REFERENCE MANUAL FOR COMBINUM ARCHITECT

THE MODELLING TOOL OF COMBINUM CPQ

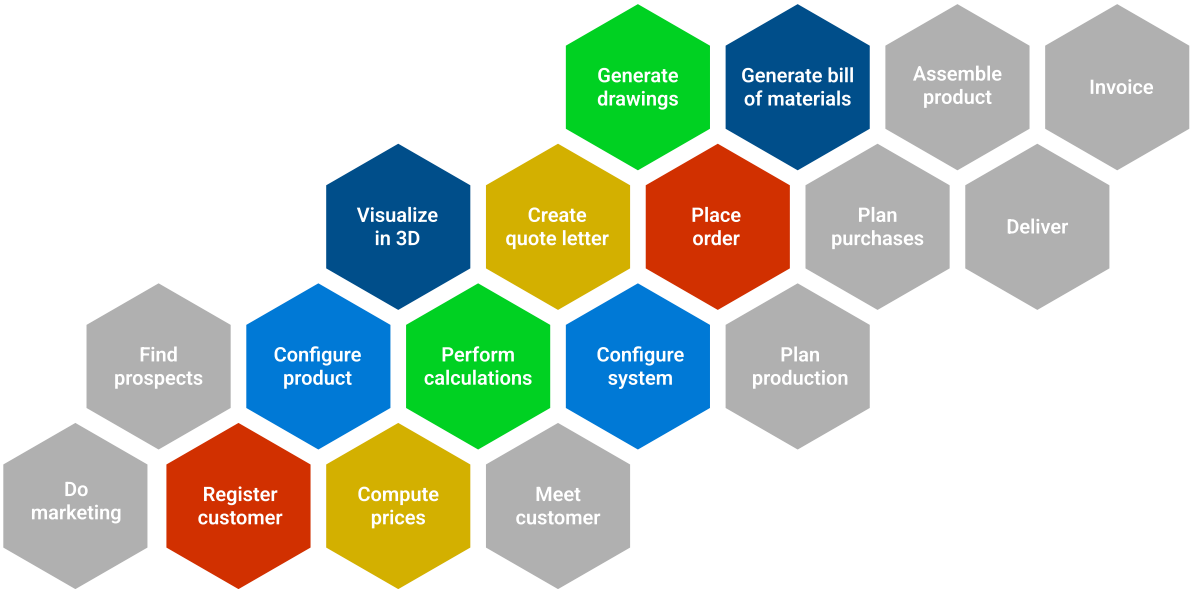

TABLE OF CONTENTS

1. WELCOME

Reference manual for Combinum Architect

Here you find the reference manual for Combinum Architect which is the modelling tool of Combinum CPQ (Configure, Price, Quote).

Full release 2025-Q3

The traditional help file of Combinum Architect is currently being converted and improved to an online help system. The full online help is scheduled for release in Q3 2025.

As a teaser we have provided the chapter about expressions and expression functions.

2. EXPRESSIONS

In Combinum, you can write rules and formulas to perform calculations, trigger events and compose texts. This is done using expressions.
Expressions are used in all parts of Combinum, for example to constrain parameters and dynamically generate bill of materials, price lists, graphics, and reports.

An expression can consist of operators, functions, variables, and constants. The variables can be parameters and attributes.

Examples

```
a * (b + 1)
```

Performs a calculation using the variables a and b that can be either parameters or attributes.

```
Model = Model.[X]
```

Returns true if the parameter Model has the value X.

```
City[SelectedCity; City.Latitude]
```

Looks up the latitude of a city in a dictionary (table) based on the variable SelectedCity.

```
ToString(Now();"dd/MM/yyyy")
```

Gets the current date and time and converts this to a string that is formatted as dd/MM/yyyy.

Remarks

The single equal sign "=" is used for comparison of two values to determine if they are equal, i.e., not for setting a variable. Setting parameters and attributes is done in other ways.

2.1. DATA TYPES

The following table lists the built-in data types of the expression evaluation engine:

c# type	Description
bool	Represents a Boolean value, which can be either true or false.
DateTime*	Represents an instant in time, typically expressed as a date and time of day.
double	Represents real numbers with a precision of 15-17 digits.
int	Represents integer numbers between -2 147 483 648 and 2 147 483 647.
string	Represents a sequence of zero or more Unicode characters.

* Date times can be managed by properties and inside expressions in Combinum, but parameters and attributes can only be defined using the value types bool, double, int and string.

2.2. OPERATORS

The following table lists the built-in operators of the expression evaluation engine:

Operator	Name	Description
Primary		
x.[y]	Parameter value	Value y of the parameter x.
d.c	Dictionary column	Column c of the dictionary d.
d["Key", d.c]	Dictionary value	Value in the column c at the row with the key "key" of the dictionary d.
Unary		
-	Numerical negation	Numerical negation of the next value.
!	Logical negation	Logical negation of the next Boolean value.
Multiplicative		
*	Multiplication	Multiplication of two values.
/	Division	Floating-point division of two values.

Operator	Name	Description
div	Integer division	Integer division of two values,
%	Remainder	The remainder from the integer division of two values.
Additive		
+	Plus	Addition of two values.
-	Minus	Subtraction of the second value from the first value.
Comparison		
=	Equal	True if the first and second values are equal, else false.
!=	Not equal	True if the first and second values are not equal, else false.
<	Less than	True if the first value is less than the second value, else false.
>	Greater than	True if the first value is greater than the second value, else false.
<=	Less than or equal	True if the first value is less than or equal to the second value, else false.
>=	Greater than or equal	True if the first value is greater than or equal to the second value, else false.
Conditional		
&&	And	True if both first and second value are true, else false.
||	Or	True if first, second or both values are true, else false.
XOR	Exclusive or	True if either first or second value is is true, else false. If both values are true, then false is returned.

Operator	Name	Description
Ternary		
?:	If then or else	If the value before "?" is true then the value after "?" is returned, else the value after ":" is returned.

The order of the operators groups above indicates the precedence of the operators. In an expression with multiple operators, the operators with higher precedence are evaluated before the operators with lower precedence. Use parentheses to change the order of evaluation.

2.3. FUNCTIONS

Below are the standard expression functions of Combinum grouped by type. Additional custom expression functions can be added using the C# editor found in the Tools menu.

2.3.1. ATTRIBUTES

Expression functions for reading and counting attribute values.

Remarks

It's a good habit to put attributes that read values of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace the referenced attribute with a parameter that is set using a template expression rule.

2.3.1.1. ACTIVEATTRIBUTE

Returns a property value of active attribute during an iteration of the attributes of a configuration.

```
string ActiveAttribute(string propertyName)
```

propertyName	The name of a property to read. The supported properties are "IsWarning", "Name", "Title", "Type", "Unit" and "Value".

Example

```
ActiveAttribute("Value")
```

Returns the value of active attribute converted to a string.
Note! Boolean values are returned as "Yes" or "No" for display reasons.

2.3.1.2. ATTRIBUTEOCC

Returns the number of attributes that matches a predicate expression within the active configuration. The predicate expression is defined using the ActiveAttribute() function.

```
int AttributeOcc(bool predicateExpression)
```

predicateExpression	The expression to evaluate for each attribute to decide if it should be counted or not.

Example

```
AttributeOcc(SubStr(ActiveAttribute("Name"); 0; 6) = "Weight" &&
ToDouble(ActiveAttribute("Value")) > 1000)
```

Returns the number of attributes whose name starts with "Weight" and whose value is greater than 1000).

2.3.1.3. CHILD

Returns the product title or the value of a named parameter, multi-choice value or attribute in the specified child configuration. If no matching child or parameter/attribute is found, then an empty string is returned.

```
string Child(int index; [string paramOrAttrName]; [string productTi-
tle])
```

index	The zero-based index of the configuration to get a value from.
paramOrAttrName	Optional name of a parameter, multi-choice value or attribute.
productTitle	Optional filter to only include child configurations of the specified product.

Example

```
Child(2; "Color")
```

Returns the value of the parameter or attribute named "Color" from the third child configuration.

```
Child(0; "Options.FastDelivery")
```

Returns the value of the multi-choice value named "FastDelivery" belonging to the multi-choice parameter "Options" from the first child configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.4. CHILDRENATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the child configurations.

```
int ChildrenAttributeOcc(string name; string value)
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
ChildrenAttributeOcc("SparePart", "True")
```

Returns the number of children for which the Boolean attribute "SpartPart" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.5. CHILDRENATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the child configurations.

```
double ChildrenAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildrenAttributeSum("Weight")
```

Returns the child configurations' sum of the double or integer attribute "Weight".

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.6. DYNCONFIGATTRIBUTE

Returns the attribute value of a dynamic configuration, i.e., from a configuration that has been auto generated from a dynamic instance.

```
string DynConfigAttribute(string productTitle; string instanceName;
int instanceIndex; string attributeName)
```

productTitle	The title of the product of the dynamic configuration.
instanceName	The name of the dynamic instance driving the dynamic configuration.

instanceIndex	The zero-based index of the dynamic configuration. Use zero for non-indexed instances.
attributeName	The name of the attribute to read.

Example

```
DynConfigAttribute("Wall block"; "Wall 1"; 3; "Weight")
```

Returns the value of the attribute "Weight" from the fourth dynamic "Wall block" configuration that is generated from the dynamic instance "Wall 1".

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.7. ISSET

Returns true if the specified parameter, attribute, multi-choice value or dictionary cell is set, otherwise false. If no matching parameter, attribute, multi-choice value or dictionary is found false is returned.

Overloads

```
bool IsSet(string name; [string dictKey]; [string dictColNameOrIndex])
bool IsSet(Parameter parameter)
bool IsSet(Attribute attribute)
bool IsSet(MultiChoiceValue multiChoiceValue)
```

name	The name of a parameter, attribute, multi-choice value or dictionary.
dictKey	Optionally the key of a dictionary row.
dictColNameOrIndex	Optionally the name or index of a dictionary column.

Example

```
IsSet("Model")
```

Returns true if a parameter or attribute with the name "Model" is set, otherwise false.

2.3.1.8. MAXCHILDVALUE

Returns the max value of the specified numerical parameter or attribute among the child configurations.

```
double MaxChildValue(string name)
```

name	The name of a parameter or attribute.

Example

`MaxChildValue("LeadTime")`

Returns the max value of the parameter or attribute "LeadTime" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.9. MINCHILDVALUE

Returns the min value of the specified numerical parameter or attribute among the child configurations.

```
double MinChildValue(string name)
```

name	The name of a parameter or attribute.

Example

`MinChildValue("InventoryBalance")`

Returns the min value of the parameter or attribute "`InventoryBalance`" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.10. NEXTSIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations with a higher index.

```
int NextSiblingsAttributeOcc(string name; string value)
```

name	The name of an attribute.

value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
NextSiblingsAttributeOcc("Connected", "True")
```

Returns the number of sibling configurations with a higher index for which the Boolean attribute "Connected" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.11. NEXTSIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations with a higher index.

```
double NextSiblingsAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
NextSiblingsAttributeSum("Weight")
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations with a higher index.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.12. PARENT

Returns the product title or the value of a named parameter, multi-choice value or attribute of the parent configuration. If no parent configuration or no matching parameter/attribute is found, then an empty string is returned.

```
string Parent([string name])
```

name	Optional name of a parameter, multi-choice value or attribute.

Example

`Parent()`

Returns the title of the parent configuration's product.

`Parent("Color")`

Returns the value of the parameter or attribute named "Color" from the parent configuration.

`Parent("Options.FastDelivery")`

Returns the value of the multi-choice value named "FastDelivery", belonging to the multi-choice parameter "Options", from the parent configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.13. PREVSIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations with a lower index.

```
int PrevSiblingsAttributeOcc(string name; string value)
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

`PrevSiblingsAttributeOcc("Connected", "True")`

Returns the number of sibling configurations with a lower index for which the Boolean attribute "Connected" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.14. PREVSIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations with a lower index.

```
double PrevSiblingsAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
PrevSiblingsAttributeSum("Weight")
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations with a lower index.

Remarks

It's a good habit to put attributes that read values of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies.

2.3.1.15. SIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations.

```
int SiblingsAttributeOcc(string name; string value; [bool in-
cludeSelf])
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".
includeSelf	Optional argument to include current configuration in the attribute count. Default is false.

Example

```
SiblingsAttributeOcc("Connected", "True")
```

Returns the number of sibling configurations for which the Boolean attribute "Connected" is true. Current configuration is not included.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.1.16. SIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations.

```
double SiblingsAttributeSum(string name; [bool includeSelf]; [string productTitle])
```

name	The name of an attribute.
includeSelf	Optional argument to include current configuration in the attribute sum. Default is false.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
SiblingsAttributeSum("Weight"; true)
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations including current configuration.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.2. BILL OF MATERIALS

Expression functions for reading data from generated bill of materials and for controlling how bill of materials are generated.

ActiveItem
BomCost

BomPrice
BomWeight
InStructureMode

2.3.2.1. ACTIVEITEM

Returns a fixed or calculated property of current parent item, current item structure or current child item when looping through a bill of materials using a repeater.

```
string ActiveItem(string target; string propertyName; [string prop-
ertyName2]; ...)
```

target	The object to read a property from. The supported values are "Parent", "Structure" and "Child" (case insensitive).
propertyName	Name of a property or a type of calculated data to read. See list of arguments for calculated data below.
propertyName2	Optional name of a property to read from an object that is referenced by the first property or an extra argument that is required by the type of calculated data.

Arguments for calculated data

The following arguments for calculated data are supported (case insensitive):

"Attribute"	Returns the name of a structure attribute that has been set for current structure node. A third argument with a zero based index is required.
"AttributeCount"	Returns the number of structure attributes that have been set for current structure node.
"AttributeUnit"	Returns the unit of a structure attribute that has been set for current structure node. A third argument with either the name of a structure attribute or a zero based index is required.
"AttributeValue"	Returns the calculated value of a structure attribute that has been set for this structure node. A third argument with either the name of a structure attribute or a zero based index is required.

"Cost"	Returns the calculated cost of current child item.
"Level"	Returns the child item's level in the bill of materials at the position given by current structure node. The root item is at level 0.
"Price"	Returns the calculated price of current child item.

Examples

`ActiveItem("Child"; "ItemNo")`

Returns the item number of the child item.

`ActiveItem("Structure"; "Quantity")`

Returns the (calculated) quantity of the child item in the structure of the parent item.

`ActiveItem("Structure"; "Level")`

Returns the child item's level in the bill of materials at the position given by current structure node.

`ActiveItem("Parent"; "ItemNo")`
Returns the item number of the parent item.

`ActiveItem("Structure"; "AttributeValue"; "Length")`

Returns the calculated value of the structure attribute "Length".

`ActiveItem("Child"; "Cost")`

Returns the calculated cost of the child item.

Remarks

Some structure properties like context, operation sequence and quantity are also calculated.

2.3.2.2. BOMCOST

Summarizes and returns the cost of the items that are included in current configuration's bill of materials. One or more structure modes can optionally be activated to control how the bill of materials is generated.

```
double BomCost([string structureMode1]; [string structureMode2];
...)
```

structureMode1	Optional name of a structure mode that shall be activated in the BoM generation.

structureMode2	Optional name of a second structure mode that shall be activated in the BoM generation.

Example

`BomCost("SpareParts")`

Returns the summarized cost of the bill of materials that is generated with the structure mode "SpareParts" activated.

2.3.2.3. BOMPRICE

Summarizes and returns the price of the items that are included in current configuration's bill of materials. One or more structure modes can optionally be included to control how the bill of materials is generated.

```
double BomPrice([string structureMode1]; [string structureMode2];
...)
```

structureMode1	Optional name of a structure mode that shall be activated in the BoM generation.
structureMode2	Optional name of a second structure mode that shall be activated in the BoM generation.

Example

`BomPrice("SpareParts")`

Returns the summarized price of the bill of materials that is generated with the structure mode "SpareParts" activated.

2.3.2.4. BOMWEIGHT

Summarizes and returns the weight of the items that are included in current configuration's bill of materials. One or more structure modes can optionally be activated to control how the bill of materials is generated.

```
double BomWeight([string structureMode1]; [string structureMode2];
...)
```

structureMode1	Optional name of a structure mode that shall be activated in the BoM generation.
structureMode2	Optional name of a second structure mode that shall be activated in the BoM generation.

Example

`BomWeight("SpareParts")`

Returns the summarized weight of the bill of materials that is generated with the structure mode "SpareParts" activated.

2.3.2.5. INSTRUCTUREMODE

Returns true if the specified structure mode is active in current BoM generation.

```
bool InStructureMode(string name)
```

name	The name of a structure mode.

Example

`InStructureMode("AsBuilt")`

Returns true if the structure mode "AsBuilt" is activated.

2.3.3. CODE

Expression functions for generating output in JSON, XML and TXT formats using code.

Code

2.3.3.1. CODE

Generates and returns a code file for current configuration.

```
string Code(string codeFileTitle)
```

codeFileTitle	The title of the code file to generate.

Example

`Code("BomToERP")`

Generates and returns the code file with the title "BomToERP".

2.3.4. COMPOSITE

Expression functions for reading and counting data in systems with parent and child configurations.

Child
ChildCount
ChildParamValue
ChildProperty
ChildPropertyF

2.3.4.1. CHILD

Returns the product title or the value of a named parameter, multi-choice value or attribute in the specified child configuration. If no matching child or parameter/attribute is found, then an empty string is returned.

```
string Child(int index; [string paramOrAttrName]; [string productTi-
tle])
```

index	The zero-based index of the configuration to get a value from.
paramOrAttrName	Optional name of a parameter, multi-choice value or attribute.
productTitle	Optional filter to only include child configurations of the specified product.

Example

```
Child(2; "Color")
```

Returns the value of the parameter or attribute named "Color" from the third child configuration.

```
Child(0; "Options.FastDelivery")
```

Returns the value of the multi-choice value named "FastDelivery" belonging to the multi-choice parameter "Options" from the first child configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.2. CHILDCOUNT

Returns the number of child configurations.

```
int ChildCount([string productTitle])
```

productTitle	Optional filter to only count child configurations of the specified product.

Example

```
ChildCount("Engine")
```

Returns the number of child configurations of the product "Engine".

2.3.4.3. CHILDPARAMVALUE

Retrieves the value of a specified parameter within a given child configuration. If the specified child or parameter is not found, an empty string is returned.

```
int ChildParamValue(string name; int index, [string productTitle])
```

name	The name of a parameter or multi-choice value.
index	Zero-based index of the child configuration to read the parameter from.

productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildParamValue("InstallationDays"; 0; "Services")
```

Returns the value of the parameter "InstallationDays" from the first child configuration of the product "Services".

2.3.4.4. CHILDPROPERTY

Returns the property value of the child configuration at specified index. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ChildProperty(int index; string propertyName; [string propertyName2]; ...)
```

index	Zero-based index of the child configuration to read the property from.
propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ChildProperty(2; "ProductId"; "Title")
```

Returns the title of the product that is referenced by the third child configuration.

2.3.4.5. CHILDPROPERTYF

Returns the formatted property value of the child configuration at specified index. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ChildPropertyF(string format; int index; string propertyName; [string propertyName2]; ...)
```

format	Format specifiers to apply when converting the property to string. Search Microsoft's documentation about "format strings".
index	Zero-based index of the child configuration to read the property from.
propertyName	Name of the property to read.

propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ChildPropertyF("yyyy-MM-dd"; 2; "CreationDate")
```

Returns the creation date in a format like "2023-12-31" of the third child configuration.

2.3.4.6. CHILDRENATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the child configurations.

```
int ChildrenAttributeOcc(string name; string value)
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
ChildrenAttributeOcc("SparePart", "True")
```

Returns the number of children for which the Boolean attribute "SpartPart" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.7. CHILDRENATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the child configurations.

```
double ChildrenAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildrenAttributeSum("Weight")
```

Returns the child configurations' sum of the double or integer attribute "Weight".

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.8. CHILDRENPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the child configurations.

```
int ChildrenParameterOcc(string name; string value)
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
ChildrenParameterOcc("Color", "Black")
```

Returns the number of child configurations for which the "Color" parameter has the value "Black".

2.3.4.9. CHILDRENPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the child configurations.

```
double ChildrenParameterSum(string name; [string productTitle])
```

name	The name of a parameter.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildrenParameterSum("Length")
```

Returns the child configurations' sum of the double or integer attribute "Length".

2.3.4.10. CONFIGURATIONINDEX

Returns the zero-based index of current configuration among its sibling configurations. If current configuration is the root configuration, then zero is returned.

```
int ConfigurationIndex([bool ignoreOtherProducts])
```

ignoreOtherProducts	Optional filter to only count sibling configurations of the same product as current configuration.

Example

```
ConfigurationIndex()
```

2.3.4.11. DYNCONFIGATTRIBUTE

Returns the attribute value of a dynamic configuration, i.e., from a configuration that has been auto generated from a dynamic instance.

```
string DynConfigAttribute(string productTitle; string instanceName;
int instanceIndex; string attributeName)
```

productTitle	The title of the product of the dynamic configuration.
instanceName	The name of the dynamic instance driving the dynamic configuration.
instanceIndex	The zero-based index of the dynamic configuration. Use zero for non-indexed instances.
attributeName	The name of the attribute to read.

Example

```
DynConfigAttribute("Wall block"; "Wall 1"; 3; "Weight")
```

Returns the value of the attribute "Weight" from the fourth dynamic "Wall block" configuration that is generated from the dynamic instance "Wall 1".

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.12. DYNCONFIGPARAMETER

Returns the parameter value of a dynamic configuration, i.e., from a configuration that has been auto generated from a dynamic instance.

```
string DynConfigParameter(string productTitle; string instanceName;
int instanceIndex; string parameterName)
```

productTitle	The title of the product of the dynamic configuration.
instanceName	The name of the dynamic instance driving the dynamic configuration.
instanceIndex	The zero-based index of the dynamic configuration. Use zero for non-indexed instances.
parameterName	The name of the parameter to read.

Example

```
DynConfigParameter("Wall block", "Wall 1", 3, "Width")
```

Returns the value of the parameter "Width" from the fourth dynamic "Wall block" configuration that is generated from the dynamic instance "Wall 1".

2.3.4.13. ISSELECTEDCONFIGURATION

Returns true if the configuration, for which the expression function is evaluated, is the selected configuration in the user interface.

```
bool IsSelectedConfiguration()
```

2.3.4.14. MAXCHILDVALUE

Returns the max value of the specified numerical parameter or attribute among the child configurations.

```
double MaxChildValue(string name)
```

name	The name of a parameter or attribute.

Example

```
MaxChildValue("LeadTime")
```

Returns the max value of the parameter or attribute "LeadTime" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.15. MINCHILDVALUE

Returns the min value of the specified numerical parameter or attribute among the child configurations.

```
double MinChildValue(string name)
```

name	The name of a parameter or attribute.

Example

`MinChildValue("InventoryBalance")`

Returns the min value of the parameter or attribute "`InventoryBalance`" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.16. NEXTSIBLING

Returns the product title, a parameter value, or an attribute value from the next sibling configuration. If no argument is attached the product title is returned. If no matching configuration, parameter or attribute is found, then an empty string is returned.

```
string NextSibling([string paramOrAttrName]; [string productTitle])
```

paramOrAttrName	Optional name of the parameter or attribute to read.
productTitle	Optional filter to find the next configuration of the specified product.

Example

`NextSibling("Power")`

Returns the value of the parameter or attribute with the name "Power" from next sibling configuration.

2.3.4.17. NEXTSIBLINGCOUNT

Returns the number of sibling configurations of higher index and optionally of a specified product.

```
string NextSiblingCount([string productTitle])
```

productTitle	Optional filter to only count configurations of the specified product.

Example

```
NextSiblingCount()
```

2.3.4.18. NEXTSIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations with a higher index.

```
int NextSiblingsAttributeOcc(string name; string value)
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
NextSiblingsAttributeOcc("Connected", "True")
```

Returns the number of sibling configurations with a higher index for which the Boolean attribute "Connected" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.19. NEXTSIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations with a higher index.

```
double NextSiblingsAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
NextSiblingsAttributeSum("Weight")
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations with a higher index.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.20. NEXTSIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations with a higher index.

```
int NextSiblingsParameterOcc(string name; string value)
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
NextSiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations with a higher index for which the Boolean parameter "Connected" is true.

2.3.4.21. NEXTSIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations with a higher index.

```
double NextSiblingsParameterSum(string name; [string productTitle])
```

name	The name of a parameter.

productTitle	Optional filter to only include configurations of the specified product.

Example

`NextSiblingsParameterSum("Length")`

Returns sum of the double or integer attribute "Length" from all sibling configurations with a higher index.

2.3.4.22. PARENT

Returns the product title or the value of a named parameter, multi-choice value or attribute of the parent configuration. If no parent configuration or no matching parameter/attribute is found, then an empty string is returned.

```
string Parent([string name])
```

name	Optional name of a parameter, multi-choice value or attribute.

Example

`Parent()`

Returns the title of the parent configuration's product.

`Parent("Color")`

Returns the value of the parameter or attribute named "Color" from the parent configuration.

`Parent("Options.FastDelivery")`

Returns the value of the multi-choice value named "FastDelivery", belonging to the multi-choice parameter "Options", from the parent configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.23. PREVSIBLING

Returns the product title, a parameter value, or an attribute value from the previous sibling configuration. If no argument is attached the product title is returned. If no matching configuration, parameter or attribute is found, then an empty string is returned.

```
string PrevSibling([string paramOrAttrName]; [string productTitle])
```

paramOrAttrName	Optional name of the parameter or attribute to read.
productTitle	Optional filter to find the previous sibling configuration of the specified product.

Example

```
PrevSibling("Power")
```

Returns the value of the parameter or attribute with the name “Power” from previous sibling configuration.

2.3.4.24. PREVSIBLINGCOUNT

Returns the number of sibling configurations of lower index and optionally of a specified product.

```
string PrevSiblingCount([string productTitle])
```

productTitle	Optional filter to only count configurations of the specified product.

Example

```
PrevSiblingCount()
```

2.3.4.25. PREVSIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations with a lower index.

```
int PrevSiblingsAttributeOcc(string name; string value)
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
PrevSiblingsAttributeOcc("Connected", "True")
```

Returns the number of sibling configurations with a lower index for which the Boolean attribute "Connected" is true.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.26. PREVSIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations with a lower index.

```
double PrevSiblingsAttributeSum(string name; [string productTitle])
```

name	The name of an attribute.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
PrevSiblingsAttributeSum("Weight")
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations with a lower index.

Remarks

It's a good habit to put attributes that read values of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies.

2.3.4.27. PREVSIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations with a lower index.

```
int PrevSiblingsParameterOcc(string name; string value)
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
PrevSiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations with a lower index for which the Boolean parameter "Connected" is true.

2.3.4.28. PREVSIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations with a lower index.

```
double PrevSiblingsParameterSum(string name; [string productTitle])
```

name	The name of a parameter.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
PrevSiblingsParameterSum("Length")
```

Returns sum of the double or integer attribute "Length" from all sibling configurations with a lower index.

2.3.4.29. SIBLINGCOUNT

Returns the number of sibling configurations and optionally of a specified product.

```
string SiblingCount([bool includeSelf]; [string productTitle])
```

includeSelf	Optional argument to include current configuration in the sibling count. Default is false.
productTitle	Optional filter to only count configurations of the specified product.

Example

```
SiblingCount()
```

2.3.4.30. SIBLINGSATTRIBUTEOCC

Returns the number of occurrences of the specified attribute/value combination among the sibling configurations.

```
int SiblingsAttributeOcc(string name; string value; [bool in-
cludeSelf])
```

name	The name of an attribute.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".
includeSelf	Optional argument to include current configuration in the attribute count. Default is false.

Example

```
SiblingsAttributeOcc("Connected", "True")
```

Returns the number of sibling configurations for which the Boolean attribute "Connected" is true. Current configuration is not included.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.31. SIBLINGSATTRIBUTESUM

Returns the sum of the specified numerical attribute/value combination among the sibling configurations.

```
double SiblingsAttributeSum(string name; [bool includeSelf]; [string productTitle])
```

name	The name of an attribute.
includeSelf	Optional argument to include current configuration in the attribute sum. Default is false.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
SiblingsAttributeSum("Weight"; true)
```

Returns the sum of the double or integer attribute "Weight" from all sibling configurations including current configuration.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.4.32. SIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations.

```
int SiblingsParameterOcc(string name; string value; [bool includeSelf])
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".
includeSelf	Optional argument to include current configuration in the parameter count. Default is false.

Example

```
SiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations for which the Boolean parameter "Connected" is true. Current configuration is not included.

2.3.4.33. SIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations.

```
double SiblingsParameterSum(string name; [bool includeSelf]; [string productTitle])
```

name	The name of a parameter.
includeSelf	Optional argument to include current configuration in the parameter sum. Default is false.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
SiblingsParameterSum("Length"; true)
```

Returns sum of the double or integer attribute "Length" from all sibling configurations including current configuration.

2.3.5. CONFIGURATION

Expression functions for extracting data from the configuration for which the expression is evaluated.

Configuration
ConfigurationF
ConfigurationIndex
ConfigurationHistory
IsSelectedConfiguration

2.3.5.1. CONFIGURATION

Returns a property value from current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string Configuration(string propertyName; [string propertyName2];
...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
Configuration("StateId"; "Name")
```

Returns the name of the configuration's state.

2.3.5.2. CONFIGURATIONF

Returns a formatted property value from current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ConfigurationF(string format; string propertyName; [string
propertyName2]; ...)
```

format	Format specifiers to apply when converting the property to string. Search Microsoft's documentation about "format strings".
propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ConfigurationF("yyyy-MM-dd"; "CreationDate")
```

Returns the creation date in a format like "2024-12-31" of current configuration.

2.3.5.3. CONFIGURATIONINDEX

Returns the zero-based index of current configuration among its sibling configurations. If current configuration is the root configuration, then zero is returned.

```
int ConfigurationIndex([bool ignoreOtherProducts])
```

ignoreOtherProducts	Optional filter to only count sibling configurations of the same product as current configuration.

Example

```
ConfigurationIndex()
```

2.3.5.4. CONFIGURATIONHISTORY

Returns a property value from current configuration's last history object of specified type. History objects are created when the state of a configuration is promoted or demoted and when the work group ownership is changed.

```
string ConfigurationHistory(int historyType; string propertyName;
[string propertyName2]; ...)
```

historyType	Filters the type of history object. Not specified = 0, State change = 1, Ownership change = 2
propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

`ConfigurationHistory(1; "CreatedById"; "Name")`

Returns the name of the user that made the last state change of the configuration.

2.3.5.5. ISSELECTEDCONFIGURATION

Returns true if the configuration, for which the expression function is evaluated, is the selected configuration in the user interface.

```
bool IsSelectedConfiguration()
```

2.3.6. DATE AND TIME

Expression functions for getting and calculating dates and times.

AddDays
Date
GetWeekOfYear
Now

2.3.6.1. ADDDAYS

Adds days to a DateTime expression and returns the result.

```
DateTime AddDays(DateTime date; double/int days)
```

date	A date time to add days to.
days	A number of whole or fractional days to add.

Example

`AddDays(Date("2023-12-31");1)`

Returns the date time value of 2024-01-01 00:00:00.

2.3.6.2. DATE

Converts a string to a DateTime and returns the result. If the conversion fails the default DateTime is returned, i.e, 0001-01-01 00:00:00.

```
DateTime Date(string dateTime; [string format])
```

dateTime	A string representing a date and optionally a time.
format	Optional format definition for the string to convert.

Example

```
Date("31/12/2023";"dd/MM/yyyy")
```

Returns the date time value of 2023-12-31 00:00:00.

2.3.6.3. GETWEEKOFYEAR

Returns the week number for the specified date.

```
int GetWeekOfYear(DateTime date; [int calendarWeekRule]; [int
startOfWeek])
```

date	The date to find the week number for.
calendarWeekRule	Optional rule for defining week one of a year. FirstDay = 0, FirstFullWeek = 1, FirstFourDayWeek = 2 (default)
startOfWeek	Optional rule for defining the first day of a week. Sunday = 0, Monday = 1, Tuesday = 2, Wednesday = 3, Thursday = 4, Friday = 5, Saturday = 6

Example

```
GetWeekOfYear(Date("2023-12-31")
```

Returns 52.

2.3.6.4. NOW

Returns current date and time as a DateTime.

```
DateTime Now()
```

Example

```
ToString(Now(); "yyyy-MM-dd hh:mm:ss")
```

Returns current date and time as a string formatted like "2023-03-20 18:38:00".

2.3.7. DICTIONARIES

Expression functions for reading and interpolating values of dictionary tables.

Dictionary
FindClosestDicKey
FindClosestDicKeyCol
InterpolateDic
InterpolateDicByCol
IsSet

2.3.7.1. DICTIONARY

Returns the dictionary cell value in the specified column at the row with the attached key.

```
string Dictionary(string dictionaryName; string key; string column-
Name)
```

dictionaryName	The name of the dictionary to read data from.
key	The key to match when selecting the row in the dictionary.
columnName	The name of the column to read data from.

Example

```
Dictionary("City"; "Gothenburg"; "Latitude")
```

Read the column "Latitude" from the dictionary "City" at the row with the key "Gothenburg". The answer might be "57.71".

Remarks

Dictionaries can also be read using direct references without calling the Dictionary() method. The above example is then written:

```
City["Gothenburg"; City.Latitude]
```

The advantage with using the Dictionary() method is that both which dictionary and which column to read can be parametric.

2.3.7.2. FINDCLOSESTDICKEY

Returns the closest matching numerical key in the dictionary. The key is returned as a string so that it can be used in a nested call that reads a cell value.

```
string FindClosestDicKey(string dictionaryName; double value; [int
closestHigherOrLower])
```

dictionaryName	The name of the dictionary to find the closest key in.
value	The value to find the closest matching key for.
closestHigherOrLower	Optional filter. Find closest key = 0, Find closest higher or equal key = 1, Find closest lower or equal key = -1

Examples

```
FindClosestDicKey("PrimeNumbers"; 6000)
```

Returns the key closest to 6000 in the dictionary "PrimeNumbers", e.g. "6007".

```
FindClosestDicKey("PrimeNumbers"; 6000; -1)
```

Returns the key closest lower or equal to 6000 in the dictionary "PrimeNumbers", e.g. "5987".

Remarks

This method requires that the keys of the dictionary are numerical or can be converted to double using "NumberFormatInfo.InvariantInfo".

2.3.7.3. FINDCLOSESTDICKEYCOL

Returns the key of the closest matching numerical column value in the dictionary. The key is returned as a string so that it can be used in a nested call that reads a cell value.

```
string FindClosestDicKeyByCol(string dictionaryName, string column-
Name; double value; [int closestHigherOrLower])
```

dictionaryName	The name of the dictionary to find the closest key in.
columnName	The name of the dictionary column to search.
value	The value to find the closest match for.
closestHigherOrLower	Optional filter. Find closest key = 0, Find closest higher or equal key = 1, Find closest lower or equal key = -1

Examples

```
FindClosestDicKeyByCol("Size"; "Height"; 2.5; 1)
```

Returns the key of the row with the "Height" column value that is closest higher or equal to 2.5 in the dictionary "Size".

Remarks

This method requires that the keys and referred column of the dictionary are numerical or can be converted to double using "NumberFormatInfo.InvariantInfo".

2.3.7.4. INTERPOLATEDIC

Finds the closest lower and higher key value, calculates the output column's value using linear interpolation and returns the result. If an exact match of the value is found, then no interpolation is done. If the value is outside the range of keys, then an extrapolation is done.

```
double InterpolateDic(string dictionaryName; double value; string
outputColumnName)
```

dictionaryName	The name of the dictionary to interpolate in.
value	The value to find the closest lower and higher key value for.
outputColumnName	The name of the column to calculate the value for using interpolation.

Example

```
InterpolateDic("MovementAtTime", 12.3; "Distance")
```

Returns the interpolated value of the column "Distance" in the dictionary "MovementAtTime" at the interpolation key value 12.3 (i.e. at the time 12.3 s).

Remarks

This method requires that the keys and referred column of the dictionary are numerical or can be converted to double using "NumberFormatInfo.InvariantInfo".

2.3.7.5. INTERPOLATEDICBYCOL

Finds the closest lower and higher value in the driving column, calculates the output column's value using linear interpolation and returns the result. If the output column name is omitted, then the value of the key column is calculated. If an exact match of the value is found, then no interpolation is done. If the value is outside the range of the driving column's values, then an extrapolation is done.

```
double InterpolateDicByCol(string dictionaryName; string driv-
ingColumnName; double value; [string outputColumnName])
```

dictionaryName	The name of the dictionary to interpolate in.
drivingColumnName	The name of the column whose values shall drive the interpolation.
value	The value to find the closest lower and higher key value for.
outputColumnName	Optional name of the column to calculate the value for using interpolation. If omitted the key column is calculated.

Example

```
InterpolateDicByCol("MovementAtTime", "Distance", 30; "Accelera-
tion")
```

Returns the interpolated value of the column "Acceleration" in the dictionary "MovementAt-Time" at the interpolation value 30 of the column "Distance".

Remarks

This method requires that the keys and referred columns of the dictionary are numerical or can be converted to double using "NumberFormatInfo.InvariantInfo".

2.3.7.6. ISSET

Returns true if the specified parameter, attribute, multi-choice value or dictionary cell is set, otherwise false. If no matching parameter, attribute, multi-choice value or dictionary is found false is returned.

Overloads

```
bool IsSet(string name; [string dictKey]; [string dictColNameOrIn-
dex])
bool IsSet(Parameter parameter)
bool IsSet(Attribute attribute)
bool IsSet(MultiChoiceValue multiChoiceValue)
```

name	The name of a parameter, attribute, multi-choice value or dictionary.
dictKey	Optionally the key of a dictionary row.
dictColNameOrIndex	Optionally the name or index of a dictionary column.

Example

```
IsSet("Model")
```

Returns true if a parameter or attribute with the name "Model" is set, otherwise false.

2.3.8. ITERATORS

Expression functions for reading data during an iteration of objects. Iterations can be triggered by repeater objects in the tree structures of code and reports, by the "Loop while included" field in graphics and item structure, by the expression function ParameterOcc, and by data selectors.

ActiveAttribute
ActiveItem
ActiveKey
ActiveLoopIndex
ActiveParameter
ActiveParamValue
ActivePriceItem

2.3.8.1. ACTIVEATTRIBUTE

Returns a property value of active attribute during an iteration of the attributes of a configuration.

```
string ActiveAttribute(string propertyName)
```

propertyName	The name of a property to read. The supported properties are "IsWarning", "Name", "Title", "Type", "Unit" and "Value".

Example

```
ActiveAttribute("Value")
```

Returns the value of active attribute converted to a string.
Note! Boolean values are returned as "Yes" or "No" for display reasons.

2.3.8.2. ACTIVEITEM

Returns a fixed or calculated property of current parent item, current item structure or current child item when looping through a bill of materials using a repeater.

```
string ActiveItem(string target; string propertyName; [string propertyName2]; ...)
```

target	The object to read a property from. The supported values are "Parent", "Structure" and "Child" (case insensitive).
propertyName	Name of a property or a type of calculated data to read. See list of arguments for calculated data below.
propertyName2	Optional name of a property to read from an object that is referenced by the first property or an extra argument that is required by the type of calculated data.

Arguments for calculated data

The following arguments for calculated data are supported (case insensitive):

"Attribute"	Returns the name of a structure attribute that has been set for current structure node. A third argument with a zero based index is required.
"AttributeCount"	Returns the number of structure attributes that have been set for current structure node.

"AttributeUnit"	Returns the unit of a structure attribute that has been set for current structure node. A third argument with either the name of a structure attribute or a zero based index is required.
"AttributeValue"	Returns the calculated value of a structure attribute that has been set for this structure node. A third argument with either the name of a structure attribute or a zero based index is required.
"Cost"	Returns the calculated cost of current child item.
"Level"	Returns the child item's level in the bill of materials at the position given by current structure node. The root item is at level 0.
"Price"	Returns the calculated price of current child item.

Examples

`ActiveItem("Child"; "ItemNo")`

Returns the item number of the child item.

`ActiveItem("Structure"; "Quantity")`

Returns the (calculated) quantity of the child item in the structure of the parent item.

`ActiveItem("Structure"; "Level")`

Returns the child item's level in the bill of materials at the position given by current structure node.

`ActiveItem("Parent"; "ItemNo")`

Returns the item number of the parent item.

`ActiveItem("Structure"; "AttributeValue"; "Length")`

Returns the calculated value of the structure attribute "Length".

`ActiveItem("Child"; "Cost")`

Returns the calculated cost of the child item.

Remarks

Some structure properties like context, operation sequence and quantity are also calculated.

2.3.8.3. ACTIVEKEY

Returns the active key value during an iteration of the dictionary that is mapped to a data selector. This method should be used when defining the expressions for a data selector's calculated columns, filters, sorters and actions.

```
string ActiveKey()
```

Example

```
Laminate[ActiveKey(); Laminate.IsWood]
```

Returns the value in the column "IsWood" for current row during an iteration of the rows of the data selector "Laminate".

2.3.8.4. ACTIVELOOPINDEX

Returns the zero-based index during an iteration of objects triggered by a "Loop while included". If no origin is specified, the index of the closest loop is returned in case of nested loops.

```
int ActiveLoopIndex([int origin])
```

origin	Optional number specifying the origin of the loop while included. Report = 0, Bill of Materials = 1, Graphics = 2, Code = 3

2.3.8.5. ACTIVEPARAMETER

Returns a property value of active parameter or active multi-choice value during an iteration of parameters and multi-choice values of a configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching property is found, then an empty string is returned.

```
string ActiveParameter(string propertyName; [string propertyName2];
...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Examples

```
ActiveParameter("Title")
```

Returns the title of active parameter or active multi-choice value. In case of a multi-choice value the returned string contains both the parameter title and the value title separated by ", ". If only the title of the multi-choice value is desired use the function ActiveParamValue.

```
ActiveParameter("Value")
```

Returns the value of active parameter or active multi-choice value converted to a string. Note! Boolean values are returned as "Yes" or "No" for display reasons. Bool parameters and multi-choice values with "three-state" active can also have the value "Indeterminate".

`ActiveParameter("Type")`

Returns the data type of active parameter or active multi-choice value. Possible return values are "bool", "double", "int" and "string".

`ActiveParameter("Class")`

Returns the class of active parameter. Possible return values are "BoolParam", "DoubleParam", "IntParam", "LookupParam", "MultiChoiceParam" and "StringParam".

2.3.8.6. ACTIVEPARAMVALUE

Returns a property value of active multi-choice value during an iteration of parameters and multi-choice values of a configuration. Properties of reference type can be followed to instead return a property of the referenced object. If current parameter isn't a multi-choice parameter or if no matching property is found, then an empty string is returned.

```
string ActiveParamValue(string propertyName; [string propertyName2]; ...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ActiveParameter("Class") = "MultiChoiceParam" ? ActiveParamValue("Title") : ActiveParameter("Title")
```

If current parameter is a multi-choice parameter, then the title of current multi-choice value is returned else the title of current parameter is returned.

2.3.8.7. ACTIVEPRICEITEM

Returns a property value of active price item during an iteration of the price items that have been selected in the price list for a configuration.

```
string ActivePriceItem(string propertyName)
```

propertyName	The name of a property to read.

Examples

`ActivePriceItem("Title")`

Returns the title of the price item.

`ActivePriceItem("ValueF")`
Returns the formatted value of the price item including thousand separator and presentation rounding.

`ActivePriceItem("OrgValueF")`

Returns the original value of the price item even if it has been manually overridden. The output is formatted with the thousand separator and presentation rounding.

2.3.9. GRAPHICS

Expression functions for generation of color values and checks for graphics modes. Colors are represented as 32-bit integers with 8 bits each for alpha, red, green, and blue (ARGB).

ColorBrighten
ColorDarken
ColorFromArgb
ColorFromName
ColorFromRal
InGraphicMode

2.3.9.1. COLORBRIGHTEN

Creates a more light version of the specified color by lighten it by a percentage.

```
int ColorBrighten(int argb; int percentage)
```

argb	Color represented as a 32-bit integer.
percentage	The percentage to lighten the color by.

Example

`ColorBrighten(ColorFromArgb(255; 0; 100; 255); 20)`

Lightens each component of R, G and B with 51 (20% of 255) and returns the resulting color i.e. argb(255, 51, 151, 255).

2.3.9.2. COLORDARKEN

Creates a more dark version of the specified color by darken it by a percentage.

```
int ColorDarken(int argb; int percentage)
```

argb	Color represented as a 32-bit integer.
percentage	The percentage to darken the color by.

Example

```
ColorDarken(ColorFromArgb(255; 0; 100; 255); 20)
```

Darkens each component of R, G and B with 51 (20% of 255) and returns the resulting color i.e. argb(255, 0, 49, 204).

2.3.9.3. COLORFROMARGB

Creates a 32-bit integer color value from the specified ARGB components.

```
int ColorFromArgb(int alpha; int red, int green; int blue)
```

alpha	The alpha component. Valid values are 0 through 255.
red	The red component. Valid values are 0 through 255.
green	The green component. Valid values are 0 through 255.
blue	The blue component. Valid values are 0 through 255.

Example

```
ColorFromArgb(127; 0; 255; 255)
```

Returns a semi-transparent cyan color.

2.3.9.4. COLORFROMNAME

Creates a 32-bit integer color value from a known color name. Check Microsoft's documentation of the KnownColor enumeration.

```
int ColorFromName(string name)
```

name	Name of a predefined color.

Example

```
ColorFromName("SandyBrown")
```

Returns the color argb(255, 244, 164, 96).

2.3.9.5. COLORFROMRAL

Creates a 32-bit integer color value from a RAL color code. The 210 color codes of "RAL 840-HR" are supported.

```
int ColorFromRal(int ralCode)
```

ralCode	The number of a RAL code.

Example

`ColorFromRal(507)`

Returns the color argb(255, 62, 95, 138).

2.3.9.6. INGRAPHICMODE

Returns true if the specified graphics mode is active in current graphics generation.

```
bool InGraphicMode(string name)
```

name	The name of a graphics mode.

Example

`InGraphicMode("Report")`

Returns true if the graphics mode "Report" is activated.

2.3.10. MATH

Expression functions for math operations and rounding of values.

Abs
ArcCos
ArcSin
ArcTan
Ceiling
Cos
Floor
IsNumerical
Log
Max
Min
PI
Pow
Round
Sin
Sqrt
Tan

2.3.10.1. ABS

Returns the absolute value of the specified number.

```
double Abs(double/int value)
```

value	A number.

2.3.10.2. ARCCOS

Returns the angle, in degrees, whose cosine is the specified number. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double ArcCos(double/int value)
```

value	A number representing a cosine.

2.3.10.3. ARCSIN

Returns the angle, in degrees, whose sine is the specified number. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double ArcSin(double/int value)
```

value	A number representing a sine.

2.3.10.4. ARCTAN

Returns the angle, in degrees, whose tangent is the specified number. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double ArcTan(double/int value)
```

value	A number representing a tangent.

2.3.10.5. CEILING

Returns the value rounded up. Optionally the number of decimal digits can be defined.

```
double Ceiling(double/int value; [int digits])
```

value	A number.
digits	The number of decimal digits in the return value. Negative numbers are also supported.

Example

```
Ceiling(731.9; -1)
```

Returns 740.

2.3.10.6. COS

Returns the cosine of the specified angle, in degrees. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double Cos(double/int angle)
```

angle	Angle in degrees.

Example

```
Cos(60)
```

Returns 0.5.

2.3.10.7. FLOOR

Returns the value rounded down. Optionally the number of decimal digits can be defined.

```
double Floor(double/int value; [int digits])
```

value	A number.
digits	The number of decimal digits in the return value. Negative numbers are also supported.

Example

```
Floor(735.9; -1)
```

Returns 730.

2.3.10.8. ISNUMERICAL

Returns true if the text can be converted to a number, otherwise false is returned. Optionally the method can check only for integer values. The conversion attempt is made using the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

```
double IsNumerical(string text; [bool onlyIntegers])
```

text	The text to check.
onlyIntegers	Optionally check only for integer values. Default false.

Example

```
IsNumerical("-320.77")
```

Returns true.

2.3.10.9. LOG

Returns the logarithm of the specified number.

```
double Log(double/int value; [double/int base])
```

value	The number whose logarithm is to be found.
base	Optional base of the logarithm. Default 10.

Example

```
Log(1000)
```

Returns 3.

2.3.10.10. MAX

Returns the larger of two or more numerical values.

```
double Max(double/int value1; double/int value2; [double/int value
3]; ...)
```

value1	The first value to compare.
value2	The second value to compare.
value3	Optional third value to compare.

Example

```
Max(-300; 200; 150)
```

Returns 200.

2.3.10.11. MIN

Returns the smaller of two or more numerical values.

```
double Min(double/int value1; double/int value2; [double/int value
3]; ...)
```

value1	The first value to compare.
value2	The second value to compare.
value3	Optional third value to compare.

Example

```
Min(-300; 200; 150)
```

Returns -300.

2.3.10.12. PI

Returns the value of the π constant, i.e., 3.1415926535897931.

```
double PI()
```

2.3.10.13. POW

Returns the number raised to a specified power.

```
double Pow(double/int value; double/int power)
```

value	The number to raise to a power.
power	The power.

Example

```
Pow(2; 8)
```

Returns 256.

2.3.10.14. ROUND

Returns the rounded value. Half values are rounded to even. Optionally the number of decimal digits can be defined.

```
double Round(double/int value; [int digits])
```

value	A number.
digits	The number of decimal digits in the return value. Negative numbers are also supported.

Example

```
Round(735; -1)
```

Returns 740.

2.3.10.15. SIN

Returns the sine of the specified angle, in degrees. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double Sin(double/int angle)
```

angle	Angle in degrees.

Example

`Sin(90)`

Returns 1.

2.3.10.16. SQRT

Returns the positive square root of the number. If the number is negative then zero is returned.

```
double Sqrt(double/int value)
```

value	The number whose square root is to be found.

Example

`Sqrt(6.25)`

Returns 2.5.

2.3.10.17. TAN

Returns the tangent of the specified angle, in degrees. Radians can be used by changing the setting *Settings* > *Web* > *Unit for angle*.

```
double Tan(double/int angle)
```

angle	Angle in degrees.

Example

`Tan(45)`

Returns 1.

2.3.11. MEMORY

Expression functions for temporary saving data to the memory and reading from the memory during the traversal of a tree structure such as when generating a report or a bill of materials.

MemGetBool
MemGetDouble
MemGetInt
MemGetString
MemSetBool
MemSetDouble
MemSetInt
MemSetString

2.3.11.1. MEMGETBOOL

Returns a Boolean value that has previously been stored to the memory during the same server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. If no variable is found false is returned.

```
bool MemGetBool([string name])
```

name	Optional name of a variable to get the value of. If omitted the value is get from the Boolean default variable.

Example

```
MemGetBool("GraphicsForReport")
```

Gets the value of the Boolean variable "GraphicsForReport".

2.3.11.2. MEMGETDOUBLE

Returns a double value that has previously been stored to the memory during the same server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. If no variable is found zero is returned.

```
double MemGetDouble([string name])
```

name	Optional name of a variable to get the value of. If omitted the value is get from the double default variable.

Example

```
MemGetDouble("Sum")
```

Gets the value of the double variable "Sum".

2.3.11.3. MEMGETINT

Returns an integer value that has previously been stored to the memory during the same server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. If no variable is found zero is returned.

```
int MemGetInt([string name])
```

name	Optional name of a variable to get the value of. If omitted the value is get from the integer default variable.

Example

```
MemGetInt("Sum")
```

Gets the value of the integer variable "Sum".

2.3.11.4. MEMGETSTRING

Returns a string value that has previously been stored to the memory during the same server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. If no variable is found the empty string is returned.

```
string MemGetString([string name])
```

name	Optional name of a variable to get the value of. If omitted the value is get from the string default variable.

Example

```
MemGetString("ConcatenatedText")
```

Gets the value of the string variable "ConcatenatedText".

2.3.11.5. MEMSETBOOL

Stores a Boolean value to the memory as temporary variable during the current server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. The set value is returned.

```
bool MemSetBool(bool value; [string name])
```

value	A value to store.
name	Optional name of a variable to store the value as. If omitted the value is set for the Boolean default variable.

Example

```
MemSetBool(true; "GraphicsForReport")
```

Sets the value of the Boolean variable "GraphicsForReport" to true.

2.3.11.6. MEMSETDOUBLE

Stores a double value to the memory as temporary variable during the current server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. The set value is returned.

```
double MemSetDouble(double value; [string name])
```

value	A value to store.
name	Optional name of a variable to store the value as. If omitted the value is set for the double default variable.

Example

```
MemSetDouble(MemGetDouble("Sum") + 1; "Sum")
```

Adds one to the double variable "Sum".

2.3.11.7. MEMSETINT

Stores an integer value to the memory as temporary variable during the current server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. The set value is returned.

```
int MemSetInt(int value; [string name])
```

value	A value to store.
name	Optional name of a variable to store the value as. If omitted the value is set for the integer default variable.

Example

```
MemSetInt(MemGetInt("Sum") + 1; "Sum")
```

Adds one to the integer variable "Sum".

2.3.11.8. MEMSETSTRING

Stores a string value to the memory as temporary variable during the current server round-trip. This is typically used during the traversal of a tree structure such as when generating a report or a bill of materials. The set value is returned.

```
string MemSetString(string value; [string name])
```

value	A value to store.
name	Optional name of a variable to store the value as. If omitted the value is set for the string default variable.

Example

```
MemSetString(firstName + " " + familyName; "ConcatenatedText")
```

Sets the value of the string variable "ConcatenatedText" to the value of the parameters or attributes firstName and familyName separated by a space character.

2.3.12. NUMBER SERIES

Expression functions for generation and reading of number series values.

NumberSeries

2.3.12.1. NUMBERSERIES

Generates and returns next value of the specified number series or returns the active configuration's value of the number series if a number has already been generated. This is typically used in the *Configuration identity expression* field for products.

```
int NumberSeries(string name)
```

name	The name of a number series.

Example

```
NumberSeries("QuoteNumber")
```

Generates and returns the next number of the series "QuoteNumber" or returns the active configuration's value if a number already has been generated.

2.3.13. PROPERTIES

Expression functions for reading properties from configurations and other classes. Properties are database fields that are stored for the object.

ChildProperty
ChildPropertyF
Configuration
ConfigurationF
ConfigurationHistory
DynConfigProperty
User
WorkGroup

2.3.13.1. CHILDPROPERTY

Returns the property value of the child configuration at specified index. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ChildProperty(int index; string propertyName; [string propertyName2]; ...)
```

index	Zero-based index of the child configuration to read the property from.
propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ChildProperty(2; "ProductId"; "Title")
```

Returns the title of the product that is referenced by the third child configuration.

2.3.13.2. CHILDPROPERTYF

Returns the formatted property value of the child configuration at specified index. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ChildPropertyF(string format; int index; string propertyName;
[string propertyName2]; ...)
```

format	Format specifiers to apply when converting the property to string. Search Microsoft's documentation about "format strings".
index	Zero-based index of the child configuration to read the property from.
propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ChildPropertyF("yyyy-MM-dd"; 2; "CreationDate")
```

Returns the creation date in a format like "2023-12-31" of the third child configuration.

2.3.13.3. CONFIGURATION

Returns a property value from current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string Configuration(string propertyName; [string propertyName2];
...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
Configuration("StateId"; "Name")
```

Returns the name of the configuration's state.

2.3.13.4. CONFIGURATIONF

Returns a formatted property value from current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string ConfigurationF(string format; string propertyName; [string propertyName2]; ...)
```

format	Format specifiers to apply when converting the property to string. Search Microsoft's documentation about "format strings".
propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ConfigurationF("yyyy-MM-dd"; "CreationDate")
```

Returns the creation date in a format like "2024-12-31" of current configuration.

2.3.13.5. CONFIGURATIONHISTORY

Returns a property value from current configuration's last history object of specified type. History objects are created when the state of a configuration is promoted or demoted and when the work group ownership is changed.

```
string ConfigurationHistory(int historyType; string propertyName; [string propertyName2]; ...)
```

historyType	Filters the type of history object. Not specified = 0, State change = 1, Ownership change = 2
propertyName	The name of a property to read.

propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ConfigurationHistory(1; "CreatedById"; "Name")
```

Returns the name of the user that made the last state change of the configuration.

2.3.13.6. DYNCONFIGPROPERTY

Returns the property value of a dynamic configuration, i.e., from a configuration that has been auto generated from a dynamic instance.

```
string DynConfigProperty(string productTitle; string instanceName;
int instanceIndex; string propertyName; [string propertyName2]; ...)
```

productTitle	The title of the product of the dynamic configuration.
instanceName	The name of the dynamic instance driving the dynamic configuration.
instanceIndex	The zero-based index of the dynamic configuration. Use zero for non-indexed instances.
propertyName	The name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
DynConfigProperty("Wall block", "Wall 1", 3, "Identity")
```

Returns the value of the property "Identity" from the fourth dynamic "Wall block" configuration that is generated from the dynamic instance "Wall 1".

2.3.13.7. USER

Returns the property value from the active user. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string User(string propertyName; [string propertyName2]; ...)
```

propertyName	Name of the property to read.

propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
User("LanguageId"; "Title")
```

Returns the title of the active user's language.

2.3.13.8. WORKGROUP

Returns the property value from the active user's work group. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string WorkGroup(string propertyName; [string propertyName2]; ...)
```

propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
WorkGroup("Title")
```

Returns the title of the active user's work group.

2.3.14. PARAMETERS

Expression functions for reading and counting parameter values.

ActiveParameter
ActiveParamValue
ChildParamValue
ChildrenParameterOcc
ChildrenParameterSum
DynConfigParameter
IsHidden
IsSet
MaxChildValue
MinChildValue
MultiChoiceAny
MultiChoiceCount
MultiChoiceMax
MultiChoiceMin
NextSiblingsParameterOcc
NextSiblingsParameterSum

Parameter
ParameterOcc
ParameterValueTitle
Parent
PrevSiblingsParameterOcc
PrevSiblingsParameterSum
SelectedMultiChoiceValues
SiblingsParameterOcc
SiblingsParameterSum

2.3.14.1. ACTIVEPARAMETER

Returns a property value of active parameter or active multi-choice value during an iteration of parameters and multi-choice values of a configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching property is found, then an empty string is returned.

```
string ActiveParameter(string propertyName; [string propertyName2];
...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Examples

`ActiveParameter("Title")`

Returns the title of active parameter or active multi-choice value. In case of a multi-choice value the returned string contains both the parameter title and the value title separated by ", ". If only the title of the multi-choice value is desired use the function ActiveParamValue.

`ActiveParameter("Value")`

Returns the value of active parameter or active multi-choice value converted to a string. Note! Boolean values are returned as "Yes" or "No" for display reasons. Bool parameters and multi-choice values with "three-state" active can also have the value "Indeterminate".

`ActiveParameter("Type")`

Returns the data type of active parameter or active multi-choice value. Possible return values are "bool", "double", "int" and "string".

`ActiveParameter("Class")`

Returns the class of active parameter. Possible return values are "BoolParam", "DoubleParam", "IntParam", "LookupParam", "MultiChoiceParam" and "StringParam".

2.3.14.2. ACTIVEPARAMVALUE

Returns a property value of active multi-choice value during an iteration of parameters and multi-choice values of a configuration. Properties of reference type can be followed to instead return a property of the referenced object. If current parameter isn't a multi-choice parameter or if no matching property is found, then an empty string is returned.

```
string ActiveParamValue(string propertyName; [string propertyName2]; ...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ActiveParameter("Class") = "MultiChoiceParam" ? ActiveParamValue("Title") : ActiveParameter("Title")
```

If current parameter is a multi-choice parameter, then the title of current multi-choice value is returned else the title of current parameter is returned.

2.3.14.3. CHILDPARAMVALUE

Retrieves the value of a specified parameter within a given child configuration. If the specified child or parameter is not found, an empty string is returned.

```
int ChildParamValue(string name; int index, [string productTitle])
```

name	The name of a parameter or multi-choice value.
index	Zero-based index of the child configuration to read the parameter from.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildParamValue("InstallationDays"; 0; "Services")
```

Returns the value of the parameter "InstallationDays" from the first child configuration of the product "Services".

2.3.14.4. CHILDRENPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the child configurations.

```
int ChildrenParameterOcc(string name; string value)
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
ChildrenParameterOcc("Color", "Black")
```

Returns the number of child configurations for which the "Color" parameter has the value "Black".

2.3.14.5. CHILDRENPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the child configurations.

```
double ChildrenParameterSum(string name; [string productTitle])
```

name	The name of a parameter.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
ChildrenParameterSum("Length")
```

Returns the child configurations' sum of the double or integer attribute "Length".

2.3.14.6. DYNCONFIGPARAMETER

Returns the parameter value of a dynamic configuration, i.e., from a configuration that has been auto generated from a dynamic instance.

```
string DynConfigParameter(string productTitle; string instanceName;
int instanceIndex; string parameterName)
```

productTitle	The title of the product of the dynamic configuration.
instanceName	The name of the dynamic instance driving the dynamic configuration.
instanceIndex	The zero-based index of the dynamic configuration. Use zero for non-indexed instances.

parameterName	The name of the parameter to read.

Example

```
DynConfigParameter("Wall block", "Wall 1", 3, "Width")
```

Returns the value of the parameter "Width" from the fourth dynamic "Wall block" configuration that is generated from the dynamic instance "Wall 1".

2.3.14.7. ISHIDDEN

Returns true if the specified parameter is hidden due to sub parameter rules. A parameter that is hidden due to sub parameter rules is treated as disabled and not part of the configuration.

```
bool IsHidden(Parameter parameter)
```

parameter	A parameter.

Example

```
IsHidden(Model)
```

Returns true if a parameter with the name "Model" is hidden due to sub parameter rules.

Remarks

Parameters that are located on hidden tabs are not necessarily hidden from the perspective of the configurator. Parameters on hidden tabs can still get values set via rules and the IsHidden() method will return false for them unless they are hidden due to sub parameter rules.

2.3.14.8. ISSET

Returns true if the specified parameter, attribute, multi-choice value or dictionary cell is set, otherwise false. If no matching parameter, attribute, multi-choice value or dictionary is found false is returned.

Overloads

```
bool IsSet(string name; [string dictKey]; [string dictColNameOrIndex])
bool IsSet(Parameter parameter)
bool IsSet(Attribute attribute)
bool IsSet(MultiChoiceValue multiChoiceValue)
```

name	The name of a parameter, attribute, multi-choice value or dictionary.
dictKey	Optionally the key of a dictionary row.

dictColNameOrIndex	Optionally the name or index of a dictionary column.

Example

`IsSet("Model")`

Returns true if a parameter or attribute with the name "Model" is set, otherwise false.

2.3.14.9. MAXCHILDVALUE

Returns the max value of the specified numerical parameter or attribute among the child configurations.

```
double MaxChildValue(string name)
```

name	The name of a parameter or attribute.

Example

`MaxChildValue("LeadTime")`

Returns the max value of the parameter or attribute "LeadTime" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.14.10. MINCHILDVALUE

Returns the min value of the specified numerical parameter or attribute among the child configurations.

```
double MinChildValue(string name)
```

name	The name of a parameter or attribute.

Example

`MinChildValue("InventoryBalance")`

Returns the min value of the parameter or attribute "`InventoryBalance`" among the child configurations.

Remarks

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.14.11. MULTICHOICEANY

Returns true if any of the true/false values of the specified multi-choice parameter has the value true.

Overloads

```
bool MultiChoiceAny(string name)
bool MultiChoiceAny(MultiChoiceParameter multiChoiceParameter)
```

name	The name of a multi-choice parameter.

Example

```
MultiChoiceAny(Options)
```

Returns true if any of the true/false values of the multi-choice parameter Options has the value true.

2.3.14.12. MULTICHOICECOUNT

Returns the number of true/false values that have the value true for the specified multi-choice parameter.

Overloads

```
int MultiChoiceCount(string name)
int MultiChoiceCount(MultiChoiceParameter multiChoiceParameter)
```

name	The name of a multi-choice parameter.

Example

```
MultiChoiceCount(Options)
```

Returns the number of true/false values that have the value true for the multi-choice parameter Options.

2.3.14.13. MULTICHOICEMAX

Returns the maximum value among the selected multi-choice values or zero if no value is selected. All value names of the multi-choice parameter must be numerical

Overloads

`double MultiChoiceMax(string name)`

`double MultiChoiceMax(MultiChoiceParameter multiChoiceParameter)`

name	The name of a multi-choice parameter.

Example

`MultiChoiceMax(AcceptedDeliveryWeeks)`

Returns the maximum value from the names of the selected values of the multi-choice parameter AcceptedDeliveryWeeks.

2.3.14.14. MULTICHOICEMIN

Returns the minimum value among the selected multi-choice values or zero if no value is selected. All value names of the multi-choice parameter must be numerical

Overloads

`double MultiChoiceMin(string name)`

`double MultiChoiceMin(MultiChoiceParameter multiChoiceParameter)`

name	The name of a multi-choice parameter.

Example

`MultiChoiceMin(AcceptedDeliveryWeeks)`

Returns the minum value from the names of the selected values of the multi-choice parameter AcceptedDeliveryWeeks.

2.3.14.15. NEXTSIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations with a higher index.

`int NextSiblingsParameterOcc(string name; string value)`

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
NextSiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations with a higher index for which the Boolean parameter "Connected" is true.

2.3.14.16. NEXTSIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations with a higher index.

```
double NextSiblingsParameterSum(string name; [string productTitle])
```

name	The name of a parameter.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
NextSiblingsParameterSum("Length")
```

Returns sum of the double or integer attribute "Length" from all sibling configurations with a higher index.

2.3.14.17. PARAMETER

Returns the property value of a parameter, lookup value or multi-choice value. Properties of reference type can be followed to instead return a property of the referenced object. If no matching property is found, then an empty string is returned.

Overloads

```
string Parameter(string name; string propertyName; [string propertyName2]; ...)
string Parameter(Parameter parameter; string propertyName; [string propertyName2]; ...)
string Parameter(LookupValue lookupValue; string propertyName; [string propertyName2]; ...)
string Parameter(MultiChoiceValue multiChoiceValue; string propertyName; [string propertyName2]; ...)
```

name	The name of the parameter, lookup value or multi-choice value to read a property of. For lookup values and multi-choice values both the name of the parameter and the name of the value should be entered separated by a dot.

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Examples

`Parameter(MyParameter; "Title")`

Returns the title of the parameter MyParameter.

`Parameter(MyParameter + ""; "Title")`

Returns the title of the parameter whose name is equal to the value of MyParameter. In this case the argument is treated as a string instead of a Parameter since it's a composite expression.

2.3.14.18. PARAMETEROCC

Returns the number of parameters and multi-choice values that matches a predicate expression within the active configuration. The predicate expression is defined using the ActiveParameter() function.

```
int ParameterOcc(bool predicateExpression)
```

predicateExpression	The expression to evaluate for each parameter to decide if it should be counted or not.

Example

`ParameterOcc(ActiveParameter("Value") = "/")`

Returns the number of parameters with the value "/" in active configuration.

```
ParameterOcc(ActiveParameter("type") = "bool" && ActiveParameter("Value") = "Yes")
```

Returns the number of bool parameters and multi-choice values that are selected.

Remarks

This method ignores multi-choice parameters and instead loops through their the multi-choice values which are regarded as Boolean parameters.

2.3.14.19. PARAMETERVALUETITLE

Returns the title of the lookup parameter's selected value. If the parameter isn't set an empty string is returned.

```
string ParameterValueTitle(LookupParameter lookupParameter)
```

lookupParameter	A lookup parameter whose selected value shall be returned.

Example

`ParameterValueTitle(Length)`

Returns the title of the selected value of the Length parameter. For example the title of the lookup value might be "1 200 mm" meanwhile the actual value is 1200.

2.3.14.20. PARENT

Returns the product title or the value of a named parameter, multi-choice value or attribute of the parent configuration. If no parent configuration or no matching parameter/attribute is found, then an empty string is returned.

```
string Parent([string name])
```

name	Optional name of a parameter, multi-choice value or attribute.

Example

`Parent()`

Returns the title of the parent configuration's product.

`Parent("Color")`

Returns the value of the parameter or attribute named "Color" from the parent configuration.

`Parent("Options.FastDelivery")`

Returns the value of the multi-choice value named "FastDelivery", belonging to the multi-choice parameter "Options", from the parent configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.14.21. PREVSIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations with a lower index.

```
int PrevSiblingsParameterOcc(string name; string value)
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

Example

```
PrevSiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations with a lower index for which the Boolean parameter "Connected" is true.

2.3.14.22. PREVSIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations with a lower index.

```
double PrevSiblingsParameterSum(string name; [string productTitle])
```

name	The name of a parameter.
productTitle	Optional filter to only include configurations of the specified product.

Example

```
PrevSiblingsParameterSum("Length")
```

Returns sum of the double or integer attribute "Length" from all sibling configurations with a lower index.

2.3.14.23. SELECTEDMULTICHOICEVALUES

Returns the names of the selected values of a multi-choice parameter. The values are separated by a space or by a separator defined as the optional second argument. The optional third argument is used for defining another property to return instead of Name.

```
string SelectedMultiChoiceValues(MultiChoiceParameter multiChoicePa-
rameter; [string seperator]; [string propertyName])
```

multiChoiceParameter	A multi-choice parameter.
seperator	Optional separator, instead of space, to add between each selected multi-choice value.

propertyName2	Optional name of a property to read from the multi-choice values instead of the Name property.

Example

```
SelectedMultiChoiceValues(Options; ", "; "Title")
```

Returns the titles of selected multi-choice values separated by a comma and a space.

2.3.14.24. SIBLINGSPARAMETEROCC

Returns the number of occurrences of the specified parameter/value combination among the sibling configurations.

```
int SiblingsParameterOcc(string name; string value; [bool in-
cludeSelf])
```

name	The name of a parameter.
value	The value to find. For Boolean values enter "True" or "False". For numeric values use the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".
includeSelf	Optional argument to include current configuration in the parameter count. Default is false.

Example

```
SiblingsParameterOcc("Connected", "True")
```

Returns the number of sibling configurations for which the Boolean parameter "Connected" is true. Current configuration is not included.

2.3.14.25. SIBLINGSPARAMETERSUM

Returns the sum of the specified numerical parameter/value combination among the sibling configurations.

```
double SiblingsParameterSum(string name; [bool includeSelf]; [string
productTitle])
```

name	The name of a parameter.
includeSelf	Optional argument to include current configuration in the parameter sum. Default is false.

productTitle	Optional filter to only include configurations of the specified product.

Example

```
SiblingsParameterSum("Length"; true)
```

Returns sum of the double or integer attribute "Length" from all sibling configurations including current configuration.

2.3.15. PRODUCT

Expression functions for getting data from configurations' products.

Child
NextSibling
Parent
PrevSibling
Product
ProductF

2.3.15.1. CHILD

Returns the product title or the value of a named parameter, multi-choice value or attribute in the specified child configuration. If no matching child or parameter/attribute is found, then an empty string is returned.

```
string Child(int index; [string paramOrAttrName]; [string productTitle])
```

index	The zero-based index of the configuration to get a value from.
paramOrAttrName	Optional name of a parameter, multi-choice value or attribute.
productTitle	Optional filter to only include child configurations of the specified product.

Example

```
Child(2; "Color")
```

Returns the value of the parameter or attribute named "Color" from the third child configuration.

```
Child(0; "Options.FastDelivery")
```

Returns the value of the multi-choice value named "FastDelivery" belonging to the multi-choice parameter "Options" from the first child configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.15.2. NEXTSIBLING

Returns the product title, a parameter value, or an attribute value from the next sibling configuration. If no argument is attached the product title is returned. If no matching configuration, parameter or attribute is found, then an empty string is returned.

```
string NextSibling([string paramOrAttrName]; [string productTitle])
```

paramOrAttrName	Optional name of the parameter or attribute to read.
productTitle	Optional filter to find the next configuration of the specified product.

Example

`NextSibling("Power")`

Returns the value of the parameter or attribute with the name "Power" from next sibling configuration.

2.3.15.3. PARENT

Returns the product title or the value of a named parameter, multi-choice value or attribute of the parent configuration. If no parent configuration or no matching parameter/attribute is found, then an empty string is returned.

```
string Parent([string name])
```

name	Optional name of a parameter, multi-choice value or attribute.

Example

`Parent()`

Returns the title of the parent configuration's product.

`Parent("Color")`

Returns the value of the parameter or attribute named "Color" from the parent configuration.

```
Parent("Options.FastDelivery")
```

Returns the value of the multi-choice value named "FastDelivery", belonging to the multi-choice parameter "Options", from the parent configuration.

Remarks

Boolean values are returned as "True" or "False". Bool parameters and multi-choice values with "three-state" active can also have the value *indeterminate* which is returned as the empty string "".

It is recommended to place attributes that read the value of other configurations' attributes at the bottom of the attribute tree to reduce the risk of circular dependencies. Another solution is to replace either the referenced or referencing attribute with a parameter that is set using a template expression rule.

2.3.15.4. PREVSIBLING

Returns the product title, a parameter value, or an attribute value from the previous sibling configuration. If no argument is attached the product title is returned. If no matching configuration, parameter or attribute is found, then an empty string is returned.

```
string PrevSibling([string paramOrAttrName]; [string productTitle])
```

paramOrAttrName	Optional name of the parameter or attribute to read.
productTitle	Optional filter to find the previous sibling configuration of the specified product.

Example

```
PrevSibling("Power")
```

Returns the value of the parameter or attribute with the name "Power" from previous sibling configuration.

2.3.15.5. PRODUCT

Returns a property value from the product of current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching property is found, then an empty string is returned.

```
string Product(string propertyName; [string propertyName2]; ...)
```

propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
Product("Title")
```

Returns the title of the configuration's product.

2.3.15.6. PRODUCTF

Returns a property value from the product of current configuration. Properties of reference type can be followed to instead return a property of the referenced object. If no matching property is found, then an empty string is returned.

```
string ProductF(string format; string propertyName; [string propertyName2]; ...)
```

format	Format specifiers to apply when converting the property to string. Search Microsoft's documentation about "format strings".
propertyName	The name of a property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
ProductF("yyyy-MM-dd"; "EffectiveUntil")
```

Returns the effective until date in a format like "2024-12-31" of current configuration's product.

2.3.16. PRICES

Expression functions for getting price and currency data.

CategoryPrice
CurrencyFactor
Price
StoredPrice

2.3.16.1. CATEGORYPRICE

Returns the calculated, summarized amount of the price items of the specified price category within the specified price list.

```
double CategoryPrice(string priceListCategory; string priceCategory)
```

priceListCategory	The name of a price list category, i.e. the name of a price list.

priceCategory	The name of a price category.

Example

```
CategoryPrice(Configuration("PriceListCategoryId"; "Title"); "Options")
```

Returns the calculated, summarized amount of the price items in the price category "Options" within the configuration's active price list.

2.3.16.2. CURRENCYFACTOR

Returns the the factor of the specified currency. The factor is the currency's exchange rate to the default currency, i.e. to the currency with the factor of 1.

```
double CurrencyFactor(string currency)
```

currency	The identity of a currency.

Example

```
CurrencyFactor("GBP")
```

Returns the currency factor of pound sterling. If the default currency is US dollar the factor could for example be 1.25 if one pound is worth more than one dollar.

2.3.16.3. PRICE

Returns the calculated total price for the configuration. If no price list category is specified the price is calculated using the price list that has been chosen for the configuration. If no currency is specified the chosen currency is used.

```
double Price([string priceListCategory]; [string currency])
```

priceListCategory	Optional name of a price list category, i.e. the name of a price list.
currency	Optional identity of a currency.

Example

```
Price("List price")
```

Returns the calculated total "List price" for the configuration.

Remarks

For composite configurations, i.e. configuration that has child configurations, the calculated price includes the price of the child configurations multiplied with their quantity and so on.

2.3.16.4. STOREDPRICE

Returns the stored total price for the configuration in the stored currency.

```
double StoredPrice()
```

Remarks

For composite configurations, i.e. configuration that has child configurations, the stored price includes the price of the child configurations multiplied with their quantity and so on.

2.3.17. REPORTS

Expression functions for controlling how reports are generated.

ActiveDocumentVariant

2.3.17.1. ACTIVEDOCUMENTVARIANT

Returns a property of active report document or active report document variant.

```
string ActiveDocumentVariant(string propertyName)
```

propertyName	The name of a property to read. Currently only "Title" is supported.

Example

```
ActiveDocumentVariant("Title")
```

Returns the title of active report document or active report document variant.

2.3.18. SITES

Expression functions to determine what kind of site the current site is.

InAdminSite
InProdSite

2.3.18.1. INADMINSITE

Returns true if current site is an administration site, i.e. the staging site where changes in configuration models, done using Architect, are tested before being published to the production site.

```
bool InAdminSite()
```

Remarks

In case of a single-site environment that doesn't have an administration site and a production site, both InAdminSite() and InProdSite() returns false.

2.3.18.2. INPRODSITE

Returns true if current site is production site, i.e. the site where real customer quotations and configurations are made.

```
bool InProdSite()
```

Remarks

In case of a single-site environment that doesn't have an administration site and a production site, both InAdminSite() and InProdSite() returns false.

2.3.19. STRING

Expression functions for string manipulations.

2.3.19.1. INDEXOF

Returns the zero-based index of the first occurrence of the specified sub string within the text. The search is done case-sensitive. Returns -1 if the sought string isn't found.

```
int IndexOf(string text; string subString)
```

text	The string to search inside.
subString	The sub string to search for.

Example

```
IndexOf("Lorem ipsum";"ipsum")
```

Returns 6.

2.3.19.2. ISNUMERICAL

Returns true if the text can be converted to a number, otherwise false is returned. Optionally the method can check only for integer values. The conversion attempt is made using the in-variant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

```
double IsNumerical(string text; [bool onlyIntegers])
```

text	The text to check.
onlyIntegers	Optionally check only for integer values. Default false.

Example

```
IsNumerical("-320.77")
```

Returns true.

2.3.19.3. LASTINDEXOF

Returns the zero-based index of the last occurrence of the specified sub string within the text. The search is done case-sensitive. Returns -1 if the sought string isn't found.

```
int LastIndexOf(string text; string subString)
```

text	The string to search inside.
subString	The sub string to search for.

Example

```
LastIndexOf("{"name":"Martin", "age":51}";":")
```

Returns 23.

2.3.19.4. STRCONTAINS

Returns true if the specified sub string is found within the text. The search is done case-sensitive. If the sub string isn't found false is returned.

```
bool StrContains(string text; string subString)
```

text	The string to search inside.
subString	The sub string to search for.

Example

```
StrContains("Lorem ipsum";"ipsum")
```

Returns true.

2.3.19.5. STRLEN

Returns the number of characters in the attached string.

```
int StrLen(string text)
```

text	The string whose length shall be returned.

Example

```
StrLen("Abcdefg")
```

Returns 7.

2.3.19.6. STRREPLACE

Returns a new string in which all occurrences of the specified string in the text are replaced with another specified string. The search is done case-sensitive.

```
string StrReplace(string text; string oldValue; string newValue;
[int startIndex])
```

text	The string to do search and replace inside.
oldValue	The string to be replaced.
newValue	The string to replace oldValue.
startIndex	Optional index for where to start doing replacements.

Example

```
StrReplace("Winter is coming.";"Winter";"Summer")
```

Returns "Summer is coming.".

2.3.19.7. SUBSTR

Returns a sub string from the text. The sub string starts at the specified character position and is optionally limited to the specified length.

```
int SubStr(string text; int startIndex; [int length])
```

text	The string to retrieve the sub string from.
startIndex	The zero-based starting character position for the sub string.
length	Optional limitation in length of the sub string.

Example

```
SubStr("Ice is cold";4;2)
```

Returns "is".

2.3.19.8. TOLOWER

Returns a copy of the string converted to lower case. The conversion is done using the invariant culture. Search Microsoft's documentation for "CultureInfo.InvariantCulture".

```
string ToLower(string text)
```

text	The string to convert to lower case.

Example

```
ToLower("Abcdefg")
```

Returns "abcdefg".

2.3.19.9. TOSTRING

Converts the specified value to its equivalent string representation.

```
string ToString(object value; [string format])
```

value	The value to convert.
format	Optional format string. See Microsoft's documentation for DateTime.ToString(), Double.ToString(), and Int.ToString() for how to define format strings.

Example

```
ToString(Now();"dd/MM/yyyy")
```

Returns the current date in the format dd/MM/yyyy, e.g. "31/07/2024".

2.3.19.10. TOUPPER

Returns a copy of the string converted to upper case. The conversion is done using the invariant culture. Search Microsoft's documentation for "CultureInfo.InvariantCulture".

```
string ToUpper(string text)
```

text	The string to convert to upper case.

Example

```
ToUpper("Abcdefg")
```

Returns "ABCDEFG".

2.3.20. TRANSLATION

Expression functions for translations.

Translate

2.3.20.1. TRANSLATE

Translates a text to the active user's language. If no translation is found the text is returned untranslated

```
string Translate(string keyText; [bool caseSensitive])
```

keyText	A text to translate.
caseSensitive	Optional argument to control if the translation shall be case sensitive or not. The default value is true.

Examples

```
Translate("Delivery terms")
```

Translates the text "Delivery terms" to the user's language.

```
Translate("5 {weeks}")
```

Provided that "replacement control" has been activated the text "weeks" is translated meanwhile "5 " is left untranslated.

2.3.21. TYPE CONVERSIONS

Expression functions for converting data to a specified type.

Date
IsNumerical
ToDouble
ToInt
ToString

2.3.21.1. DATE

Converts a string to a DateTime and returns the result. If the conversion fails the default DateTime is returned, i.e, 0001-01-01 00:00:00.

```
DateTime Date(string dateTime; [string format])
```

dateTime	A string representing a date and optionally a time.

format	Optional format definition for the string to convert.

Example

```
Date("31/12/2023";"dd/MM/yyyy")
```

Returns the date time value of 2023-12-31 00:00:00.

2.3.21.2. ISNUMERICAL

Returns true if the text can be converted to a number, otherwise false is returned. Optionally the method can check only for integer values. The conversion attempt is made using the invariant format. Search Microsoft's documentation about "NumberFormatInfo.InvariantInfo".

```
double IsNumerical(string text; [bool onlyIntegers])
```

text	The text to check.
onlyIntegers	Optionally check only for integer values. Default false.

Example

```
IsNumerical("-320.77")
```

Returns true.

2.3.21.3. TODOUBLE

Converts a specified value to a double-precision floating-point number. Empty strings are converted to zero. Boolean values are converted to zero for false or one for true. All other conversions are done using NumberFormatInfo.InvariantInfo. If the conversion fails an exception is thrown.

```
double ToDouble(object value)
```

value	The value to convert.

Example

```
ToDouble("12.7")
```

Returns the double value 12.7.

2.3.21.4. TOINT

Converts a specified value to a 32-bit signed integer. Empty strings are converted to zero. Boolean values are converted to zero for false or one for true. All other conversions are done using NumberFormatInfo.InvariantInfo. If the conversion fails an exception is thrown.

```
int ToInt(object value)
```

value	The value to convert.

Example

```
ToInt("12")
```

Returns the integer value 12.

2.3.21.5. TOSTRING

Converts the specified value to its equivalent string representation.

```
string ToString(object value; [string format])
```

value	The value to convert.
format	Optional format string. See Microsoft's documentation for DateTime.ToString(), Double.ToString(), and Int.ToString() for how to define format strings.

Example

```
ToString(Now();"dd/MM/yyyy")
```

Returns the current date in the format dd/MM/yyyy, e.g. "31/07/2024".

2.3.22. USERS AND GROUPS

Expression functions for accessing data about the active user and her group memberships.

MemberOf
MemberOfAuthorizationGroup
MemberOfWorkGroup
User
WorkGroup

2.3.22.1. MEMBEROF

Returns true if the active user is a member of a work group or authorization group with the specified title, or else false is returned.

```
bool MemberOf(string groupTitle)
```

groupTitle	The title of a work group or an authorization group.

Example

`MemberOf("Admins")`

Returns true if the active user is a member of a work group or an authorization group with the title "Admins".

2.3.22.2. MEMBEROFAUTHORIZATIONGROUP

Returns true if the active user is a member of an authorization group with the specified title, or else false is returned.

```
bool MemberOfAuthorizationGroup(string groupTitle)
```

groupTitle	The title of an authorization group.

Example

`MemberOfAuthorizationGroup("SalesManagers")`

Returns true if the active user is a member of the authorization group "`SalesManagers`".

2.3.22.3. MEMBEROFWORKGROUP

Returns true if the active user is a member of a work group with the specified title, or else false is returned.

```
bool MemberOfWorkGroup(string groupTitle)
```

groupTitle	The title of a work group.

Example

`MemberOfWorkGroup("HQ")`

Returns true if the active user is a member of the work group "HQ".

2.3.22.4. USER

Returns the property value from the active user. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string User(string propertyName; [string propertyName2]; ...)
```

propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
User("LanguageId"; "Title")
```

Returns the title of the active user's language.

2.3.22.5. WORKGROUP

Returns the property value from the active user's work group. Properties of reference type can be followed to instead return a property of the referenced object. If no matching child or property is found, then an empty string is returned.

```
string WorkGroup(string propertyName; [string propertyName2]; ...)
```

propertyName	Name of the property to read.
propertyName2	Optional name of a property to read from an object that is referenced by the first property.

Example

```
WorkGroup("Title")
```

Returns the title of the active user's work group.