements in document for full geometry.

Example in a strategy

DocTitle(). Returns the current document title, corresponding to its file name without the extension. An unsaved document returns its default title; a detached document may return an empty value.

`$[Mark]=If(DocTitle()~КЖ,Да,Нет)`  `$[Mark]=да`

Example in a strategy

Aggregate functions

GAvg(e). Returns the average of <e> over the group. An empty group returns an empty string.

```
$[Comments]=GAvg($[Area])
```

Example in a strategy

GCount(r). Returns the group element count. Role <r> is s for distinct source elements or t for distinct targets. An empty argument is permitted only in strategies without source elements.

```
$[Comments]= GCount(t)
```

Example in a strategy

GIndex() or **GIndex(t).** Returns the target's position in its group, starting at 1 and following the grouping sort order. The only permitted optional argument is t: the index always refers to a target.

```
$[Number]=GIndex()
```

Example in a strategy

GJoin(e,s,f). Combines group values of <e> using separator <s>. Optional flags <f> are d to remove duplicates, s to sort values and r to compress consecutive integers into ranges. Flags can be combined, for example ds. Unlike GCount, s here means sorting, not the source-element role.

```
$[Comments]=GJoin(@[Name],, )  
$[Comments]=GJoin($[Mark],\,,ds)  
$[Comments]=GJoin($[Number],\,,r)
```

Example in a strategy

GMax(e). Returns the maximum of <e> over the group. An empty group returns an empty string.

```
$[Comments]=GMax($[Length])
```

Example in a strategy

GMin(e). Returns the minimum of <e> over the group. An empty group returns an empty string.

```
$[Comments]=GMin($[Length])
```

Example in a strategy

GSum(e). Returns the sum of <e> over the group. An empty group returns 0, so the function can be used in arithmetic without an additional check.

`$[Comments]=GSum($[Area])`

`$[Доля]={$[Volume] / GSum($[Volume])}`

Example in a strategy

Write-back functions

Avg(p). After all qualifying targets are processed, writes the average numeric result to source parameter <p>, which may be numeric or text. If any result is nonnumeric, nothing is written.

Avg(@[Comments])=\$[Length]

First(p,s). After all qualifying targets are processed, writes the first accumulated result to source parameter <p>.

One or two arguments are permitted. Target-processing order is undefined, so without the second argument the chosen result is unpredictable. Optional s sorts the accumulated results before selection, as in Join().

First(@[Comments],s)=\$[Comments]

Example in a strategy

Join(p,d,f). Collects expression results into one string using separator <d>, then writes it to source text parameter <p> after all qualifying targets are processed. The separator does not require control-character escaping. Optional flags <f>: d removes duplicates, s sorts alphabetically and ds applies both.

Join(@[Comments],)= \$[Mark]
Join(@[Comments],,)= \$[Mark]
Join(@[Comments],,d)= \$[Mark]
Join(@[Comments],, ,d)= \$[Mark]
Join(@[Comments],,s)= \$[Mark]
Join(@[Comments],, ,ds)= \$[Mark]

Example in a strategy 1

Example in a strategy 2

Example in a strategy 3

Last(p,s). After all qualifying targets are processed, writes the last accumulated result to source parameter <p>.

One or two arguments are permitted. Target-processing order is undefined, so without the second argument the chosen result is unpredictable. Optional s sorts the accumulated results before selection, as in Join().

Last(@[Comments],s)=\$[Comments]

Example in a strategy 1

Max(p). After all qualifying targets are processed, writes the maximum numeric result to source parameter <p>, which may be numeric or text. If any result is nonnumeric, nothing is written.

Max(@[Comments])=\$[Length]

Min(p). After all qualifying targets are processed, writes the minimum numeric result to source parameter <p>, which may be numeric or text. If any result is nonnumeric, nothing is written.

Min(@[Comments])=\$[Length]

Sum(p). After all qualifying targets are processed, writes the sum of numeric results to source parameter <p>, which may be numeric or text. If any result is nonnumeric or a target parameter is empty, nothing is written.

Sum(@[Comments])=\$[Length]

Example in a strategy 1

Example in a strategy 2

Checking parameter values

Process elements in Document ▾

☒ Processing elements from linked files
☐ Select linked files

 Execute

 Check

Starting a check

Click Check to compare current parameter values with calculated formula results.

For $\$(parameter)=...$, the current value is compared with the calculated right-hand side.

When the check is complete, a results window opens, similar to Parametrization Results.

Check-only expressions

Replace “=” after the target parameter with a comparison operator to mark the expression Check only.

The operator defines the condition against which the parameter is checked.

Check operators

An expression contains a target parameter, a check operator and a right-hand value or list.

Use $=|$ to test list membership, or $!=|$ to test that a value is absent from the list.

Checking against lists

The right side of $=|$ or $!=|$ contains a list of permitted or excluded values.

List constructs a list of values. CCList reads an Excel column; CRList reads a row.

Select the Excel data source in the strategy settings.

Check results

Results are organized by strategy, rule and expression. Messages comparing values show both the current and calculated parameter values.

Expand or collapse the results. The selected entry displays the IDs of the relevant elements.

Select all and Isolate all locate the elements in the model. Results can be exported to txt or Excel.

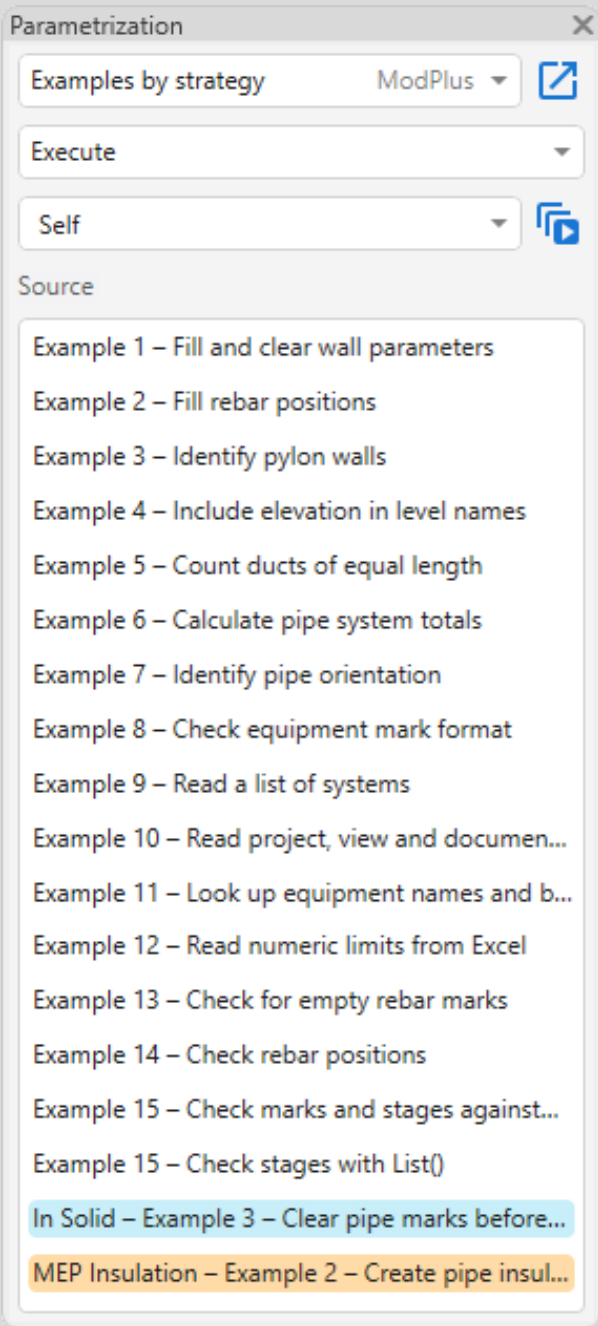
Examples in Self

Example 13 – find rebar with an empty Mark.

Example 14 – compare “MP_Position” with its calculated value.

The formulas and results windows appear alongside these examples.

Palette



The palette is a compact window for quickly running prepared strategies and rules.

Show or hide it using the button at the upper right of the main Parametrization window.

Select the configuration and the Execute or Check mode at the top. Then select a strategy.

Only strategies and rules checked in the main plugin window appear in the palette.

Click a rule's name to run that rule.

The button to the right of the strategy list runs every checked rule in the selected strategy.

Hover over a rule name to display its tooltip. It shows the rule number and name, element filters, formula and options, and whether the formula contains Check-only expressions.