

BAZIS-Script

User Guide

June 03 2021 г.

The information in this document is subject to change without prior notice.

No part of this document may be reproduced or transmitted in any form and in any way for any purpose without written permission from LLC Bazis-Center.

© LLC Bazis-Center, 2021. All rights reserved.

BAZIS, BAZIS-Woodworker, and Bazis logo are the registered trademarks of LLC Bazis-Center.

Other trademarks mentioned in this document belong to their respective owners.

Table of content

General information 19
Demonstration examples of BAZIS-script 19
Scripts editor 19
Editor launch 19
Script editor interface 19
Editor commands 19
Script tab 20
Scrips debug 20
Organization of interaction with user 20
Toolbar control elements 20
Control elements of Properties auxiliary panel 20
Sending messages to the user 21
Language syntax 21
List of keywords of BAZIS-Script programming language: 21
List of reserved words: 22
Comment 23
Variables 23
Constants 24
Arithmetic operations 24
Logical operations 25
Comparison operations 25
Operator if...else 26
Operator switch 27
Conditional expression operator 27
Operators break and continue 28
Loop for 28
Loop while 29
Loop do...while 29
Operator with 30
Operator try...catch...throw 30
Functions 31
Built-in script language objects 32
General information 32
Object Global 32

Object Number 32
Object Boolean 33
Object String and operations on strings 33
Object RegExp (regular expressions) 33
Object Array and work with arrays functions 35
Object Date 38
Object Math 39
Number module 39
Numbers rounding 39
Maximum and minimum determining 40
Exponentiation 40
Square root calculating 40
Random number generator 40
Trigonometry methods 40
Natural logarithm calculating 41
Common for all modules 41
System objects 41
NewVector 41
NewPoint 42
NewContour 42
NewCOMObject 42
NewForm 43
Global constants 43
AxisX 43
AxisY 43
AxisZ 43
Axis_X 43
Axis_Y 44
Axis_Z 44
Global enumerated types 44
TProjection3dEnum 44
PanelSideEnum 44
SalonAnimationType 44
TextureDir 45
WindowPosition 45
FurnSchemeType 45
SchemeFurnPos 46

ErrorType 46
DatumMode 46
FurniturePosition 47
ElementType 47
FurnPositionMode 47
Global properties and methods 48
Global properties 48
system 48
Model 48
Action 48
ActiveMaterial 48
AnimationType 48
Global methods 49
prompt 49
alert 49
confirm 49
OpenFurniture 50
SelectAll 50
UnSelectAll 50
ViewAll 50
SetCamera 51
GetPoint 51
GetObject 51
GetPanel 51
GetEdge 52
AddPanel 52
AddFrontPanel 52
AddHorizPanel 53
AddVertPanel 53
AddExtrusion 54
AddTrajectory 54
AddBlock 55
AddAssembly 55
AddDraftBlock 56
AddCopy 56
DeleteNewObjects 56
DeleteObject 56
StartEditing 57
BeginBlock 57
EndBlock 57
NewButtonInput 57
NewFloatInput 58
NewNumberInput 58
NewMaterialInput 59
NewButtMaterialInput 59

NewFurnitureInput 60
FormatMatName 60
OrientCamera 60
ExtractMatName 60
ExtractMatCode 61
Geometry2D 61
Methods 61
Intersect 61
Compare 61
Distance 62
Area 62
System 62
Methods 63
include 63
log 63
fileExists 63
writeTextFile 63
askWriteTextFile 64
readTextFile 64
askReadTextFile 64
secureExec 64
exec 65
getFileName 66
getFileNameWithoutExtension 66
askFileNameSave 66
askFolder 66
Properties 67
apiVersion 67
developerApiVersion 67
Model3D 67
Properties 68
Selected 68
SelectionCount 68
Selections 68
Count 68
Objects 68
Action3D 69
Properties 70
Interactive 70
MouseX 70
MouseY 70
Pos3 71
ViewDir 71
UpDir 71

RightDir . . . 71
ShowPoints . . . 71
ShowEdges . . . 72
Hint . . . 72
ErrorHint . . . 72
BlinkHint . . . 72
OnClick . . . 72
OnMove . . . 73
OnStart . . . 73
OnFinish . . . 73
Properties . . . 73
OnRayTraceFinished . . . 73
ModelFilename . . . 74
Methods . . . 74
Continue . . . 74
Commit . . . 74
Finish . . . 74
Cancel . . . 74
AsyncExec . . . 74
BeginOrtho3 . . . 75
EndOrtho3 . . . 75
CursorToClosestPoint . . . 75
CursorToClosestLine . . . 76
CursorToMiddleOfLine . . . 76
Find3DPoint . . . 76
Find3DPointXZPlane . . . 76
Get3Dobject . . . 76
RayTraceScene . . . 76
LoadModel . . . 76
SaveModel . . . 77
NewModel . . . 77
NewFurniture . . . 77
NewFragment . . . 77
Revert . . . 77
ArrangePositions . . . 77
ReplaceFurniture . . . 78
ChooseMaterial . . . 78
ScriptProperty . . . 78
Properties . . . 79
Name . . . 79
Enabled . . . 80
ChildrenEnabled . . . 80
Visible . . . 80
OnChange . . . 80
Count . . . 80
Items . . . 80

Expanded . . . 81
OnDeactivate . . . 81
OnValueValidate . . . 81
ValueValid . . . 81
NameEditable . . . 81
BackColor . . . 82
PopupMenu . . . 82
DropDownMenu . . . 82
OnActivate . . . 82
OnValueChange . . . 82
Tag . . . 83
Methods . . . 83
Clear . . . 83
NewCOMObject . . . 83
Save . . . 83
Load . . . 83
NewGroup . . . 84
NewImage . . . 84
NewString . . . 85
NewBool . . . 85
NewNumber . . . 85
NewButton . . . 86
NewMaterial . . . 86
NewButt . . . 86
NewFurniture . . . 87
NewColor . . . 87
NewSeparator . . . 87
Validate . . . 87
ScriptGroupProperty . . . 88
Properties . . . 88
Image . . . 88
MaxHeight . . . 88
Scrollable . . . 88
ScriptStringProperty . . . 89
Properties . . . 89
Value . . . 89
ScriptBooleanProperty . . . 89
Properties . . . 89
Value . . . 89
ScriptNumberProperty . . . 89
Properties . . . 89
MaxValue . . . 89
MinValue . . . 90
ValueStep . . . 90

Value. 90
ScriptButtonProperty 90
Properties 91
Value. 91
OnClick 91
ScriptMaterialProperty 91
Properties 91
Width 91
Methods 91
SetActive 91
ScriptButtProperty 92
Properties 92
Thickness 92
Width 92
Methods 92
SetActive 92
ScriptFurnitureProperty 92
Properties: 93
Value. 93
ScriptColorProperty 93
Properties 93
Value. 93
ScriptMenu 93
Properties 94
Name 94
OnChange 94
Count 94
Ite ms94
Store. 94
Methods 95
Clear 95
Save 95
Load 95
NewGroup. 95
NewString. 96
New Bool96
NewNumber 96
NewButton 96
ScriptForm 96
Properties 97
Properties 97

Caption . . . 97
Width . . . 97
Height . . . 98
MinWidth . . . 98
MinHeight . . . 98
Visible . . . 98
Left . . . 98
Top . . . 99
OKButton . . . 99
OKButtonCaption . . . 99
CancelButton . . . 99
CancelButtonCaption . . . 99
OnClose . . . 99
OnShow . . . 100
Resizable . . . 100
Dockable . . . 100
OnOkButtonClick . . . 100
OnCancelButtonClick . . . 100
Methods . . . 101
Show . . . 101
ShowModal . . . 101
Close . . . 101
FurnMaterial . . . 101
Properties . . . 101
Name . . . 101
Thickness . . . 102
Width . . . 102
Methods . . . 102
Make . . . 102
Vector . . . 102
Properties: . . . 102
x . . . 102
y . . . 103
z . . . 103
Point . . . 103
Properties: . . . 103
x . . . 103
y . . . 103
Edge3 . . . 104
Properties . . . 104
First . . . 104
Last . . . 104
GFirst . . . 104
GLast . . . 104

Object3 105

Properties: 106

Name 106
ArtPos 106
Owner 106
Visible 107
Selected 107
List 107
AsList 107
AsPanel 107
Position 108
PositionX 108
PositionY 108
PositionZ 108
GMax 108
GMin 109
GSize 109
GabMin 109
GabMax 109
UserPropCount 109
UserProperty 110
UserPropertyName 110

Methods 110

SetDefaultTransform 110
Translate 110
Rotate 111
TranslateGCS 111
RotateGCS 111
RotateX 111
RotateY 112
RotateZ 112
Orient 112
OrientGCS 113
Reflect 113
ToObject 114
ToGlobal 114
NToObject 114
NToGlobal 115
Build 115
IsOwner 115

List3D 115

Properties: 116

Count 116
IsElastic 116
Objects 116

FurnType . . . 117
SalonType . . . 117
Methods: . . . 117
ElasticResize . . . 117
Load . . . 117
Panel . . . 117
Properties . . . 118
Contour . . . 118
Butts . . . 118
Cuts . . . 119
Plastics . . . 119
ContourHeight . . . 119
ContourWidth . . . 119
Thickness . . . 119
MaterialName . . . 120
MaterialWidth . . . 120
Methods . . . 121
AddButt . . . 121
AddCut . . . 121
AddPlastic . . . 122
IsButtVisible . . . 122
Extrusion . . . 122
Properties . . . 123
Contour . . . 123
Thickness . . . 123
MaterialName . . . 123
Trajectory . . . 124
Properties . . . 124
Contour2D . . . 124
Trajectory2D . . . 124
MaterialName . . . 124
Example: . . . 125
Block . . . 125
Properties: . . . 125
IsFastener . . . 125
Assembly . . . 125
Properties . . . 125
An mType . . . 125
PanelButts . . . 126
Properties . . . 126
Count . . . 126
Methods . . . 127

Butts[] 127
Add 127
PanelButt 127
Properties 128
ElemIndex 128
Sign 128
Material 128
Thickness 128
Width 128
ClipPanel 129
Overhung 129
Allowance 129
CutIndex 129
Profile 129
PanelPlastics 130
Properties 130
Count 130
Methods 130
Plastics[] 130
Add 131
PanelPlastic 131
Properties: 131
Material 131
Thickness 131
TextureOrientation 131
PanelCuts 132
Properties 132
Count 132
Methods 132
Add 132
Cuts 133
PanelCut 133
Properties 133
Name 133
Sign 133
Trajectory 133
Contour 134
Contour2D 134
Properties 136
Count 136
Objects 136
Width 136

Height . . . 137
Min . . . 137
Max . . . 137
Methods . . . 137
Clear . . . 137
Move . . . 137
Rotate . . . 138
AddRectangle . . . 138
AddRoundRect . . . 138
AddLine . . . 138
AddCircle . . . 138
AddArc . . . 139
AddArc3 . . . 139
AddEquidistantRecursive . . . 139
Subtraction . . . 140
Addition . . . 140
RoundingEx . . . 140
FacetEx . . . 140
Rounding . . . 141
Facet . . . 141
Find . . . 141
Fit . . . 141
Elastic . . . 142
Symmetry . . . 142
Load . . . 142
OrderContours . . . 142
AddEquidistant . . . 143
IsPointInside . . . 143
IsInContour . . . 143
IIsClosedContour . . . 143
IsContourRectangle . . . 144
InControl . . . 144
Properties . . . 144
id . . . 144
Enabled . . . 144
Visible . . . 144
Hint . . . 145
OnChange . . . 145
InButton . . . 145
Properties: . . . 145
Caption . . . 145
Combo . . . 145
Methods: . . . 146
NewSubMenu . . . 146

InFloat 146
Properties 146
Value 146
ReadOnly 146
Fixed 146
InNumber 147
Properties 147
Value 147
ReadOnly 147
Fixed 147
InMaterial 147
Properties 148
Name 148
Thickness 148
Width 148
Methods 148
SetActive 148
Apply 148
InButtMaterial 149
Properties 149
Name 149
Sign 149
Thickness 149
Width 150
Overhung 150
Allowance 150
ClipPanel 150
InFurniture 150
Properties 150
DatumModeFilter 150
DatumMode 151
Methods 151
Mount 151
Mount1 151
MountScheme 151
MountBox 152
ModelInspector 152
Properties 152
ErrorList 152
Options 152
Methods 153
Run 153

InspectorOptions 153
Properties 153
ObjIntersectionAnalyze 153
FastIntersectionAnalyze 153
FastIncorrectAnalyze 154
PanelNotFixedAnalyze 154
PanelTooLargeAnalyze 154
PlasticTooLargeAnalyze 154
MatNotExistsAnalyze 154
MatOutOfStockAnalyze 154
InspectorError 155
Properties 155
ErrorType 155
ErrorObjectsCount 155
ErrorObjects 155
ErrorMessage 155
ObjectsNames 156
ImportExportSVG 156
Properties 156
CurveQuality 156
GroupElems 156
Methods 156
Save 156
Load 157
ImportExport 157
Properties 157
SVG 157
InfFurniture 157
Properties 158
DatumModeFilter 158
DatumMode 158
Methods 158
Mount 158
Mount1 158
MountScheme 159
MountBox 159
Make 159
EncodeToString 160
DecodeFromString 160
Choose 160
RootProperties 160
Methods 160

NewFurnitureValue 160
Elem2D 161
Properties 161
ElType 161
Methods 161
IsLine 161
AsLine 162
IsCircle 162
AsCircle 162
IsEllipse 162
AsEllipse 162
IsArc 162
AsArc 163
IsList 163
AsList 163
ObjLength 163
Line2D 163
Properties 164
Pos1 164
Pos2 164
Arc2D 164
Properties 164
Pos1 164
Pos2 164
Center 165
ArcDir 165
Circle2D 165
Properties 165
Center 165
CirRadius 165
Dir 166
Ellipse2D 166
Properties 166
Center 166
MajorRadius 166
MinorRadius 166
MajorAxisAngle 167
Dir 167
FurnArticle 167
Properties 167
Name 167
OrderCode 168

OrderName . . . 168
DatumMode . . . 168
Methods . . . 168
NameWithCode . . . 168
BAZIS-Woodworker . . . 168
Global constants of BAZIS-Woodworker . . . 168
p3dFront . . . 168
p3dLeft . . . 169
p3dRight . . . 169
p3dTop . . . 169
p3dBottom . . . 169
p3dIsometric . . . 169
BAZIS-Estimate . . . 169
Global properties of BAZIS-Estimate . . . 169
Panel . . . 169
Holes . . . 169
Material . . . 170
Result . . . 170
Objects . . . 170
EstimateObject . . . 170
Methods . . . 171
AsPanel . . . 171
IsPanel . . . 171
AsButt . . . 171
IsButt . . . 171
AsPlastic . . . 171
IsPlastic . . . 172
AsExtrusionBody . . . 172
IsExtrusionBody . . . 172
AsTrajectoryBody . . . 172
IsTrajectoryBody . . . 172
EstimateObjectList . . . 173
Properties . . . 173
Count . . . 173
Items . . . 173
PanelHoles . . . 173
Properties . . . 173
Count . . . 173
Holes . . . 173

PanelHole 174
Properties 174
Depth 174
Diameter 174
Door design wizard 174
DoorsMaker 174
Methods 175
Silent 175
ShowErrors 175
Save 175
Load 175
Setup 175
Drawer design wizard 176
BoxesMaker 176
Methods 176
ShowErrors 176
Save 176
Load 176
Setup 176
BAZIS-Salon 177
TovarItems 177
Properties 177
TovarName 177
TovarModel 177
Price 178
IsNotStandart 178
KodTovID 178
Metka 178
Discount 178
Markup 179
Methods 179
FindByName 179
TScItemTovar 179
Properties 180
Article 180
Name 180
Count 180
Material 180
Price 180
GroupMaterial 181
TypeElement 181
ObjList 181

SalonUtils 181
Properties 182
PathAttachments 182
SalonId 182
Methods 182
GetFullPathAttachment 182
UserProperties 182
Properties 183
Value 183
Name 183
Count 183
Methods 183
Remove 183

Appendix I.
Screen form creation in visual mode 185

General information

BAZIS-Scripts are furniture design automation tools for application programmer on the basis of BAZIS-Woodworker module. BAZIS-Scripts have Java scripts format and they are saved in files of *js* extension.

Demonstration examples of BAZIS–script

BAZIS system delivery set includes demonstration examples of scripts use. Using these examples you can create your own scripts.

During BAZIS-Woodworker module installation script files will be automatically saved in folder *C:\Users\<CurrentUserName>\Documents\Bazis\Scripts*, where *X* stands for BAZIS system version number.

Scripts editor

Editor launch

Commands for operation with scripts are placed in **Scripts** section of Main menu in BAZIS-Woodworker module. **Script editor** command allows to open to open editor window. Other section commands coincide with script file names. If you click on script name with mouse left button, it will be automatically started. If you click on it, hold *<Ctrl>* button pushed, **Script editor** window will be opened and script text will be viewed (fig. 1).

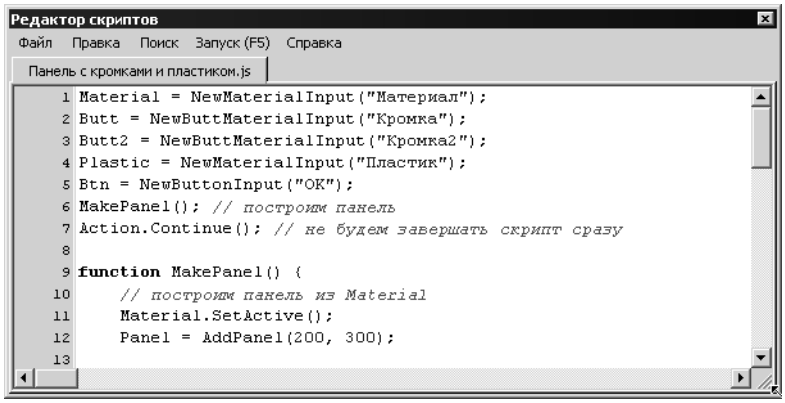


Fig. 1.

Script editor interface

Editor commands

Main menu commands are standard and they allow to perform operations with script files, as well as to close editor window. Several scripts can be opened in editor simultaneously. The texts are placed on window tabs. One of the tabs is active. Editor commands can be applied to the script, which text is on this tab. **Launch** command allows to start active script.

Script tab

Script editor is a text editor. Script commands syntax is highlighted.



Script text can be created in any third-party text editor.

Scrips debug



To debug the scripts use Visual Studio Code with Debugger for Bazis-Woodworker's scripts plugin. To install a plugin start Visual Studio Code and open **Extensions** section. Enter *bazissoft.bazis-debug2* line in search field and install the plugin.

Organization of interaction with user

User interaction with the program can be performed using the following methods:

- control elements (buttons, input field, etc.) on toolbar,
- control elements on **Properties** auxiliary panel,
- sending messages to the user, including those that require their reaction.

For simple commands that have a small number of parameters, it is reasonable to place control elements on the toolbar. The standard BAZIS system commands can be used as analogue, e.g. **Operations** group commands. At the same time «hook-eys» are automatically assigned to speed up access to them:

- for access to buttons *<Shift>+1, <Shift>+2,* etc.
- for access to input fields *<Alt>+1, <Alt>+2,* etc.

During complicated scripts creation, designed for parametrical models creation, it is reasonable to arrange control elements on **Properties** auxiliary panel. This allows to group them including on several levels.

Toolbar control elements

To create control elements on toolbars, use the following methods:

- NewButtonInput
- NewFloatInput
- NewNumberInput
- NewMaterialInput
- NewButtMaterialInput
- NewFurnitureInput

Control elements of Properties auxiliary panel

To use control elements of **Properties** auxiliary panel, use the following methods and properties of Action.Properties object:

NewGroup – Create nested property group.

NewImage – Create nested property group with image**.**

NewString – Create property of String type.

NewBool – Create property of Yes/No type.

NewNumber – Create property of Number type.

NewButton – Create property with editing button.

NewMaterial – Create property of Material type.

NewButt – Create property of Edge material type.

NewFurniture – Create property of Accessories type.

NewColor – Create property of Color type.

NewSeparator – Create separator.

PopupMenu – Popup menu.

DropDownMenu – Dropdown menu.

Sending messages to the user

Messages are sent to the user using the following methods:

alert – Display message or parameters values.

confirm – Display confirmation and cancel dialogue.

Hint – Set hint line.

ErrorHint –Set error message.

Language syntax

The syntax of the scripting language is similar to C++ and Java. But any programming language has its own specific representation of key and reserved words, that make up the core for programming in this language. Key words are always available to developer, but he should follow the correct syntax.

List of keywords of BAZIS-Script programming language:

- **break**
- **switch**
- **for**
- **typeof**
- **continue**
- **try**
- **new**
- **while**
- **const**
- **finally**

- **case**
- **this**
- **return**
- **throw**
- **in**
- **default**
- **var**
- **else**
- **if**
- **catch**
- **do**
- **function**
- **with**

List of reserved words:

- **boolean**
- **double**
- **goto**
- **interface**
- **byte**
- **enum**
- **implements**
- **long**
- **char**
- **export**
- **import**
- **native**
- **throws**
- **float**
- **int**
- **short**

Comment

Comments include a part of the script code, that will never be executed by the program, however it is very useful for further development. For example using comments you can easily find a section of a previously created program or allow other users to do so. Comments may start with symbols <//>, or you can comment a code section, begin with comment opening symbol </*> and close with symbol <*/> respectively.

Examples:

box(0,0,0,100,100,100,"box1"); //parallelepided construction. Everything that is written after sign <//> is a code comment.

/

This code section will never be executed, it is required to understand what was done in the script, and for program section debugging...

**/*

The entire code section beginning with symbol </*> and ending with <*/> is ignored by the program and is a comment.

Variables

For variable creation use key word **var**.

Examples:

var x; //create variable x without initialization

var x=100; //create variable x equal to 100 of integer type, possible values from -2147483648 to 2147483647.

var x,y=100; //create promptly several variables

var x = new Array(4); //create array with 4 elements (x[0],x[1],x[2],x[3])

var x = "The text to display on the screen\nOn the second line - information\nOn the third line..."; //create variable with text content

var x = true; //create logical content variable true or false

var x = 1516.2298; //creating a real-type variable with possible values from -1.17549435E-38 to 3.40282347E+38.



Attention! For scripts correct work variable names should begin with lowercase or capital Latin letter or underscore character; the name should not contain special characters, e.g.: !, ?,| etc; name should not match the language's keywords. If you specify incorrect variable name, during script execution, message about an error and its description will appear.

When you create a text variable, you can use special characters shown in tab. 1.

Tab.1.

Character	Description
\b	Returns one character and deletes it
\t	Horizontal tab
\n	New line
\v	Vertical tab
\r	Carriage return
\"	Double quote
\'	Single quote
****	Backslash

Constants

For constant creation use keyword **const**. The difference between constants is that their value cannot be changed. The main advantage of these types is fewer bytes occupied by these variables and, as a result, faster code execution when using **const** wherever possible.

Examples:

const message = "Constant text"; //create text type constant

Arithmetic operations

- Addition. E.g. x+y
- Subtraction. E.g. x-y
- Multiplication . E.g. x*y
- Division. E.g. x/y
- Remainder of division. E.g.: x % y

For ease of use you can apply abbreviated entries for these operations in the script language (tab. 2).

Tab.2.

Operation	Description
x += y	Abbreviation x = x + y
x –= y	Abbreviation x = x - y
x *= y	Abbreviation x = x * y

Tab.2.

Operation	Description
x /= y	Abbreviation x = x / y
x %= y	Abbreviation x = x % y
++x	Abbreviation x = x + 1. Increase by 1 before assignment
--x	Abbreviation x = x - 1. Decrease by 1 before assignment
x++	Abbreviation x = x + 1. Increase by 1 after assignment
x--	Abbreviation x = x - 1. Decrease by 1 after assignment

Examples:

var i = 0;

var a = ++i; //In this example variable i will be increased by 1 and then the result will be assigned to variable. Result: a = 1.

var i = 0;

var a = i++; //In this example variable i value will be assigned to variable a and then it will be increased by 1. Result: a = 0.

Logical operations

- Logical "AND" - &&
- Logical "OR" - ||

Examples:

var i = 0;

var f = 5;

if (f>i && f!=0) {} //Result: true because first and second condition - true

var i = 10;

var f = 15;

if (i==f || f==0) {} //Result: false, because both conditions are false.

Comparison operations

Such operations are necessary for comparing one variable or constant with another. There is a list of all possible operations in tab. 3:

Tab.3.

Operation	Description
x == y	Returns true, if x and y are equal
x != y	Returns true, if x and y are not equal
x === y	Returns true, if x and y are equal and their types match
x < y	Returns true, if x is less than y
x <= y	Returns true, if x is less or equal to y
x > y	Returns true, if x is greater than y
x >= y	Returns true, if x is greater or equal to y

Examples:

true==1 // Result: true (true is converted to 1)

true===1 // Result: false (true is not converted to 1)

5 <= "5" // Result: true (line "5" is converted to a number)

5 === "5" // Result: false (line "5" is not converted to a number)

5 > 4 // Result: true (5 is greater than 4)

Operator if...else

The principle of operation is as follows: first **if** condition is checked and if it is true, the following code is run, and if not, then **else** code is running in record, if it is specified

Examples:

```
if (10 > 5)
{
//Here we execute the code if the condition is true
}
else
{
//Here we execute the code if the condition didn't work
}

if (10 < 11)
```

{

//Here we execute the code if the condition is true

} //Operator else is omitted, therefore, if the condition does not work, no code will be executed

Operator switch

Selection is made based on specified value. This function is a replacement for **if ... else** operators and is more convenient when you need to iterate over a set of values

Example:

```
var num = 10;
switch (num)
{
case 1:
  //Execute the code, if num == 1
  break;
case 2:
  //Execute the code, if num == 2
  break;
case 100:
  //Execute the code, if num == 100
  break;
default:
  //Execute the code, if the value was not found
}
```

Conditional expression operator

A very compact and convenient function for selecting one of the two available values. Principle of operation:

result = (condition) ? value1(if condition is true) : value2(if condition didn't work)



In C/C++ language this operator is slower than the usual **if ... else** operator. However, when writing scripts the issue of optimization is not so acute and therefore this form of writing is common.

Example:

var S = (1 > 0) ? 1500 : 1600; //Accordingly in this expression S = 1500, because condition is true

Operators break and continue

- **break** – The command is used inside the loop for force loop termination.
- **continue** – Skip the incomplete part of the loop and continue running the loop with the following values.

Example:

```
var S = "Object_name1";
var FindedName = "";
for (var i = 0; i < 10; ++i)
{
  var tested = "Object_name" + i;
   if (tested != S) continue; //Continue searching with the following values
   else
   {
   FindedName = tested;
  break; //Forcibly stop the search, because the object name has already been found and the loop is no longer needed
  }
}

  if (FindedName != "")
   infomsg("Find object by name: " + FindedName);
  else
   warning("Object not found");
```

Loop for

It is a convenient and flexible way to use loops.

Principle of operation:

```
for ( variable_name and its starting value; maximum variable value; variable change )
{
//Loop body
}
```

Example:

```
var num = 0;
for (var i = 0; i < 10; ++i)
```

{

num += i; //Each time the loop passes, num value increases by i value

}

history(num); //Put the result in the command history

//It is easy to guess that after executing such num program code = 45 (0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45).

Loop while

Analogue of **For** loop organization operator; When using **while** there is no initialization of the variable and its changes.

Principle of operation:

while (condition)

{

//Loop body

}

Example:

//The previous example with operator for can be written as follows:

var num = 0, i = 0; //Initialization of variable i in contrast to For loop occurs in advance

while (i < 10)

{

num += i; //Each time the loop passes, num value increases by the value i

++i; //In contrast to For loop we change variable i value in loop body

}

history(num); //The result is the same as in the previous example num = 45

Loop do...while

A type of **while** loop. The difference is that regardless of **while** condition, the loop is executed at least once.

Principle of operation:

do

{

//Loop body

} while (condition)

Example:

```
//The previous example with for operator can be written as follows:
var num = 0, i = 0; //Initialization of variable i in contrast to For loop occurs in advance
do
{
  num += i; //Each time the loop passes the num value increases by the value i
  ++i; //In contrast to For loop, we change variable i value in loop body
} while (i < 10)
history(num); //The result is the same as in the previous example num = 45
```

Operator with

Required for repeated access to the same object.

Example:

```
//Without with operator:
history(Math.PI);
history(Math.abs(-2));
history(Math.max(4, 10, 7, 6));

//With operator with:
with(Math)
{
history(PI);
history(abs(-2));
history(max(4, 10, 7, 6));
}
```

Operator try...catch...throw

Error trapping of program code execution.

Example:

```
try
{
//Program section,  which may cause an execution error
var S = 100;
if (S > 99)
   throw ("S value is too large: " + S); //Make a throw
```

```
//Here you can continue to run the program
}

catch (...)
{
//This code will be executed if an error occurs
critical(e); //Output an error message
}

finally
{
//The code is executed in any case
infomsg("Completed the program");
}
```

Functions

Create custom functions in script file

Principle of operation:

```
function NameOfFunc(arg1,arg2...argn)
{
//Function code
return 0; //The return value is 0 or any other value
}
```

Example:

```
function helloUser(name)
{
infomsg("Good time of day, "+name)
}

var name = GetPar("Your name");
helloUser(name);
```



You can create function without parameters, and if necessary, transfer parameters to it and find out their value using **arguments** array inside the function.

Example:

```
function multiply() //Multiplication function
{
var result = 1;
for (var i = 0; i < arguments.length; ++i)
  result *= arguments[i]; //Multiply the arguments among themselves

return result;
}

infomsg("The result of multiplying 2,3,5,7: " + multiply(2,3,5,7));
```

Built-in script language objects

General information

BAZIS-Script language provides a series of built-in objects:

- global object **Global**,
- object **Object**,
- object **Function**,
- object **Array**,
- object **String**,
- object **Boolean**,
- object **Number**,
- object **Math**,
- object **Date**,
- object **RegExp**,
- error objects **EvalError**, **RangeError**, **ReferenceError**, **SyntaxError**, **TypeError** and **URiError**.

Object Global

This is the base object of the script language that sets the properties and methods of global namespace. It is a top-level object and does not have a parent object. It defines constructor properties for all objects used in the script. We can extend this object with any new functions and methods.

Object Number

This object is used for numbers storing. It is very easy to create it:

var myVar = 1 (the number of integer type)

or

var myVar = 12.1125 (the number of real type).

Among the methods of this object:

- isNaN(variable); //Returns true, if variable is not a number
- isInfinite(variable); //Returns true, if variable value is greater than allowed limits of Number object
- parseFloat(variable); //Returns a number of real type by the variable value
- parseInt(variable); //Returns an integer number by variable value

Object Boolean

The object is used for storing the values true and false. It is very easy to create it:

var myVar = true or var myVar = false.

Object String and operations on strings

The object is used for text values storing. It is very easy to create it:

var myVar = "text value".

Operations onstrings:

- Function for string conversion to lower case: toLowerCase();
- Function for string conversion to upper case: toUpperCase();
- Part of string replace: replace("previous value","new value");
- Getting a character: charAt(num); where num - symbol number
- String length: length
- Getting a string part: substring(start,nmb); where start - is the beginning of string reading, nmb - number of characters

Examples:

```
var myStr = "Here is my test String";
history("My lowercase string looks like this: " + myStr.toLowerCase());
history("My uppercase string looks like this: " + myStr.toUpperCase());
history("Words replacing: " + myStr.replace("test","current"));
history("Second symbol value: " + myStr.charAt(1));
history("String length: " + myStr.length);
history("The first four letters of the string: " + myStr.substring(0,4));
```

Object RegExp (regular expressions)

Regular expressions are required to check whether strings match the specified template. Such expressions in scripts must be set in the #C style, that is, enclosed in slashes / and /; or using the new operator.

There are special characters for regular expressions in tab. 3.1.

Tab. 3.1. Special characters of regular expressions

Character	Description
\	Normal characters escaping. Allows you to use special characters in templates.
^	Indicates the beginning of the input data.
$	Indicates the end of the input data.
*	Repeat from zero to infinity.
+	Repeat from one to infinity.
?	Indicates that the template element may or may not be present. The question symbol is also used when grouping regular expression elements.
.	Any character other than a newline: \n \r \u2028 or \u2029.
(x)	Finds and remembers the specified string.
(?:x)	Finds the specified string, but does not remember it.
x(?=y)	Finds the specified string x, provided that x is followed by y.
x(?!y)	Finds the specified string x, provided that x is not followed by y.
x|y	Finds x or y.
{n}	Exact number of repetitions.
{n,}	Number of repetitions from n to infinity.
{n,m}	Number of repetitions from n to m.
[xyz]	Finds the specified character set.
[^xyz]	Excludes the specified set of characters from the search.
[\b]	Finds the backspace character.
\b	Finds a word boundary, such as a space.
\B	Eliminates the layers boundary.
\cX	Where X is the letter from A to Z. Indicates a control character in a string.
\d	Finds any number.
\D	Excludes any number from the search.
\f,\r,\n	Corresponding special characters form-feed, line-feed, newline

Tab. 3.1. Special characters of regular expressions

Character	Description
\s	Finds any whitespace characters, including the space itself, newline, etc.
\S	Excludes whitespace characters from the search.
\t	Tab character.
\v	Vertical tab character.
\w	Any letter.
\W	Excluding any letter.
\n	A backlink to the n-th substring stored with brackets.
\0	NUL character.
\xhh	Finds specified character in two-digit hexadecimal format.
\uhhhh	Finds a Unicode character by its four-digit code.

Examples:

var myRegexp = /([A-Za-z]+)=(\d+)/; //Declaring a regular expression in #C style

var myRegexp = new RegExp("([A-Za-z]+)=(\d+)"); //Declaring a regular expression using operator new

Regular expressions made a breakthrough in electronic text processing at the end of the XX century, however, BASIS scripts are more focused on performing geometric constructions and calculations. Therefore, regular expressions are not discussed in detail in this document. However, here are some examples of text processing using regular expressions.

Examples:

var regexp = '1234567890'.replace(/(\d)(?=(\d\d\d)+(?!\d))/g, '$1,'); //Divide the number in ternary (thousands) group

history(regexp);

Object Array and work with arrays functions

This object is used for arrays creating and editing. There are two ways to create an array in the script language:

- creating an array through initialization.

  var arr = ["value1", 10 , false, "text"]; //As the exaqmple shows, the array can contain different types of values.

- creating an array using the new operator.

  var arr = new Array("value1","value2","value3");



When programming scripts, the array is actually Std::Map, i.e. that you don't have to specify array elements with an ID number. You can also set them with a text value and access such array elements via a text value respectively.

Example:

var myMassive = new Array(); //Create an empty array

myMassive["first"] = "value1"; //Add "value1" to myMassive array with name "first"

myMassive["second"] = "value2"; //Add "value2" to myMassive array with name "second"

var myVar1 = myMassive["first"]; //Accessing an array element with name "first"

var myVar2 = myMassive["second"]; //Accessing an array element with name "second"

You can use the following methods to add elements to an array:

- Element (s) adding to the end of the array: push(value1,value2...valueN);
- Element (s) adding to the beginning of the array: unshift(value1,value2...valueN);
- Element (s) adding to the specified location in the array (startInsert) with possible replacement of subsequent elements (nmbToRemove): splice(startInsert,nmbToRemove,value1,value2...valueN);
- You can also add array elements as follows: arr[3] = "value"; that is to the specified location in the array

Examples:

var myMassive = new Array("value1", 10 , false, "text"); //Create an array with four values

myMassive[4] = "new value"; //Add a fifth value to the array. Note how the array is numbered: the size of the array has become equal to 5 elements, and access to them occurs from 0 to 4, respectively. Result: myMassive("value1", 10 , false, "text","new value")

myMassive.push("push value1","push value2"); //I add 2 more elements to the end of the array using the push() method. Result: myMassive("value1", 10 , false, "text","new value","push value1","push value2")

myMassive.unshift("some first1"); //Add 1 element to the beginning of the array. Result: myMassive("some first1","value1", 10 , false, "text","new value","push value1","push value2")

myMassive.splice(2,6,"Inserted Value1","Inserted Value2"); //Delete elements of the array from 2 to 6 and insert two new elements. Result: myMassive("some first1","value1","Inserted Value1","Inserted Value2");

var S = "";

for (var i = 0; i < myMassive.length; ++i)

S += "\"" + myMassive[i] + "\" "; //The text will be displayed on the screen: "some first1" "value1" "Inserted Value1" "Inserted Value2"

infomsg(S);

To change the order of an array, use function reverse();

Example:

//Changing the array order:

var myMassive = new Array("first","second","third"); //Create array with three values

var reversedMassive = myMassive.reverse(); //Change the order of the array to reverse

var S = "";

for (var i = 0; i < reversedMassive.length; ++i)

S += "\"" + reversedMassive[i] + "\" "; //The text will be displayed on the screen: "third" "second" "first"

infomsg(S);

In the previous examples, we convert an array to a string using the *for* loop, but there is an easier way to get a string from an array - the join("connectionSymbol") function, where connectionSymbol is any text or space.

Example:

//Retrieve a string from an array:

var myMassive = new Array("first","second","third"); //Create array with three values

var myStr = myMassive.join(" "); //Combine array elements separated by a space into a string

infomsg(myStr);

To join arrays, there is a concat function(array_name)

Example:

//Two arrays union

var myMassive1 = new Array("first","second","third"); //Create array with three values

var myMassive2 = new Array("four","five"); //Create array with two values

var resultMassive = myMassive1.concat(myMassive2);//Combine array

var myStr = resultMassive.join(" "); //Combine array elements separated by a space into a string

infomsg(myStr);//We get the following output: first second third four five

To sort an array, use sort() function

Example:

// Array sorting

var myMassive = new Array("B","A","C",0,19,-10); //Create array

var sortedMassive = myMassive.sort(); //Get a sorted array

var myStr = sortedMassive.join(" "); //Combine array elements separated by a space into a string

infomsg(myStr);//We get the following output: -10 0 19 A B C

Object Date

This object is used for dates and time storing. To get the current date, use constructor:var myDate = new Date(); to process other date values, you can also set year, month, date, hours, minutes, seconds, and milliseconds: var prevDate = new Date(2010,6,18,13,00,30).

To read values from the object Date, use the functions:

- getYear(),
- getMounth(),
- getDay(),
- getHours(),
- getMinutes(),
- getSeconds(),
- getMilliseconds();

The following functions are used to set the values of the object Date:

- setYear(),
- setMounth(),
- setDay(),
- setHours(),
- setMinutes(),
- setSeconds(),

- setMilliseconds();

Example:

// Output of the current day of the week

var myMassive = new Array("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"); //Create array with all days of the week

var curDate = new Date(); //Read the current date

infomsg("Today: " + myMassive[curDate.getDay() - 1]);//Display the current day of the week

Object Math

The object allows to perform certain mathematical operations in scripts. The object contains a number of constants (tab. 3.2).

Tab. 3.2. Mathematical constants

Constant	Description
E	Number E. Euler's constant value (2.718281828459045)
LN2	Natural logarithm of number 2 (0.6931471805599453)
LN10	Natural logarithm of number 10 (2.302585092994046)
LOG2E	Logarithm of number E to base 2 (1.4426950408889633)
LOG10E	Logarithm of number E to base 10 (0.4342944819032518)
PI	Number PI (3.141592653589793)
SQRT1_2	Square root of number 1/2 (0.7071067811865476)
SQRT2	Square root of number 2 (1.4142135623730951

Access to these constants: Math.E; Math.LN2, etc.

Number module

Returns a positive value of any number using function Math.abs(value);

Example:

var someValue = -133.1527;

var result = Math.abs(someValue); //Exctract the number module

infomsg("Extracted the number module" + someValue + ". Result: " + result);//Display

Numbers rounding

Three number rounding modes are used:

- Math.ceil(number); – returns an integer greater than or equal to the passed value
- Math.floor(number); – returns an integer less than or equal to the passed value

- Math.round(number); – rounds a number to the nearest integer value

Example:

var someValue = 5.6;

var result1 = Math.ceil(someValue); //Result: 6

var result2 = Math.floor(someValue); //Result: 5

var result3 = Math.round(someValue); //Result: 6

history(result1 + " " + result2 + " " + result3);//Display

Maximum and minimum determining

- Math.max(number1,number2); - returns a larger number
- Math.min(number1,number2); - returns a smaller number

Example:

var someValue1 = 5.6;

var someValue2 = 10.653;

var result1 = Math.max(someValue1,someValue2); //Result: 10.653

var result2 = Math.min(someValue1,someValue2); //Result: 5.6

history(result1 + " " + result2);//Display

Exponentiation

Math.pow(number, degree indicator);

Example:

var result = Math.pow(2,3); //Raise 2 to the 3rd degree. Result: 8

history(result);//Display

Square root calculating

Math.sqrt(number);

Example:

var result = Math.sqrt(16); //Square root extraction from 16. Result: 4

history(result);//Display

Random number generator

Math.random();//Returns a random number from 0 to 1

Example:

*var result = Math.floor(Math.random()*1000); //Returns an integer from 0 to 1000*

history(result);//Display

Trigonometry methods

- Math.cos(rad);//Extracting the cosine of the angle set in radians

- Math.sin(rad);//Extracting the sine of an angle set in radians
- Math.tan(rad);//Extracting the tangent of an angle set in radians
- Math.acos(value);//Extracting an angle in radians by its cosine
- Math.asin(value);//Extracting an angle in radians by its sine
- Math.atan(value);//Extracting an angle in radians by its tangent

Scripts do not work with angles set in degrees, so before performing trigonometric calculations, these angles must be converted to radians or vice versa for the results of inverse trigonometric functions:

- var rad = angle * Math.PI / 180;//Converting an angle to radians rad.
- var angle = rad / Math.PI * 180;//Converting the angle set in radians rad to degrees angle.

Example:

//Finding the sine of the 270 degree angle (-90)

*var rad = -90 * Math.PI / 180; //Convert the angle to radians*

history(Math.sin(rad)); //Display: -1



Math object does not have a function for calculating the cotangent. You can implement it yourself: var ctg = Math.cos(angle) / Math.sin(angle).

Natural logarithm calculating

Math.log(value);//The function returns the natural logarithm of a number value

Example:

var x = Math.log(Math.E); //The natural logarithm of E

history(x); //Display: 1



Math object does not have a function for calculating decimal logarithms. You can implement it yourself: var log10 = Math.log(value) / Math.log(10)

Common for all modules

System objects

NewVector

Get a 3D point.

Syntax:

Vector (x, y, z);

Input parameters:

Values of coordinates x, y, z.

Returned value:

Vector

Example:

p = V

ector (100, 100, 200); //Set a point with coordinates x=100, y=100, z=200

NewPoint

2D point.

Syntax:

Point(x,y);

Input parameters:

Values of coordinates x, y.

Returned value :

Point.

Example:

p = Point (100, 100); //Set a point with coordinates x=100, y=100

NewContour

Create flat contour.

Syntax:

NewContour;

Input parameters:

No.

Returned value:

Contour2D

Example:

Hole = NewContour();

Note.

Contour created by NewContour command, can be used for contours editing using Subtraction and Addition commands.

NewCOMObject

Create new COM object by its type.

Syntax:

NewCOMObject(CLSID);

Input parameters:

CLSID

Output parameters:

Object

Example:

```
Word = NewCOMObject('Word.Application');
Word.Visible = true;
Word.Documents.Add();
Word.Selection.TypeText('Hello Bazis\n');
```

NewForm

Create a new screen form.

Syntax:

NewForm;

Input parameters:

No.

Returned value:

ScriptForm

Example:

```
DlgBox = NewForm();
```

Note.

The screen form can be created in visual mode.

Global constants

AxisX

Axis X.

Data type:

Vector

AxisY

Axis Y.

Data type:

Vector

AxisZ

Axis Z.

Data type:

Vector

Axis_X

Axis X, negative direction.

Data type:

Vector

Axis_Y

Axis Y, negative direction.

Data type:

Vector

Axis_Z

Axis Z, negative direction.

Data type:

Vector

Global enumerated types

TProjection3dEnum

View direction

View direction. Coincide with the main views.

Value	Description
p3dFront	Front view.
p3dLeft	Left-hand view.
p3dRight	Right-hand view.
p3dTop	Top view.
p3dBottom	Bottom view.
p3dIsometric	Axonometry.

PanelSideEnum

Panel side.

Value	Description
true	Face side.
false	Back side.

SalonAnimationType

Assembly animation types.

Value	Description
None	Not accounted in salon.

Value	Description
Custom	Considered in salon and has no animation.
DoorLeft	Left door.
DoorRight	Right door.
DoorFlap	Drop door.
DoorLift	Vertical lifting door.
SDoorLeft	Sliding wardrobe left door.
SDoorRight	Sliding wardrobe right door.
Drawer	Drawer.
Support	Leg.
Handle	Handle.
Facade	Front panel.

TextureDir

Panel plastic texture orientation.

Value	Description
None	Absent.
Horizontal	Horizontal.
Vertical	Vertical.

WindowPosition

Form position.

Value	Description
Default	Standard position of the form.
Left	The form is joined from the left
Right	The form is joined from the right

FurnSchemeType

Map type of fasteners arrangement.

Value	Description
WithBase	Map type with a fixed margin.
Symmetric	Symmetric map type.

Value	Description
VariableStep	Variable step map type.

SchemeFurnPos

Accessories mounting modes.

Value	Description
Inside	Accessories mounting inside the joint
Outside	Accessories mounting outside the joint
Up	Accessories mounting at the top of the joint (only for joints with a horizontal panel)
Down	Accessories mounting at the bottom of the joint (only for joints with a horizontal panel)

ErrorType

Error type of model analysis

Value	Description
ObjIntersection	Objects intersection
FastIntersection	Fasteners intersection
FastIncorrect	Accessories incorrect mounting
MatNotExists	Material not available
MatOutOfStock	Material not available in stock
PanelTooLarge	The panel cannot be placed on the board
PlasticTooLarge	Plastic cannot be placed on the panel
PanelNotFixed	Panel is not fixed

DatumMode

Type of accessories/fragment mounting.

Value	Description
None	Missing
Face	Onto plane
FaceFace	On two nonparallel planes
FaceButt	Along the plane and panel edge middle

Value	Description
FaceEdge	On face and edge
ParallelFaces	On 2 parallel planes
Box	Section
Scheme	Fasteners by map

FurniturePosition

Accessories position when installing the fasteners by map.

Value	Description
Inside	Accessories mounting inside the joint
Outside	Accessories mounting outside the joint
Up	Accessories mounting at the top of the joint (only for joints with a horizontal panel)
Down	Accessories mounting at the bottom of the joint (only for joints with a horizontal panel)

ElementType

Element 2D type.

Value	Description
Unknown	Unknown
Line	Line
Arc	Arc
Circle	Circle
List	List of elements
Ellipse	Ellipse

FurnPositionMode

Positions layout map.

Value	Description
All	Place positions anew
New	Place positions of new objects

Value	Description
Check	Check positions

Global properties and methods

Global properties

system

System functions.

Syntax:

system;

Data type:

System

Model

Model structure

Syntax:

Model;

Data type:

Model3D;

Action

Active script

Syntax:

Action;

Data type:

Action3D

ActiveMaterial

Current material

Syntax:

ActiveMaterial;

Data type:

FurnMaterial

AnimationType

Assemblies and blocks animation types.

Syntax:

AnimationType;

Data type:

SalonAnimationType

Global methods

prompt

Text parameter input in the dialogue.

Syntax:

prompt('<hint>');

Input parameters:

hint – is a hint displayed in status bar.

Data type:

String

Output parameters:

Text entered in the dialogue.

Example:

```
name  = prompt('Enter panel name');
alert(name);
```

alert

Display a message or parameter values.

Syntax:

alert('<message text>');

Input parameters:

Message text, string, or parameter name which value should be displayed on the screen. It is possible to combine several parameters using symbol +.

Output parameters:

No.

Example:

```
alert ('Length = '+len1);
```

confirm

Display the confirmation or cancellation dialogue.

Syntax:

confirm('<query text>');

Input parameters:

Query text, string.

Data type:

Boolean

Example:

confirm('Construct model?');

OpenFurniture

Open accessory file to install in the model.

Syntax:

OpenFurniture(<file name>);

Input parameters:

<file name> – accessory file name.

Return result:

InFurniture.

Example:

f = OpenFurniture('C:\\Program Files\\BazisSoft\\Bazis 8\\Samples\\Samples\\L shaped bracket.f3d');



- <file name> can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.
- After adding an accessory item, you must install it using Mount or Mount1 commands.

SelectAll

Select all objects.

Syntax:

SelectAll();

UnSelectAll

Unselect all objects.

Syntax:

UnSelectAll();

ViewAll

View all.

Syntax:

ViewAll();

Description

The command allows to assign the maximum possible scale for model displaying so that all objects fit in the window.

SetCamera

Set the current view.

Syntax:

SetCamera(TProjection3d);

Input parameters:

TProjection3dEnum

GetPoint

Point request.

Syntax:

GetPoint('<hint>'),

Input parameters:

<hint> – the query string that is displayed in the status bar.

Return result:

Vector

Example:

point = GetPoint('Specify hole center');

GetObject

Model object request.

Syntax:

GetObject('<hint>'),

Input parameters:

<hint> – the query string that is displayed in the status bar.

Return result:

Object3

Example:

Obj = GetObject('Choose object');

GetPanel

Panel request.

Syntax:

GetPanel('<hint>'),

Input parameters:

<hint> – the query string that is displayed in the status bar.

Return result:

Panel

Example:

```
Pan1 = GetPanel('Choose panel');
```

GetEdge

Request to select an edge parallel to the specified vector.

Syntax:

```
GetEdge('<hint>'),
```

Input parameters:

<hint> – request text displayed in the status bar.

Return result:

Edge3

Example:

```
Obj = GetEdge('Specify the edge');
```

AddPanel

Add panel.

Syntax:

```
AddPanel (width, height)
```

Input parameters:

width – panel width,

height – panel height.

Return result:

Panel

Example:

```
VPanel = AddPanel (400, 600);
```

Description

After the command execution, a panel with the specified parameters is added to the model. The panel is located in the plane that is closest to the plane parallel to the screen. The panel vertex is located at the origin.

AddFrontPanel

Add front panel.

Syntax:

AddFrontPanel (x1, y1, x2, y2, z);

Input parameters:

x1, y1 – initial point coordinates,

x2, y2 – end point coordinates,

z – distance from the origin to the panel on Z axis.

Return result:

Panel

Example:

FPanel = AddFrontPanel (40, 40, 640, 700, 300);

Description

After the command execution, a panel with the specified parameters is added to the model.

AddHorizPanel

Add horizontal panel.

Syntax:

AddHorizPanel(xl, zl, x2, z2, y);

Input parameters:

x1, z1 – initial point coordinates,

x2, z2 – end point coordinates,

y – distance from the origin to the panel on Z axis.

Return result:

Panel

Example:

HPanel = AddHorizPanel (40, 40, 640, 700, 300);

Description

After the command execution, a panel with the specified parameters is added to the model.

AddVertPanel

Add vertical panel.

Syntax:

AddVertPanel(z1, y2, z2, y2, x);

Input parameters:

z1, y1 – initial point coordinates,

z2, y2 – end point coordinates,

x – distance from the origin to the panel on Z axis.

Return result:

Panel

Example:

VPanel = AddVertPanel (40, 40, 640, 700, 300);

Description

After the command execution, a panel with the specified parameters is added to the model.

AddExtrusion

Add extruded boss.

Syntax:

AddExtrusion('name');

Input parameters:

name – object name, string.

Return result:

Extrusion.

Example:

Profile = AddExtrusion ("Profile");

Description

After command execution, an extruded boss with specified name is added to the model. The boss is initially "empty". You must set its contour and thickness.

AddTrajectory

Add a kinematic element (body along the trajectory).

Syntax:

AddTrajectory('name');

Input parameters:

name – trajectory name.

Return result:

Trajectory.

Description

Trajectory is a body that is obtained by moving the profile contour along the trajectory contour. After command execution, an empty trajectory is added to the model. You must set the contour shape and trajectory of contour move.

Example:

```
Traj = AddTrajectory('Trajectory');
p1 = Vector(0, 0, 0);
p2 = Vector(40, 50, 70);
p3 = Vector(100, 150, 120);
Traj.Trajectory2D.AddArc3(p1,  p2,  p3);
Traj.Contour2D.AddRoundRect(30, 50, 50, 70, 8);
```

AddBlock

Add block.

Syntax:

AddBlock("Block name");

Input parameters:

Block name, string.

Return result:

Block

Description

After command execution, an empty block with the specified name is added to the model.

Example:

```
Block = AddBlock ("Block");
```

AddAssembly

Add assembly.

Syntax:

AddAssembly("<Assembly name>");

Input parameters:

Assembly name, string.

Return result:

Assembly

Example:

```
Asm = AddAssembly ("Assembly");
```

Description:

After command execution, an empty assembly with the specified name is added to the model.

AddDraftBlock

Create semiproduct.

Syntax:

AddDraftBlock("<name>");

Input parameters:

name, string.

Return result:

Block

Example:

Draft = AddDraftBlock("Semiproduct");

Description:

After command execution, an empty assembly with the specified name is added to the model.

AddCopy

Create object copy.

Syntax:

AddCopy(<obj>);

Input parameters:

obj – copied object.

Output parameters:

Object3

DeleteNewObjects

Delete objects created in script.

Syntax:

DeleteNewObjects();

DeleteObject

Delete object from the model.

Syntax:

DeleteObject(obj);

Input parameters:

obj – deleted object.

Example:

p1 = AddPanel(100, 200);

```
p2 = AddPanel(140, 300);
if (confirm("Delete?")) {
  DeleteObject(p1);
}
```

StartEditing

Start editing an object that is in the model. For this purpose call the command so that changes made to the model during editing can be canceled by **Cancel(Ctrl+z)** command.

Syntax:

StartEditing(obj);

Input parameters:

obj – model object.

BeginBlock

Start block creation. All objects created after this command and before EndBlock fall into the block.

Syntax

BeginBlock("Name_block");

Input parameters:

Block name.

Output parameters:

Block.

EndBlock

Finish block creation. All objects created after BeginBlock command and before EndBlock fall into the block.

Syntax

EndBlock();

Example:

```
BeginBlock("Block");
AddFrontPanel(0, 0, 50, 50, 0);
AddFrontPanel(0, 0, 50, 50, 100);
EndBlock();
```

NewButtonInput

Button creation on Property manager.

Syntax:

NewButtonInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InButton

Example:

```
p = NewButtonInput(‘Button1’);
n = NewButtonInput(‘Button2’);
true;
function $input(id)
{
if (p.id == id)
alert(p.id)
else
alert (n.id);
}
```

NewFloatInput

Real number input.

Syntax:

NewFloatInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InFloat.



You can enter mathematical expressions.

Example:

p = NewFloatInput (‘Enter number’);

NewNumberInput

An integer input.

Syntax:

NewNumberInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InNumber



You can enter mathematical expressions, e.g. 45*3. The result is rounded to an integer.

Example:

p = NewNumberInput (‘Enter number’);

NewMaterialInput

Material input.

Syntax:

NewMaterialInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InMaterial.

Example:

```
k = NewMaterialInput("Choose material");
true;
function $input()
{
alert(k.Thickness);
alert(k.Name);
alert ('Width = '+k.Width);
return true;
}
```

NewButtMaterialInput

Enter edgebanding material.

Syntax:

NewButtMaterialInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InButtMaterial.

Example:

k = NewButtMaterialInput ("Choose material");

NewFurnitureInput

Create control element for accessories choice.

Syntax:

NewFurnitureInput (’<hint>’);

Input parameters:

<hint> – the query string that is displayed in the status bar.

Output parameters:

InFurniture

Example:

k = NewFurnitureInput ("Choose accessory");

FormatMatName

Create fasteners mounting map

Syntax:

FormatMatName(<matName>);

Input parameters:

<matName> – material name, string.

Output parameters:

string

OrientCamera

Create fasteners mounting map

Syntax:

OrientCamera(dir);

Input parameters:

<dir> – new view direction, Vector.

Output parameters:

no

ExtractMatName

Extract material name from the full name.

Syntax:

ExtractMatName(<matName>);

Input parameters:

<matName> – material full name, string.

Output parameters:

string

ExtractMatCode

Extract material article from the full name.

Syntax:

ExtractMatCode(<matName>);

Input parameters:

<matName> – material full name, string.

Output parameters:

string

Geometry2D

A system object to implement the auxiliary geometric algorithms.

Methods:

Intersect – Calculate the intersection points of two-dimensional elements.

Compare – Compare elements.

Distance – Find the shortest distance between elements.

Area – Find contour area.

Methods

Intersect

Calculate the intersection points of two-dimensional elements.

Syntax:

Intersect(elem1,elem2);

Input parameters:

elem1,elem2 – intersecting 2D elements.

Returned value:

Array.<Point>

Compare

Compare elements.

Syntax:

Compare(elem1,elem2);

Input parameters:

elem1,elem2 – 2D elements.

Return value:

Boolean

Distance

Find the shortest distance between elements.

Syntax:

Distance(elem1,elem2);

Input parameters:

elem1,elem2 – 2D elements.

Return value:

Number

Area

Find contour area.

Syntax:

Area(contour);

Input parameters:

contour – 2D contour, Contour2D.

Return value:

Number

System

System object

Methods:

include – Connect specified file JavaScript.

log – Display a diagnostic message (for debugging).

fileExists – Check whether the specified file exists.

writeTextFile – Write text to a file.

askWriteTextFile – Write text to a file with a file name request.

readTextFile – Read text from a file.

askReadTextFile – Read text from a file with a file selection request.

secureExec – To execute encrypted code.

exec – Run an external program.

askFolder – Open folder selection dialogue.

getFileName – Get a file name without path.

askFileNameSave – Open file saving dialogue.

getFileNameWithoutExtension – Get the file name without path and extension.

Properties:

apiVersion – Current version of Bazis API.

developerApiVersion – The maximum version of Bazis API, that guarantees the script work.

Methods

include

To connect the specified file JavaScript.

Syntax:

include(<file.js>);

Input parameters:

<file.js> – Name of the file to connect.

log

Display a diagnostic message (for debugging).

Syntax:

log(<msg>);

Input parameters:

<msg> – Message text.

fileExists

Check whether the specified file exists.

Syntax:

fileExists(<file>);

Input parameters:

<file> – file name, including the full path or path relative to the current folder where the script file is located.

writeTextFile

Write text to a file.

Syntax:

writeTextFile(<file>,<content>);

Input parameters:

<file> – file name, including the full path or path relative to the current folder where the script file is located.

<content> – file content.

askWriteTextFile

Write text to a file with a file name request in the dialogue.

Syntax:

askWriteTextFile(<ext>,<content>);

Input parameters:

<ext> – file name extension.

<content> – file content.

readTextFile

Read text from a file.

Syntax:

readTextFile(<file>);

Input parameters:

<file> – file name, including the full path or path relative to the current folder where the script file is located.

askReadTextFile

Read twxt from a file with a request to select a file in the dialogue.

Syntax:

askReadTextFile(<ext>);

Input parameters:

<ext> – file name extension.

secureExec

To execute encrypted code.

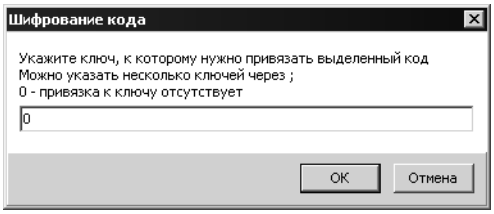
Syntax:

secureExec(str);

Input parameters:

str – encrypted script text.

This method allows to execute an encrypted script. Current script text can be encrypted via Script editor command **Edit** – **Code encryption**. Before command execution, highlight the script text that you want to encrypt. If nothing is selected, the entire script is encrypted. After command call **Code encryption** dialogue will appear on the screen. In this dialogue, you can enter key number of BAZIS system hardware protection. A script that is "linked" to the key in this way can only be executed in BAZIS-Woodworker module, which is protected by this key. If the number is not specified (key number = 0), there is no linking, the script can be executed in any BAZIS-Woodworker module. You can enter several key numbers separated by a sign <;>.



As a result of command execution, the script will be automatically changed in the following way:

- system.secureExec() command is added;
- encrypted text is used as its parameter.

Encrypted script example:

```
system.secureExec("" +
   "AQAA/wF4nG2OMQ7CMAxFfxkYkLiDBwa6lKEj6oAQ" +
   "M0gcgKEtlgMJKi0L6ik4MPy4JVSALcfJz0v8R+hi" +
   "zNq7psrLJyPyUvS+enBZIUcNAweLBGvt/mzRoMQc" +
   "MZaYYqK1YB6p28EbwUzpC/Va+buyorQhL1QNCvaM" +
   "KWQdDqyKfBz+lS8nG64FtkqdqKUDL55vg6t/js7c" +
   "3Xr/EhwJXSbY8e5K9sNnP9M7Ju3ntXgBBekyhA==" +
   "");
```

All further actions with the script (saving, launch, etc.) are standard.

exec

Run an external program.

Syntax:

exec(str);

Input parameters:

str – full name of executive file.

Example:

system.exec('c:/Windows/system32/calc.exe');

getFileName

Get a file name without path.

Syntax:

getFileName(filename);

Input parameters:

filename – file full name, string.

Returned value:

file name without path, string.

getFileNameWithoutExtension

Get a file name without path and extension.

Syntax:

getFileNameWithoutExtension(filename);

Input parameters:

filename – file full name, string.

Returned value:

file name without path and extension, string.

askFileNameSave

Open dialogue for file saving.

Syntax:

askFileNameSave(ext);

Input parameters:

ext – file name extension, string.

Returned value:

file full name, string.

askFolder

Open folder selection dialogue.

Syntax:

askFolder(caption, defaultFolder);

Input parameters:

caption – folder title, string

defaultFolder – default folder, string

Returned value:

folder name, string.

Properties

apiVersion

Bazis API current version

Syntax:

apiVersion;

Data type:

Number

Note.

system.apiVersion = 90 -> Bazis9

system.apiVersion = 85 -> Bazis8

Example:

alert(system.apiVersion);

developerApiVersion

The maximum version of Bazis API, that guarantees the script work

Syntax:

developerApiVersion;

Data type:

Number



Bazis API version (system.apiVersion) now has the Number type and is subject to "Semantic versioning" specification in XXYY format, where XX is the major version and YY is the minor version. It starts with the number 1001.
Minor version (YY, in this case 01) - increases when changes are made in API that are compatible with previous versions or functions are added. For example: a script written in version 1001 will work in version 1003. But it is not guaranteed that a script written in version 1003 will work in version 1001.
Major version (in this case 10) – increases when changes are made in API that are incompatible with the previous version. E. g.: a script written in version 10XX is likely to work incorrectly in version 11XX and vice versa.

Model3D

List of model objects.

Properties:

Selected – The model selected object.

SelectionCount – Number of selected elements.

Selections – List of selected elements.

Count – Number of objects in the model.

Objects – List of model objects.

Properties

Selected

Selected model object. If several objects are selected, then the first of them.

Syntax:

Selected;

Data type:

Object3

SelectionCount

Number of selected elements.

Syntax:

SelectionCount;

Data type:

Number

Selections

List of selected elements.

Syntax:

Selections;

Data type:

Array.Object3

Count

Number of objects in the model.

Syntax:

Count;

Data type:

Number

Objects

Model objects list

Syntax:

Objects;

Data type:

Array.Object3

Action3D

Current script.

Properties

Interactive – Ability to interact with the user.

MouseX – Coordinate X of the mouse cursor position on the screen.

MouseY – Coordinate Y of the mouse cursor position on the screen.

Pos3 – Current cursor position in the model window.

ViewDir – Normal to the current view.

UpDir – Up vector of the current view.

RightDir – Vector to the right of the current view.

ShowPoints – Allow to highlight the points.

ShowEdges – Allow to highlight the edges.

Hint – Set the hint string.

ErrorHint – Install an error message.

BlinkHint –Set a blinking hint.

OnClick – Mouse click handler.

OnMove – Mouse cursor movement handler.

OnFinish – The script completion handler.

Properties – Set of editable properties.

OnRayTraceFinished – image rendering completion.

ModelFilename – Current model file name.

Methods

Continue – Continue calling interrupts at the end of the main body of the script, without completing the command.

Commit – Apply changes to the model made in the script.

Finish – Finish command.

Cancel – Cancel command.

AsyncExec – Execute a function where Get* requests are available.

BeginOrtho3 – Enable the mode for orthogonal constructions relative to a point.

EndOrtho3 – Disable the mode for orthogonal constructions relative to a point.

CursorToClosestPoint – Move the cursor to the nearest snap point.
CursorToClosestLine –Move the cursor to the nearest line.
CursorToMiddleOfLine –Move the cursor to the nearest midpoint of the line.
Find3DPoint – Find a point on the model in the current position of the cursor.
Find3DPointXZPlane – Find a point on the plane X0Z.
Get3DObject – Find an object under the cursor.
RayTraceScene – Launch image rendering.
LoadModel – Load a model from a file.
SaveModel – Save a model to a file.
NewModel – Create a new model.
NewFurniture – Create new accessories.
NewFragment – Create new fragment.
Revert – Cancel changes to the model made in the script.
ArrangePositions – Place positions.
ReplaceFurniture – Replace accessories.
ChooseMaterial – Call material selection window from material database.

Properties

Interactive

Ability to interact with the user. If false, any user interaction functions are forbidden.

Syntax:

Interactive;

Data type:

Boolean

MouseX

X coordinate of the cursor position on the screen.

Syntax:

MouseX;

Data type:

Number

MouseY

Y coordinate of the cursor position on the screen.

Syntax:

MouseY;

Data type:

Number

Pos3

Current cursor position in the model window.

Syntax:

Pos3;

Data type:

Vector

ViewDir

Normal to the current view

Syntax:

ViewDir;

Data type:

Vector

UpDir

Up vector of the current view.

Syntax:

Updir;

Data type:

Vector

RightDir

Vector to the right of the current view

Syntax:

RightDir;

Data type:

Vector

ShowPoints

Allow to highlight the points.

Syntax:

ShowPoints;

Data type:

Boolean

ShowEdges

Allow to highlight the edges.

Syntax:

ShowEdges;

Data type:

Boolean

Hint

Set the hint string.

Syntax:

Hint;

Data type:

String

ErrorHint

Set an error message.

Syntax:

ErrorHint;

Data type:

String

BlinkHint

Set a blinking hint.

Syntax:

BlinkHint;

Data type:

String

OnClick

Mouse click handler.

Syntax:

OnClick;

Data type:

Function

Example:

```
Action.OnClick = function() {
   alert('Click in the model area ' + Action.MouseX + ' ' + Action.MouseY)
```

};

OnMove

Handler of cursor movement.

Syntax:

OnMove;

Data type:

Function

Example:

```
Action.OnMove = function() {
alert('Cursor move')
};
```

OnStart

Handler of script work start. Called after loading property values.

Syntax:

OnStart;

Data type:

Function

OnFinish

Script completion handler.

Syntax:

OnFinish;

Data type:

Function

Properties

Set of editable properties.

Syntax:

Properties;

Data type:

RootProperties

OnRayTraceFinished

Handler for image rendering completion.

Syntax:

OnRayTraceFinished;

Data type:

Function

ModelFilename

File name of the current model.

Syntax:

ModelFilename;

Data type:

string

Methods

Continue

Continue interrupts processing after the main body of the script finishes, without completing the script.

Syntax:

Continue();

Example:

```
x = NewButtonInput("Button");
Action.Continue();
```

Commit

Apply changes to the model made in the script without completing the script.

Syntax:

Commit();

Example:

```
Commit();
```

Finish

Finish script work.

Syntax:

Finish();

Cancel

Complete the script without changes.

Syntax:

Cancel();

AsyncExec

Execute a function where Get* requests are available.

Syntax:

AsyncExec(func);

Input parameters:

func – function to execute (Function).

BeginOrtho3

Enable the mode for orthogonal constructions relative to a point.

Syntax:

BeginOrtho3(point);

Input parameters:

point – button (Vector)

Example:

```
p = AddPanel(100, 100);//mount the panel
pos = GetPoint("Specify a point");//specify a point for panel positioning
Action.BeginOrtho3(pos);//enable orthogonal construction mode.
Action.Continue();
function $move() {
  p.Position = Action.Pos3;
}
```

Note.



- In the example the panel follows the cursor. After enabling the orthogonal construction mode, if the angle between the direction from the *pos* point to the current cursor position and an imaginary line parallel to one of the coordinate axes is less than the angle set at the orthogonality zone setup, the cursor and the snap point of the panel respectively will be located on this imaginary line.
- To disable the orthogonal construction mode, use method EndOrtho3.

EndOrtho3

Disable orthogonal construction mode relative to a point (see method BeginOrtho3).

Syntax:

EndOrtho3();

CursorToClosestPoint

Move the cursor to the nearest snap point.

Syntax:

CursorToClosestPoint();

CursorToClosestLine

Move the cursor to the nearest line

Syntax:

CursorToClosestLine();

CursorToMiddleOfLine

Move the cursor to the nearest midpoint of the line.

Syntax:

CursorToMiddleOfLine();

Find3DPoint

Find a point on the model in the cursor current position.

Syntax:

Find3DPoint();

Return result:

Vector

Find3DPointXZPlane

Find a point on the plane X0Z.

Syntax:

Find3DPointXZPlane();

Return result:

Vector

Get3Dobject

Find an object under the cursor.

Syntax:

Get3DObject();

Return result:

Object3

RayTraceScene

Launch image rendering.

Syntax:

RayTraceScene();

LoadModel

Load a model from a file.

Syntax:

LoadModel(filename);

Input parameters:

filename (string)

Return result:

boolean

SaveModel

Save model to file.

Syntax:

SaveModel(filename);

Input parameters:

filename (string)

NewModel

Create new model.

Syntax:

NewModel();

NewFurniture

Create new accessory.

Syntax:

NewFurniture();

NewFragment

Create new fragment.

Syntax:

NewFragment();

Revert

Cancel changes to the model made in the script.

Syntax:

Revert();

ArrangePositions

Place positions.

Syntax:

ArrangePositions(Mode, Model);

Input parameters:

Mode(FurnPositionMode) – positioning mode

Model(List3D) – structural object

Return result:

boolean

ReplaceFurniture

Replace accessories.

Syntax:

ReplaceFurniture(Old, New, Fasteners);

Input parameters:

Old – list of old accessories names, string

New – list of new accessories values, InfFurniture

Fasteners – list of objects to replace, Object3

Return result:

no

ChooseMaterial

Material selection window call from material database.

Syntax:

ChooseMaterial();

Return result:

Material full name, string.

ScriptProperty

Set of editable properties.

Properties:

Name – Property name.

Enabled – Ability to change the property by the user.

ChildrenEnabled – Availability of nested properties for editing.

Visible – Property visibility in the property panel.

OnChange – Handler for property and nested properties changing.

Count – Number of nested properties of the first nesting level.

Items – List of nested properties of the first nesting level.

Expanded – Whether nested properties are expanded.

OnDeactivate – Handler for property or menu deactivating.

OnValueValidate – Validating the value.

ValueValid – Checkbox for entered value correctness, set by the handler OnValueValidate.

NameEditable – Allow name editing (for table creation).

BackColor – Background color

PopupMenu – Create a context-sensitive menu.

DropDownMenu – Create an objects combo box.

OnActivate – Handler for property activation by the user

OnValueChange – Handler for property changes

Tag – User number

Methods:

Clear – Clear nested properties.

NewCOMObject – Create a new COM object by its type.

Save – Save the property values entered by the user in an xml file.

Load – Load property values saved by method Save from a file of xml format.

NewGroup – Create a nested property group.

NewImage – Create a nested property group with an image.

NewString – Create a String type property.

NewBool – Create a property of Yes/No type.

NewNumber – Create a property of Number type.

NewButton – Create a property with an edit button.

NewMaterial – Create a property of Material type.

NewButt – Create a property of Edge material type.

NewFurniture – Create a property of Accessories type.

NewColor – Create a property of Colour type.

NewSeparator – Create a separator.

Validate – Check the property and nested properties value')

Properties

Name

Property name.

Syntax:

Name;

Data type:

String

Enabled

Ability to change the property by the user.

Syntax:

Enabled;

Data type:

Boolean

ChildrenEnabled

Availability of nested properties for editing.

Syntax:

ChildrenEnabled;

Data type:

Boolean

Visible

Property visibility in the property panel.

Syntax:

Visible;

Data type:

Boolean

OnChange

Syntax:

Handler for properties and nested properties changing.

Syntax:

OnChange;

Count

Number of nested properties of the first nesting level.

Syntax:

Count;

Data type:

Number

Items

Property of the first nesting level.

Syntax:

Items[i];

Input parameters:

i – property index, starting from 0.

Data type:

ScriptProperty

Expanded

Whether nested properties are expanded.

Syntax:

Expanded;

Data type:

Boolean

OnDeactivate

Handler for property or menu deactivating.

Syntax:

OnDeactivate;

Data type:

Function

OnValueValidate

Validating the value.

Syntax:

OnValueValidate;

Data type:

Function

ValueValid

Checkbox for entered value correctness, set by the handler OnValueValidate.

Syntax:

ValueValid;

Data type:

Boolean

NameEditable

Allow name editing (for table creation).

Syntax:

NameEditable;

Data type:

Boolean

BackColor

Background colour

Syntax:

BackColor;

Data type:

Number

PopupMenu

Popup menu.

Syntax:

PopupMenu;

Data type:

ScriptMenu

DropDownMenu

Dropdown menu.

Syntax:

DropDownMenu;

Data type:

ScriptMenu

OnActivate

Handler for property activation by the user.

Syntax:

OnActivate;

Data type:

Function

OnValueChange

Handler for property change.

Syntax:

OnValueChange;

Data type:

Function

Tag

Custom number.

Syntax:

Tag;

Data type:

Number

Methods

Clear

Clear nested properties.

Syntax:

Clear();

NewCOMObject

Create new COM object by its type.

Syntax:

NewCOMObject(CLSID);

Input parameters:

CLSID

Returned value:

IDispatch

Save

Save user-entered property values in xml format file.

Syntax:

Save('<file name>.xml');

Input parameters:

<file name>.xml



- Parameter <file name> can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.
- A set of values saved in a file can be loaded using Load method.

Load

Load property values saved by Save method from an xml file.

Syntax:

Load('<file name>.xml');

Input parameters:

<file name>.xml



Parameter <file name> can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.

NewGroup

Create a nested property group.

Syntax:

NewGroup('<GroupName>');

Input parameters:

<GroupName> –name of created group.

Output parameters:

ScriptGroupProperty

NewImage

Create a nested property group with image**.**

Syntax:

NewImage:('<GroupName>','imagefile');

Input parameters:

<GroupName> – group name.

imagefile – image file name.

Output parameters:

ScriptGroupProperty

- Parameter imagefile can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.
- The following image file formats are supported:
  - vector:
  - Windows MetaFile (extension .wmf, .wmz);
  - Enhanced MetaFile (extension .emf, .emz);
  - bitmap:
  - JPEG (extension .jpg);
  - Windows Bitmap (extension .jpg);
  - Portable Network Graphic (extension .png);
  - Tagged Image File Format (extension .tif).

NewString

Create a String type property.

Syntax:

NewString('propName', 'default_val');

Input parameters:

propName – Property name.

default_val – Property default value.

Output parameters:

ScriptStringProperty

NewBool

Create a property of Yes/No type.

Syntax:

NewBool('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptBooleanProperty

NewNumber

Create a property of Number type.

Syntax:

NewNumber('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptNumberProperty

NewButton

Create a property with an edit button.

Syntax:

NewButton('propName','default_val'),

Input parameters:

propName – Property name.

default_val – Property default value.

Output parameters:

ScriptButtonProperty

Note.

You need to create a button click handler function.

Example:

```
o = Action.Properties;
m = o.NewGroup('Material');
g = m.NewMaterial('Upper shelves');
v12 = g.NewButton('Fragment name', 'test.frw');
v12.OnClick = function() {v12.Value = system.askFileName('frw')};
```

NewMaterial

Create a property of Material type.

Syntax:

NewMaterial('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptMaterialProperty

NewButt

Create a property of Edge material type.

Syntax:

NewButt('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptButtProperty

NewFurniture

Create a property of Accessories type.

Syntax:

NewFurniture('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptFurnitureProperty

NewColor

Create property of Color type.

Syntax:

NewColor('propName');

Input parameters:

propName – Property name.

Output parameters:

ScriptColorProperty

NewSeparator

Create a separator.

Syntax:

NewSeparator();

Output parameters:

ScriptProperty

Validate

Check property and nested properties value.

Syntax:

Validate();

Data type:

Boolean

ScriptGroupProperty

Script properties group.

Properties

Image – path to the image file for the button.

MaxHeight – maximum height.

Scrollable – adding a scroll bar.

Properties

Image

Path to the image file for the button.

Syntax:

Image;

Data type:

String

MaxHeight

Image maximum height.

Syntax:

MaxHeight;

Data type:

Number



If property value is not set, height is calculated automatically. Image width is equal to the width of **Value** column of **Properties** panel, if you change the column width the height will be changed in a way that image proportions are saved. If property value is set, when zooming automatically, the image height will not exceed this value.

Scrollable

Adding a scroll bar.

Syntax:

Scrollable;

Data type:

boolean

Taken values

True – to add scrollbars to a component of "group" type;

False – do not add scrollbars;

ScriptStringProperty

Property of String type

Properties:

Value – property value.

Properties

Value

String value.

Syntax:

Value;

Data type:

String

ScriptBooleanProperty

Property of Boolean type

Properties:

Value – property value.

Properties.

Value

Property value.

Syntax:

Value;

Data type:

Boolean

ScriptNumberProperty

Property of Number type

Properties:

MaxValue – property value.

MinValue – property value.

Value – property value.

ValueStep – property value.

Properties

MaxValue

Property maximum value.

Syntax:

MaxValue;

Data type:

Number

Note. The value of a numeric field can be set using the slider. To make it visible, you must set the MaxValue greater than MinValue.

MinValue

Property minimum value.

Syntax:

MinValue;

Data type:

Number

Note. The value of a numeric field can be set using the slider. To make it visible, you must set the MinValue greater than MaxValue.

ValueStep

Note. The value of a numeric field can be set using the slider. ValueStep value defines the distance between the slider divisions.

Syntax:

ValueStep;

Data type:

Number

Value

Property value.

Syntax:

Value;

Data type:

Number

ScriptButtonProperty

Handler for button pushing.

Properties:

Value – display name.

OnClick – the handler function of button pushing.

Properties

Value

Display parameter name

Syntax:

Value;

Data type:

String

OnClick

The handler function of button pushing.

Syntax:

OnClick;

Data type:

Function

ScriptMaterialProperty

Property of Material type

Properties:

Width – Sheet width.

Methods:

SetActive – Set the active material. All subsequent elements will be constructed from this material.

Properties

Width

Sheet width.

Syntax:

Width;

Data type:

Number

Methods

SetActive

Set the active material. All subsequent elements will be constructed from this material.

Syntax:

SetActive();

ScriptButtProperty

Property of Edgeband material type

Methods:

SetActive – Set the active edgeband material. All subsequent elements will be constructed of this material.

Properties:

Thickness – Edge thickness.

Width – Band width.

Properties

Thickness

Edge thickness.

Syntax:

Thickness;

Data type:

Number

Width

Band width.

Syntax:

Width;

Data type:

Number

Methods

SetActive

Set the active edge material. All subsequent elements will be constructed from this material.

Syntax:

SetActive();

ScriptFurnitureProperty

Property of Accessories type

Properties:

Value – accessories.

Properties:

Value

Accessories.

Syntax:

ScriptFurnitureProperty;

Data type:

InFurniture

ScriptColorProperty

Property of Color type

Properties:

Value – color index.

Properties

Value

Syntax:

Value;

Data type:

Number

ScriptMenu

A pop-up or drop-down menu item

Properties:

Name – Property name

OnChange – Handler for properties and nested properties changing

Count – Number of nested properties

Items – List of nested properties

Store – Checkbox whether the property is saved to a file

Methods:

Clear – Clear nested properties

Save – Save data entered by the user in an xml file

Load – Load field values from an xml file

NewGroup – Create a nested property group

NewString – Create property of String type

NewBool – Create Boolean type property

NewNumber – Create a numeric property

NewButton – Create button

Properties

Name

Property name

Syntax:

Name;

Data type:

String

OnChange

Handler for properties and nested properties changing

Syntax:

OnChange;

Data type:

Function

Count

Number of nested properties

Syntax:

Count;

Data type:

Number

Items

Number of nested properties

Syntax:

Items;

Data type:

Array.ScriptProperty

Store

Whether the property is saved to a file

Syntax:

Store;

Data type:

Boolean

Methods

Clear

Clear nested properties.

Syntax:

Clear();

Save

Save the data entered by the user in xml file.

Syntax:

Save('<file name>.xml');

Input parameters:

<file name>.xml



- Parameter <file name> can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.
- A set of values saved in a file can be loaded using method Load.

Load

Load properties values saved by method Save from xml file.

Syntax:

Load('<file name>.xml');

Input parameters:

<file name>.xml



Parameter <file name> can include a relative path from the folder where the script is saved, or an absolute path. Instead of using a single \ character, you must use a double (\\) character as separators for path elements.

NewGroup

Create a nested property group

Syntax:

NewGroup();

Input parameters:

Group name

Data type:

ScriptGroupProperty

NewString

Create a property of string type

Syntax:

NewString();

Data type:

ScriptGroupProperty

NewBool

Create property of Boolean type

Syntax:

NewBool();

Data type:

ScriptBooleanProperty

NewNumber

Create property of number type

Syntax:

NewNumber();

Data type:

ScriptNumberProperty

NewButton

Create button

Syntax:

NewButton();

Data type:

ScriptButtonProperty

ScriptForm

A custom screen form that allows you to set object parameters.

Properties:

Properties – A set of editable properties

Caption – Form caption

Width – Form width

Height – Form height

MinWidth – Form minimum width

MinHeight – Form minimum height

Visible – Form visibility

Left – Left form edge position

Top – Top form edge position

OKButton – Show **OK** button on form

OKButtonCaption – **OK** button text

CancelButton – Show **Cancel** button on form

CancelButtonCaption – **Cancel** button text

OnClose – Form closing handler

OnShow – Form opening handler

Resizable – The ability to change form size

Dockable – Ability to join to non-modal form

OnOkButtonClick – Handler for **OK** button pushing

OnCancelButtonClick – Handler for **Cancel** button pushing

Methods:

Close – Close form

Show – Show form

ShowModal – Show a modal form

Properties

Properties

Set of editable properties

Syntax:

Properties;

Data type:

ScriptProperty

Caption

Form caption

Syntax:

Caption;

Data type:

String

Width

Form width

Syntax:

Width;

Data type:

Number

Height

Form height

Syntax:

Height;

Data type:

Number

MinWidth

Minimum form width

Syntax:

MinWidth;

Data type:

Number

MinHeight

Minimum form width

Syntax:

MinHeight;

Data type:

Number

Visible

Form visibility

Syntax:

Visible;

Data type:

Boolean

Left

Form left edge position

Syntax:

Left;

Data type:

Number

Top

Top form edge position

Syntax:

Top;

Data type:

Number

OKButton

Show **OK** button on form

Syntax:

OKButton;

Data type:

Boolean

OKButtonCaption

OK button text

Syntax:

OKButtonCaption;

Data type:

String

CancelButton

Show **Cancel** button on form

Syntax:

CancelButton;

Data type:

Boolean

CancelButtonCaption

Cancel button text

Syntax:

CancelButtonCaption;

Data type:

String

OnClose

Form closing handler

Syntax:

OnClose;

Data type:

Function

OnShow

Form opening handler

Syntax:

OnShow;

Data type:

Function

Resizable

The ability to change the form size

Syntax:

Resizable;

Data type:

Boolean

Dockable

Ability to join non-modal form

Syntax:

Dockable;

Data type:

Boolean

OnOkButtonClick

Handler for **OK** button pushing

Syntax:

OnOkButtonClick;

Data type:

Function

OnCancelButtonClick

Handler for **Cancel** button pushing

Syntax:

OnCancelButtonClick;

Data type:

Function

Methods.

Show

Show form

Syntax:

ScriptForm.Show(WindowPosition);

Input parameters:

WindowPosition

ShowModal

Show modal form

Syntax:

ScriptForm.ShowModal();

Returned value:

Boolean

Close

Close form

Syntax:

ScriptForm.Close();

FurnMaterial

Material.

Properties

Name – Material name.

Thickness – Material thickness.

Width – Material width.

Methods

Make – Create material by name and thickness or width.

Properties

Name

Material name

Syntax

Name();

Data type

String

Thickness

Material thickness.

Syntax

Thickness();

Data type

Number

Width

Material thickness.

Syntax

Width();

Data type

Number

Methods.

Make

Create material by name and thickness or width.

Syntax:

FurnMaterial.Make(name, thickness,[width]);

Input parameters:

name – material name,

thickness – material thickness,

width (optional parameter) – material thickness.

Vector

Point in 3D space.

Properties:

x: – X point coordinate,

y: – Y point coordinate,

z: – Z point coordinate,

Properties:

x

X point coordinate.

Syntax:

x;

Data type:

Number

y

Y point coordinate.

Syntax:

y;

Data type:

Number

z

Z point coordinate.

Syntax:

z;

Data type:

Number

Point

Point on the plane.

Properties:

x – X point coordinate.

y – Y point coordinate.

Properties:

x

X point coordinate.

Syntax:

x;

Data type:

Number

y

Y point coordinate on the plane.

Syntax:

y;

Data type:

Number

Edge3

Edge.

Properties:

First – Edge start in LCS.

Last – Edge end in LCS.

GFirst – Edge start.

GLast – Edge end.

Properties

First

Edge start in LCS.

Data type:

Vector

Example:

Last

Edge end in LCS.

Data type:

Vector

Example:

GFirst

Edge start in the global coordinate system.

Data type:

Vector

Example:

GLast

Edge end in the global coordinate system.

Data type:

Vector

Example:

Example:

```
Edge = GetEdge("Specify the edge", AxisX);
alert(Edge.First.x);
alert(Edge.Last.x);
alert(Edge.GFirst.x);
```

```
alert(Edge.GLast.x);
```

Object3

3D object.

Is parent for Panel, Extrusion, Trajectory, List3D.

Properties:

Name – Name.

ArtPos – Position.

Owner – Object parent.

Visible – Object visibility.

Selected – Whether the object is selected.

List – Whether the object is structural.

AsList – Bring the object to a structural state.

AsPanel – Bring the object to the panel type.

Position – Object position.

PositionX – Coordinate x.

PositionY – Coordinate y.

PositionZ – Coordinate z.

SetDefaultTransform – Set the zero position and orientation of the object.

GSize – Model size (distance between GMax and GMin).

GMin – The vertex of the overall parallelepiped that is the least distant from the beginning of the parent coordinate system.

GMax – The vertex of the overall parallelepiped that is farthest from the beginning of the parent coordinate system.

GabMin – The vertex of the overall parallelepiped that is the least distant from the beginning of the global coordinate system.

GabMax – The vertex of the overall parallelepiped that is farthest from the beginning of the global coordinate system.

UserPropCount – Number of user properties.

UserProperty – Property values by its name or index.

UserPropertyName – Property name.

Methods:

SetDefaultTransform – Set zero position and object orientation.

Translate – Move the object.

Rotate – Rotate around the specified axis.

TranslateGCS – Move the object in the global coordinate system.

RotateGCS – Rotate the object in the global coordinate system.

RotateX – Rotate the object around the X axis.

RotateY – Rotate the object around the Y axis.

RotateZ – Rotate the object around the Z axis.

Orient – Expand the object relative to two axes.

OrientGCS – Expand the object relative to two axes in the global coordinate system.

Reflect – Reflect an object relative to a plane.

ToObject – Convert point coordinates from the global coordinate system to object LCS.

ToGlobal – Convert point coordinates from object CS to the global CS.

NToObject – Convert a noraml to object LCS.

NToGlobal – Convert a noraml from object LCS.

Build – Rebuild an object after its properties changing.

UserPropCount – Number of user properties.

UserProperty – Property values by its name and index.

UserPropertyName – Property name.

IsOwner – Determine whether the obj object is the parent of the object.

Properties:

Name

Name.

Syntax:

Name;

Data type:

String

ArtPos

Position.

Syntax:

ArtPos;

Data type:

String

Owner

Object parent.

Syntax:

Owner;

Data type:

List3D

Visible

Object visibility

Syntax:

Visible;

Data type:

Boolean

Selected

Whether the object is selected.

Syntax:

Selected;

Data type:

Boolean

List

Whether the object is structural.

Syntax:

List;

Data type:

Boolean

AsList

Bring the object to a structural state.

Syntax:

AsList;

Data type:

List3D

AsPanel

Bring an object to the panel type.

Syntax:

AsPanel;

Data type:

Panel

Position

Object position.

Syntax:

Position;

Data type:

Vector

PositionX

X coordinate.

Syntax:

PositionX;

Data type:

Number

PositionY

Y coordinate.

Syntax:

PositionY;

Data type:

Number

PositionZ

Z coordinate.

Syntax:

PositionZ;

Data type:

Number

GMax

The vertex of the overall parallelepiped that is the farthest from the origin of the parent object's coordinate system.

Syntax:

GMax;

Data type:

Vector

Example:

GMin

The vertex of the overall parallelepiped that is the least distant from the origin of the parent object's coordinate system.

Syntax:

GMin;

Data type:

Vector

Example:

GSize

The length of the object overall parallelepiped diagonal, the distance from GMax to Gmin.

Syntax:

GSize;

Data type:

Vector

Example:

GabMin

The vertex of the overall parallelepiped that is the least distant from the beginning of the global coordinate system.

Syntax:

GabMin;

Data type:

Vector

Example:

GabMax

The vertex of the overall parallelepiped that is farthest from the beginning of the global coordinate system.

Syntax:

GabMax;

Data type:

Vector

Example:

UserPropCount

Number of user properties.

Syntax:

UserPropCount;

Data type:

Number

UserProperty

The property value by its name or index.

Syntax:

UserProperty;

Data type:

Array.<Object>

UserPropertyName

Property name

Syntax:

UserPropertyName;

Data type:

Array.<String>

Example:

```
p = AddFrontPanel(50, 50, 100, 100, 50);
alert(p.GMax.x+' '+p.GMax.y+' '+p.GMax.z);
alert(p.GMin.x+' '+p.GMin.y+' '+p.GMin.z);
alert(p.GSize.x+' '+p.GSize.y+' '+p.GSize.z);
alert(p.GabMax.x+' '+p.GabMax.y+' '+p.GabMax.z);
alert(p.GabMin.x+' '+p.GabMin.y+' '+p.GabMin.z);
alert(p.GSize.x+' '+p.GSize.y+' '+p.GSize.z);
```

Methods

SetDefaultTransform

Set the zero position and object orientation.

Syntax:

SetDefaultTransform();

Translate

Move the object.

Syntax:

Translate(dir);

Input parameters:

dir – Shift direction. Set as Vector, or as a set of coordicates X, Y, Z (Number).

Rotate

Rotate around the specified axis in coordinate system of the parent object.

Syntax:

Rotate(axis,angle);

Input parameters:

axis – rotation axis, e.g., AxisX or direction from CS origin to the point (Vector)

angle – rotation angle, degrees.

Example:

```
p = AddPanel(100, 30);
p.Rotate(NewVector(50, 50, 50), 45);
```

TranslateGCS

Move the object in the global coordinate system.

Syntax:

TranslateGCS(dir);

Input parameters:

dir – Move direction. It is set as a Vector, or as a set of coordinates X, Y, Z (Number).

RotateGCS

Rotate around the specified axis in the global coordinate system.

Syntax:

RotateGCS(axis,angle);

Input parameters:

axis – rotation axis, e.g., AxisX or direction from the global CS origin to the point (Vector).

angle – rotation angle, degrees.

Example:

```
p = AddPanel(100, 30);
p.RotateGCS(NewVector(50, 50, 50), 45);
```

RotateX

Rotate around X axis.

Syntax:

RotateX(angle);

Input parameters:

angle – rotation angle, degrees.

Example:

```
p = AddPanel(100, 30);
p.RotateX(45);
```

RotateY

Rotate around Y axis.

Syntax:

```
RotateY(angle);
```

Input parameters:

angle – rotation angle, degrees.

Example:

```
p = AddPanel(100, 30);
p.RotateY(45);
```

RotateZ

Rotate around Z axis.

Syntax:

```
RotateZ(angle);
```

Input parameters:

angle – rotation angle, degrees.

Example:

```
p = AddPanel(100, 30);
p.RotateZ(45);
```

Orient

Rotate the object relative to two axes.

Syntax:

```
Orient(axis1,axis2);
```

Input parameters:

axis1 – direction along which the Z axis of the object's local coordinate system will be directed.

axis2 – direction along which the Y axis of the object's local coordinate system will be directed.

CS coordinate axes of parent object or direction from CS origin to the point (Vector) can be selected as axis1 and axis2. axis1 and axis2 can not be parallel to each other. First, the Z axis of the object is set, then the Y axis position is determined relative to it.

Example:

```
e = AddPanel(100, 50);
Ax2 = NewVector(1, 20, 40);
e.Orient(AxisY, Ax2);
```

OrientGCS

Expand the object relative to the two axes in the global CS.

Syntax:

OrientGCS(axis1,axis2);

Input parameters:

axis1 – direction along which the Z axis of the object's local coordinate system will be directed.

axis2 – direction along which the Y axis of the object's local coordinate system will be directed.

The global CS coordinate axes, e.g. AxisX or direction from the global CS origin to the point (Vector) can be selected as axis1 and axis2. axis1 and axis2 can not be parallel to each other. First, the Z axis of the object is set, then the Y axis position is determined relative to it.

Example:

```
p = AddFrontPanel(0, 0, 50, 40, 0);
y = AddFrontPanel(0, 0, 50, 40, 0);
Ax1 = NewVector(30, 60, 30);
Ax2 = NewVector(45, 45, 45);
y.Orient(Ax1, Ax2);
```

Reflect

Reflect an object relative to a plane.

Syntax:

Reflect(pos,normal);

Input data:

pos – the point through which the plane passes, Vector.

normal – the point through which the perpendicular to the plane passes, Vector.

Example:

```
y = AddFrontPanel(0, 0, 50, 40, 0);
Ax1 = NewVector(100, 60, 30);
Ax2 = NewVector(45, 45, 45);
y.Reflect(Ax1, Ax2);
```

ToObject

Convert the point coordinates from the global CS to the object LCS.

Syntax:

```
ToObject(pos);
```

Input data:

pos – point, Vector.



It is used for coordinates conversion. E.g. the CS origin point of the panel contour is set using GetPoint method. Its coordinates are obtained in the global CS. The contour is built on the panel face. Its points coordinates – in the panel LCS. To start contour construction from the specified point, you need to convert its coordinates to the panel LCS.

Example:

```
Pos = Model.Selected.ToObject(GetPoint("Specify the point"));
alert(Pos.x + ' ' + Pos.y + ' ' + Pos.z);
```

ToGlobal

Convert point coordinates from the object CS to the global CS.

Syntax:

```
ToGlobal(pos);
```

Input data:

pos – point, Vector.

Example:

```
Pos = Model.Selected.ToGlobal(NewVector(0, 0, 0));
alert(Pos.x + ' ' + Pos.y + ' ' + Pos.z);
```

NToObject

Convert normal from the global CS to the object LCS.

Syntax:

```
NToObject(dir);
```

Input data:

dir – point, Vector.

NToGlobal

Convert a normal from an object CS to the global CS.

Syntax:

NToGlobal(dir);

Input data:

dir – point, Vector.

Build

Rebuild an object after its properties changing.

Syntax:

Build;

IsOwner

Determine whether the obj object is the parent of the object

Syntax:

IsOwner;

Input parameters:

obj – the object prospective parent, Object3.

Data type:

boolean

List3D

3D objects list.

It is heir Object3.

Is the parent object for Block, Assembly, Model.

Properties:

Count – Number of list items.

IsElastic – Whether the object is elastic.

Objects – Objects list.

FurnType – Block type.

SalonType – Type for Salon.

Methods:

Load() – Add model to the list from file.

ElasticResize – Stretch an object to the required size.

Properties:

Count

Number of list items.

Syntax:

Count;

Output parameters:

Number of list items.

Example:

```
Block = AddBlock("Block");
Block.Load('Files\\IvoryW.fr3d');
Block.Load('Files\\IvoryB.fr3d');
n = Block.Count;
alert ('Number of block elements = '+n);
```

IsElastic

If the object is elastic.

Syntax:

IsElastic();

Output parameters:

Boolean

Objects

Objects list.

Syntax:

Objects;

Data type:

Array.Object3

Example:

```
block = AddBlock("Block");
block.Load('Files\\22.b3d');
block.Load('Files\\33.b3d');
block.Load('Files\\44.b3d');
k = block.Objects[1];
alert (k.Name);
```

FurnType

Block type

Syntax:

FurnType;

Data type:

String

SalonType

Type for Salon

Syntax:

SalonType;

Data type:

SalonAnimationType

Methods:

ElasticResize

Stretch the object to the required size.

Syntax:

ElasticResize(newSize);

Input parameters:

newSize – object new size. It is set as Vector.

Load

Add a model from a file to the list.

Syntax:

Load (‘file name’);

Input parameters:

Relative file name.

Output parameters:

List element.

Example:

```
block = AddBlock("Block");
block.Load('Files\\Cabinet.b3d');
block.Build();
```

Panel

Panel.

The object is the heir of Object3 objects.

Created by commands AddFrontPanel, AddHorizPanel, AddVertPanel, AddPanel.

Properties:

Contour – contour.

ContourHeight – contour height.

ContourWidth – contour width.

Thickness – thickness.

Butts – panel banded edges.

Cuts – slots.

Plastics – face banding.

MaterialName – material name.

MaterialWidth – maximum width of panel material.

Methods:

AddButt – add edge.

AddCut – add slot.

AddPlastic – add face banding.

IsButtVisible – whether the edge is covered by other panels.

Properties

Contour

Panel contour.

Syntax:

Contour;

Data type:

Contour2D

Example:

Butts

Panel edge.

Syntax:

Butts;

Data type:

PanelButts

Example:

Cuts

Panel slots.

Syntax:

Cuts;

Data type:

PanelCuts

Example:

Plastics

Panel face banding.

Syntax:

Plastics;

Data type:

PanelPlastics

Example:

ContourHeight

Panel contour height.

Syntax:

ContourHeight;

Data type:

Number

Example:

ContourWidth

Panel contour width.

Syntax:

ContourWidth;

Data type:

Number

Example:

Thickness

Panel thickness.

Syntax:

Thickness;

Data type:

Number

Example:

MaterialName

Panel material name.

Syntax:

MaterialName;

Data type:

String

Example:

MaterialWidth

Panel material width.

Syntax:

MaterialWidth;

Data type:

Number

Example:

Example:

```
panMat = NewMaterialInput("Material")
butMat = NewButtMaterialInput("Edge");
plastMat = NewMaterialInput("Plastic");
Btn = NewButtonInput("OK");
p1 = Vector(100, 100, 16);
p2 = Vector(300, 300, 16);
$input();
true;
function $input(id) {
   DeleteNewObjects();
   Pan = AddFrontPanel(100, 100, 200, 200, 20);
   Pan.AddButt(butMat, 1);
   Pan.AddButt(butMat, 3);
   Cut = Pan.AddCut('Slot');
   Cut.Contour.AddCircle(10, 10, 10);
   Cut.Trajectory.AddLine(p1, p2);
```

```
    Pan.AddPlastic(plastMat, true);
    Pan.ArtPos = 'Article';
    Pan.Build();
    height = Pan.ContourHeight;
    width = Pan.ContourWidth;
    MW = Pan.MaterialWidth;
    if (id == Btn.id)
    {
    alert ('Panel height= '+height+', Panel width= '+width);
    alert(Pan.MaterialName);
    alert(MW);
    alert(Pan.ArtPos);
     alert('Distance from the panel origin of coordinates to the vertex of the overall parallelepiped '+Pan.GMax.x+' '+Pan.GMax.y+' '+Pan.GMax.z);
     alert('Distance from the global origin of coordinates to the vertex of the overall parallelepiped '+Pan.GabMax.x+' '+Pan.GabMax.y+' '+Pan.GabMax.z);
    alert(Pan.GSize.x+' '+Pan.GSize.y+' '+Pan.GSize.z);
    return true;
    }
}
```

Methods

AddButt

Add edge to element contour.

Syntax:

AddButt (material, elem);

Input parameters:

material – edgeband material,

elem – contour element index, beginning with 0, counterclockwise from the origin of coordinates.

Output parameters:

PanelButt

Example:

AddCut

Add slot.

Syntax:

AddCut(name);

Input parameters:

name – slot name.

Output parameters:

PanelCut

Exampl:

AddPlastic

Add face banding.

Syntax:

AddPlastic(material, true);

Input parameters:

material – edgeband name,

true – panel side.

Output parameters:

PanelPlastic

Exampl:

IsButtVisible

Whether the edge is covered with other panels?

Syntax:

IsButtVisible(index, dist);

Input parameters:

index – edge index,

dist – closure condition.

Output parameters:

Boolean



An edge is considered unclosed if there are no panels at a distance greater than dist from it.

Extrusion

Extruded boss.

It is a body formed by moving the profile contour along a straight line. The profile contour must be closed.

The object is a heir of Object3.

Created by command AddExtrusion.

Properties:

Contour – profile contour,

Thickness – extruded boss thickness,

MaterialName – material name.

Properties

Contour

Extruded boss profile contour. The contour should be closed.

Syntax:

Contour;

Data type:

Contour2D

Example:

Thickness

Extruded boss thickness.

Syntax:

Thickness;

Data type:

Number

Example:

MaterialName

Extruded boss material name.

Syntax:

MaterialName;

Data type:

String

Example:

Example:

```
Extr = AddExtrusion('Boss');
Extr.Contour.AddCircle(100, 100, 50);
Extr.Thickness = 50;
Extr.Build();
ExtrName = Extr.MaterialName;
```

alert (ExtrName);

Trajectory

A trajectory is a curve along which the body formed by moving the profile contour along the trajectory. The trajectory contour can be closed or open. The profile contour must be closed.

The object is a heir of Object3.

Created by command AddTrajectory.

Properties:

Contour2D – profile contour,

Trajectory2D – trajectory contour,

MaterialName – material name.

Properties

Contour2D

The profile contour of kinematic body. Contour must be closed.

Syntax:

Contour2D;

Data type:

Contour2D

Example:

Trajectory2D

The trajectory contour of kinematic body.

Syntax:

Trajectory2D;

Data type:

Contour2D

Example:

MaterialName

Material name of kinematic body.

Syntax:

MaterialName;

Data type:

String

Example:

Example:

```
Traj = AddTrajectory('Trajectory');
p1 = NewVector(0, 0, 0);
p2 = NewVector(40, 50, 70);
p3 = NewVector(100, 150, 120);
Traj.Trajectory2D.AddArc3(p1,  p2,  p3);
Traj.Contour2D.AddRoundRect(30, 50, 50, 70, 8);
TrName = Traj.MaterialName;
alert (TrName);
```

Block

Block. It is a named group of model elements. Created by command AddBlock(). The block elements can be accessed by index. Numbering starts from 0.

The object is a heir of List3D.

Properties:

IsFastener – Checkbox for composite accessories.

Properties:

IsFastener

Checkbox for composite accessories

Syntax:

IsFastener;

Data type:

Boolean

Assembly

Assembly. It is a named group of model elements. Created by command AddAssembly(). The assembly elements can be accessed by index. Numbering starts from 0.

The object is a heir of List3D.

Properties:

AnimType – Animation type.

Properties

An mType

Assembly animation type.

Syntax:

AnimType;

Data type:

AnimationType

Example:

```
asm = AddAssembly("Assembly");
p = AddFrontPanel(0, 0, 40, 40, 0);
p.Owner = asm;
asm.AnimType = AnimationType.DoorRight;
```

PanelButts

Edges.

Created by Butts command.

Properties:

Count – edges count.

Methods:

Butts[] – get the panel edge by index.

Add – add edge.

Properties

Count

Edges count.

Syntax:

Count;

Data type:

Number

Example:

```
 Pan = AddFrontPanel(0, 0, 400, 400, 0);
    Pan.AddButt(butMat, 1);
    Pan.AddButt(butMat, 3);
    Pan.AddPlastic(plastMat, true);
    Pan.Build();
    height = Pan.ContourHeight;
    width = Pan.ContourWidth;
    butts = Pan.Butts;
```

Methods

Butts[]

Get the panel edge by index.

Syntax:

Butts[i];

Input parameters:

i – panel edge index starting from 0.

Data type:

PanelButt

Add

Add panel edge.

Syntax:

Add;

Output parameters:

PanelButt



We recommend to use this command only if you add any specific edge. As a result of command execution, an "empty" edge is created, its parameters must be set using additional commands. In most cases it is advisable to use AddButt command.

PanelButt

Panel edge.

Properties:

ElemIndex – Edge index.

Sign – Edge designation.

Material – Edge material.

Thickness – Edge thickness.

Width – Edge width.

ClipPanel – Trim the panel to the edge thickness.

Overhung – Overhang, how much the edge band should be longer than the end of the part.

Allowance – Allowance, how much to mill the edge before edging.

CutIndex – Cut number (to edge banding on multiple edges with a single cut).

Profile – Edge profile.

Properties

ElemIndex

Edge index.

Syntax:

ElemIndex;

Data type:

Number

Sign

Edge designation.

Syntax:

Sign;

Data type:

String

Material

Edge material.

Syntax:

Material;

Data type:

String

Thickness

Edge thickness.

Syntax:

Thickness;

Data type:

Number

Width

Edge width.

Syntax:

Width;

Data type:

Number

ClipPanel

Trim the panel to the edge thickness.

Syntax:

ClipPanel;

Data type:

boolean

Property value:

True – undercut

False – do not undercut

Overhung

Overhang, how much the edgeband should be longer than panel edge.

Syntax:

Overhung;

Data type:

Number

Allowance

Allowance - how much to mill the end before edging.

Syntax:

Allowance;

Data type:

Number

CutIndex

Cut number (for edgebanding pasting on several edges with a single cut).

Syntax:

CutIndex;

Data type:

Number

Profile

Edge profile.

Syntax:

Profile;

Data type:

Contour2D

PanelPlastics

Created by the command Plastics.

Properties:

Count – number of panel bandings.

Methods:

Plastics[] – to get the face banding by index.

Add – add banding.

Properties

Count

Face banding count.

Syntax:

Count;

Data type:

Number

Example:

```
Pan = AddFrontPanel(0, 0, 400, 400, 0);
  Pan.AddButt(butMat, 1);
  Pan.AddButt(butMat, 3);
  Pan.AddPlastic(plastMat, true);
  Pan.Build();
  height = Pan.ContourHeight;
  width = Pan.ContourWidth;
  butts = Pan.Butts;
```

Methods

Plastics[]

Get edge banding by index.

Syntax:

Plastics[i];

Input parameters:

i – banding index starting from 0.

Output parameters:

PanelPlastic

Add

Add banding.

Syntax:

Add();

Output parameters:

PanelPlastic



We recommend to use this command only if you add any specific face edge banding. As a result of command execution, an "empty" banding is created, its parameters must be set using additional commands. In most cases it is advisable to use AddPlastic command.

PanelPlastic

Panel plastic.

Properties:

Material – Plastic material.

Thickness – Plastic thickness.

TextureOrientation – Plastic texture orientation.

Properties:

Material

Panel plastic material.

Syntax:

Material;

Data type:

String

Thickness

Panel plastic thickness.

Syntax:

Thickness;

Data type:

Number

TextureOrientation

Panel plastic texture orientation.

Syntax:

TextureOrientation;

Data type:

TextureDir

PanelCuts

Panel slots.

Properties:

Count – slots count.

Methods:

Cuts[] – get the slot by index.

Add – add slot.

Properties

Count

Slot count.

Syntax:

Count;

Data type:

Number

Example:

```
Pan = AddFrontPanel(0, 0, 400, 400, 0);
  Pan.AddButt(butMat, 1);
  Pan.AddButt(butMat, 3);
  Pan.AddPlastic(plastMat, true);
  Pan.Build();
  height = Pan.ContourHeight;
  width = Pan.ContourWidth;
  butts = Pan.Butts;
```

Methods

Add

Add slot.

Syntax:

Add();

Output parameters:

PanelCut

Cuts

Get a slot by index.

Syntax:

Cuts[i];

Input parameters:

i – slot index starting from 0.

Output parameters:

PanelCut

PanelCut

Slot.

Properties:

Name – Slot name.

Sign – Slot designation.

Trajectory – Slot trajectory.

Contour – Slot profile.

Properties

Name

Slot name.

Syntax:

Name;

Data type:

String

Sign

Slot designation.

Syntax:

Sign;

Data type:

String

Trajectory

Slot trajectory.

Syntax:

Trajectory;

Data type:

Contour2D

Contour

Slot profile.

Syntax:

Contour;

Data type:

Contour2D

Contour2D

Contour.

Properties:

Count – Number of contour elements.

Width – Contour width.

Height – Contour height.

Min – Lower left corner of the enclosing rectangle.

Max – Upper right corner of the enclosing rectangle.

Methods:

Clear – Clear the contour.

Move – Move all contour elements.

Rotate – Rotate the contour around a point.

AddRectangle – Add a rectangle.

AddRoundRect – Add a rectangle with rounded edges.

AddLine – Add a line.

AddCircle – Add a circle.

AddArc – Add an arc along the start, end point, and center.

AddArc3 – Add an arc by 3 points.

AddEquidistantRecursive – Add contour equidistant (including nested contours).

Subtraction – Subtract a closed contour.

Addition – Fold with a closed contour.

RoundingEx – Create a fillet in intersection point of two contour segments.

FacetEx – Create a chamfer in intersection point of two contour segments.

Rounding – Create a fillet in intersection point of two contour elements.

Facet – Create a chamfer in intersection point of two contour segments.

Find – Find the nearest element by coordinates.

Fit – Fit the entire contour to the specified dimensions.

Elastic – Change the contour with the rubber band.

Symmetry – Create an object copy that is symmetrical to the line.

Load – Load contour from a fragment file.

OrderContours – Arrange contour elements in one direction.

AddEquidistant – Add contour equidistant.

IsPointInside – Whether the point is inside the contour?

IsInContour – Whether the contour is inside another one?

IsClosedContour – Whether the contour is closed?

IsContourRectangle – Whether the contour is rectangular?

Example:

```
Obj = AddPanel(300, 300);
Pan = Obj.AsPanel; //whether the selected object is a panel
if (Pan) {
  c = Pan.Contour;
  Undo.Changing(Obj); // specify that this object will be edited
  //p = Action.GetPoint('Specify the hole center');
  p = NewVector(40, 40, 40);
  p = Obj.ToObject(p); // convert the point to the coordinate system of the panel contour
  Hole = NewContour();
  Hole.AddRectangle(40, 40, 100, 110);
  //Hole.Rounding(59, 45, 30);
  Hole.Facet(80, 100, 10);
 // Hole.Facet(80, 100, 40);
  Hole.Rotate(40,40,30);
  c.Substraction(Hole);
      ro = NewContour();
      e1 = ro.AddLine(100, 160, 190, 190);
      e2 = ro.AddLine(160, 60, 190, 190);
      e3 = ro.AddLine(100, 160, 160, 60);
      ro.RoundingEx(e1, e2, 170, 170, 20);
        f = ro.Find(110, 140);
      ro.RoundingEx(e2, f, 150, 90, 26,5);
```

```
      ro.FacetEx(e1, e3, 20, 20);
    ro.Fit(100, 30, 170, 200);
    ro.Elastic(100, 30, 170, 60, 50, 70);
    ro.Symmetry(185, 30, 150, 200, false);
    c.Substraction(ro);
        lo = NewContour();
        lo.Load('contour.frw');
        lo.Move(55, 180);
        c.Substraction(lo);
  Obj.Build(); // rebuild the object after all changes
}
```

Properties

Count

Number of contour elements.

Syntax:

Count;

Data type:

Number

Example:

Objects

List of contour elements.

Syntax:

Objects;

Data type:

Array.Elem2D

Width

Contour width.

Syntax:

Width;

Data type:

Number

Example:

Height

Contour height.

Syntax:

Height;

Data type:

Number

Example:

Min

Lower left corner of the enclosing rectangle.

Syntax:

Min;

Returned value:

Point

Max

Upper right corner of the enclosing rectangle.

Syntax:

Max;

Returned value:

Point

Methods

Clear

Clear the contour.

Syntax:

Clear();

Example:

Move

Move all contour elements.

Syntax:

Move(dx,dy);

Input parameters:

dx, dy – move values along the corresponding axes.

Example:

Rotate

Rotate the contour around a point.

Syntax:

Rotate(x, y, angle);

Input parameters:

x, y – coordinates of rotation center.

angle – rotation angle

AddRectangle

Add a rectangle.

Syntax:

AddRectangle(x1, y1, x2, y2);

Input parameters:

x1, y1, x2, y2 – coordinates of start and end points of rectangle diagonal.

Example:

AddRoundRect

Add rectangle with rounded vertexes.

Syntax:

AddRoundRect(x1, y1, x2, y2, rad);

Input parameters:

x1, y1, x2, y2 – coordinates of start and end points of the rectangle diagonal.

rad – fillet radius.

Example:

AddLine

Add line.

Syntax:

AddLine(x1, y1, x2, y2);

Input parameters:

x1, y1, x2, y2 – coordinates of start and end points of the line.

Example:

AddCircle

Add a circle.

Syntax:

AddCircle(xc, yc, rad);

Input parameters:

xc, yc – coordinates of circle center.

rad – circle radius.

Example:

AddArc

Add an arc along the start, end points, and center.

Syntax:

AddArc(p1, p2, centre, dir);

Input parameters:

p1, p2 – start and end points (Vector).

centre – arc center (Vector).

dir –arc direction:

- true – from p1 to p2;
- false – from p2 to p1;

Example:

AddArc3

Add a three-point arc.

Syntax:

AddArc3(p1, p2, p3);

Input parameters:

p1, p2, p3 – arc points (Vector).

Example:

AddEquidistantRecursive

Add contour equidistant (including nested contours). The last 2 parameters are responsible for direction and rounding.

Syntax:

AddEquidistantRecursive(contour, offset, Side, Rounding);

Input parameters:

p1, p2, p3 – arc points (Vector).

contour – contour, Contour2D

offset – shift, number

Side – shift direction, boolean

Rounding – vertexes rounding, boolean

Subtraction

Subtract a closed contour.

Syntax:

Subtraction(contour);

Input parameters:

contour – subtracted contour (Contour2D).

Example:

Addition

Add a enclosed contour.

Syntax:

Addition(contour);

Input parameters:

contour - added contour (Contour2D).

Example:

RoundingEx

Create a rounding at the intersection of two contour segments.

Syntax:

RoundingEx(elem1, elem2, x, y, radius);

Input parameters:

elem1, elem2 – the contour segments, between which the rounding is built,

x, y – point coordinates that determine the selection of the rounding arc in the contour CS.

radius – the fillet radius.



This method is reasonable if the contour consists of separate named segments. To round rectangle vertexes use method Rounding.

Example:

FacetEx

Build a chamfer at the intersection point of two contour segments.

Syntax:

FacetEx(elem1, elem2, l1, l2);

Input parameters:

elem1, elem2 – contour elements that have a chamfer between them,

l1, l2 – indents from the vertex for the corresponding elements.

Example:

Rounding

Create a rounding at the intersection of two contour elements.

Syntax:

Rounding(x, y, radius);

x, y – point coordinates that determine the selection of a rounding arc in the contour CS.

radius – fillet radius.



Method is reasonable for rectangle vertexes rounding. If the contour consists of separate named segments, use method RoundingEx.

Example:

Facet

Build a chamfer at the intersection of two contour elements.

Syntax:

Facet(x, y, l);

Input parameters:

x, y – point coordinates that determine the choice of a vertex for building a chamfer in the contour CS.

l – the chamfer length (the same for both elements).

Example:

Find

Find the element that is the closest to the point with the specified coordinates.

Syntax:

Find(x, y);

Input parameters:

x, y – point coordinates.

Example:

Fit

Fit the contour to the specified dimensions.

Syntax:

Fit(x1, y1, x2, y2);

Input parameters:

x1, y1, x2, y2 – coordinates of the diagonal ends of the overall rectangle

Example:

Elastic

Change the contour with a rubber thread.

Syntax:

Elastic(x1, y1, x2, y2, dx, dy);

Input parameters:

x1, y1, x2, y2 – coordinates of the diagonal ends of the overall rectangle that cover a contour part.

dx, dy – shift direction.

Example:

Symmetry

Create an object copy that is symmetrical to the line.

Syntax:

Symmetry(x1, y1, x2, y2, del);

Input parameters:

x1, y1, x2, y2 – coordinates of symmetry axis ends.

del – indicates whether the source object was deleted:

- true – do not delete;
- false – delete.

Example:

Load

Download a contour from the fragment file.

Syntax:

Load(“file”);

Input parameters:

file – fragment file name with frw extension.

Example:

OrderContours

Arrange contour elements in one direction.

Syntax:

OrderContours;

Returned value:

Boolean

AddEquidistant

Add contour equidistant.

Syntax:

AddEquidistant(contour,offset,false,false);

Input parameters:

contour – pointer to the source contour.

offset – shift.

dir – direction checkbox.

fillet – fillet checkbox.

Example:

```
X=NewContour();
X.Load(ttt.frw);
Y=NewContour();
Y.AddEquidistant(X,5, false,true);
```

IsPointInside

Check whether the point is inside the contour?

Syntax:

IsPointInside(x,y);

Input parameters:

x, y – point coordinates.

Returned value:

Boolean

IsInContour

Check whether the contour is inside another one.

Syntax:

IsInContour(contour);

Input parameters:

contour – contour, that is being checked for being inside it.

Returned value:

Boolean

IIsClosedContour

Check whether the contour is closed.

Syntax:

IsClosedContour;

Returned value:

Boolean

IsContourRectangle

Check whether the contour is rectangular.

Syntax:

IsContourRectangle;

Returned value:

Boolean

InControl

Control element.

Properties:

id – ID.

Enabled – object availability.

Visible – object visibility.

Hint – text of the displayed hint.

OnChange – the handler function.

Properties

id

Identifier.

Syntax:

id;

Data type:

Number

Enabled

Object availability.

Syntax:

Enabled;

Data type:

Boolean

Visible

Object visibility.

Syntax:

Visible;

Data type:

Boolean

Hint

Text of the displayed hint.

Syntax:

Hint;

Data type:

String

OnChange

The handler function.

Syntax:

OnChange;

Data type:

Function

InButton

Button.

Properties:

Caption – The text on the button.

Combo – Separate arrow for the submenu.

Methods:

NewSubMenu – Create a submenu.

Properties:

Caption

Text on button.

Syntax:

Caption;

Data type:

String

Combo

Separate arrow for the submenu.

Syntax:

Combo;

Data type:

Boolean

Methods:

NewSubMenu

Create a submenu.

Syntax:

NewSubMenu(MenuItemName);

Input data:

MenuItemName – Name of the menu item (String).

Data type:

InButton

InFloat

Number with floating point.

Properties:

Value – Value.

ReadOnly – Only for reading.

Fixed – The value is set in the input field.

Properties

Value

Value.

Syntax:

Value;

Data type:

Number

ReadOnly

Only for reading.

Syntax:

ReadOnly;

Data type:

Boolean

Fixed

The value is set in the input field.

Syntax:

Fixed;

Data type:

Boolean

InNumber

Properties:

Value – Value.

ReadOnly – Only for reading.

Fixed – The value is set in the input field.

Properties

Value

Value.

Syntax:

Value;

Data type:

Number

ReadOnly

Only for reading.

Syntax:

ReadOnly;

Data type:

Boolean

Fixed

The value is set in the input field.

Syntax:

Fixed;

Data type:

Boolean

InMaterial

Material

Properties:

Name – material name.

Thickness – material thickness.

Width – material width.

Methods:

SetActive – set active.

Apply – apply material to the specified object.

Properties

Name

Material name.

Syntax:

Name;

Data type:

String

Thickness

Material thickness.

Syntax:

Thickness;

Data type:

Number

Width

Material width.

Syntax:

Width;

Data type:

Number

Methods

SetActive

Set active material. All subsequent elements will be constructed from this material.

Syntax:

SetActive();

Apply

Apply material to the specified object.

Syntax:

Apply(obj);

Input data:

obj – object of material application (Object3).

Data type:

Object3

InButtMaterial

Edge material.

Properties:

Name – material name.

Sign – material designation.

Thickness – material thickness.

Width – material width.

Overhung – overhang.

Allowance – allowance.

ClipPanel – panel undercutting.

Properties

Name

Material name.

Syntax:

Name;

Data type:

String

Sign

Material designation.

Syntax:

Sign;

Data type:

String

Thickness

Material thickness.

Syntax:

Thickness;

Data type:

Number

Width

Material width.

Syntax:

Width;

Data type:

Number

Overhung

Overhang.

Syntax:

Overhung;

Data type:

Number

Allowance

Allowance.

Syntax:

Allowance;

Data type:

Number

ClipPanel

Panel undercutting.

Syntax:

ClipPanel;

Data type:

Boolean

InFurniture

Accessories.

Properties:

DatumModeFilter – Filter by accessories/fragment mounting type.

DatumMode – Type of selected accessories/fragment mounting.

Properties

DatumModeFilter

Filter by accessories/fragment mounting type.

Syntax:

DatumModeFilter;

Data type:

DatumMode

DatumMode

Type of selected accessories/fragment mounting.

Syntax:

DatumMode;

Data type:

DatumMode

Methods:

Mount – Install fasteners between two panels.

Mount1 – Install fasteners on the panel plane.

Methods

Mount

Install fasteners between two panels.

Syntax:

Mount(panel1,panel2,x,y,z);

Input data:

panel1,panel2 – connected panels, Object3.

x, y, z – coordinates of accessories installation point; can be set as Vector.

Mount1

Install fasteners on the panel plane.

Syntax:

Mount1(panel, x, y, z, angle);

Input parameters:

panel – panel where accessories are installed, Object3.

x, y, z – coordinates of accessories installation point, number, can be set as Vector.

angle – accessories rotation angle around the horizontal axis in degrees, number.

MountScheme

Install accessories map at the panel joint.

Syntax:

MountScheme (panel1, panel2, FurnPos, basisPoint);

Input parameters:

panel1, panel2 – panels that have a joint, Panel.

FurnPos – Accessories position when installing the fasteners, FurniturePosition.

basisPoint – the base point for accessories installing; can be set as Vector.

MountBox

Install section.

Syntax:

MountBox (Position,Size,AxisZ,AxisY

Input parameters:

Position – Section position, Vector.

Size – Section size, Vector.

AxisZ – Direction of Z axis of the section, Vector.

AxisY – Direction of Y axis of the section, Vector.

ModelInspector

Model analysis

Properties:

ErrorList – A list of errors.

Options – Analysis options.

Methods:

Run – Check the model.

Properties

ErrorList

A list of errors.

Syntax:

ErrorList;

Data type:

Array <InspectorError>

Options

Options of analysis.

Syntax:

Options;

Data type:

InspectorOptions

Methods

Run

Check the model.

Syntax:

Run(Model);

Input data:

Model – checked model (List3D).

Returned value:

Absent

InspectorOptions

Parameters of model analysis

Properties:

ObjIntersectionAnalyze – Intersection of objects check.

FastIntersectionAnalyze – Intersection of accessories check.

FastIncorrectAnalyze – Correctness of accessories check.

PanelNotFixedAnalyze – Panels fixing check.

PanelTooLargeAnalyze – Panel size on the board check.

PlasticTooLargeAnalyze – Plastic size on the panel check.

MatNotExistsAnalyze – Available material check.

MatOutOfStockAnalyze – Material in stock check.

Properties

ObjIntersectionAnalyze

Intersection of objects check.

Syntax:

ObjIntersectionAnalyze;

Data type:

boolean

FastIntersectionAnalyze

Intersection of accessories check.

Syntax:

FastIntersectionAnalyze;

Data type:

boolean

FastIncorrectAnalyze

Correctness of accessories check.

Syntax:

FastIncorrectAnalyze;

Data type:

boolean

PanelNotFixedAnalyze

Panels fixing check.

Syntax:

PanelNotFixedAnalyze;

Data type:

boolean

PanelTooLargeAnalyze

Panel size on the board check.

Syntax:

PanelTooLargeAnalyze;

Data type:

boolean

PlasticTooLargeAnalyze

Plastic size on the panel check.

Syntax:

PlasticTooLargeAnalyze;

Data type:

boolean

MatNotExistsAnalyze

Available material check.

Syntax:

MatNotExistsAnalyze;

Data type:

boolean

MatOutOfStockAnalyze

Material in stock check.

Syntax:

MatOutOfStockAnalyze;

Data type:

boolean

InspectorError

Parameters of found error

Properties:

ErrorType – Error type.

ErrorObjectsCount – Number of objects related to the error.

ErrorObjects – List of objects related to the error.

ErrorMessage – Error message.

ObjectsNames – Object names in the error.

Properties

ErrorType

Error type.

Syntax:

ErrorType;

Data type:

ErrorType

ErrorObjectsCount

Number of objects related to the error.

Syntax:

ErrorObjectsCount;

Data type:

number

ErrorObjects

List of objects related to the error.

Syntax:

ErrorObjects;

Data type:

Array<Object3>

ErrorMessage

Error message.

Syntax:

ErrorMessage;

Data type:

string

ObjectsNames

Object names in the error.

Syntax:

ObjectsNames;

Data type:

string

ImportExportSVG

Import-export в SVG

Properties:

CurveQuality – Approximation quality of curves into lines [0-100].

GroupElems – Combine elements into polygons, polylines, and curves.

Methods:

Save – Save contour.

Load – Load contour.

Properties

CurveQuality

Approximation quality of curves into lines [0-100].

Syntax:

CurveQuality;

Data type:

number

GroupElems

Combine elements into polygons, polylines, and curves.

Syntax:

GroupElems;

Data type:

boolean

Methods

Save

Save contour.

Syntax:

Save(filename, contour);

Input data:

filename – file name, string

contour – contour which data is saved to a file, Contour2D.

Returned value:

boolean

Load

Load contour.

Syntax:

Load(filename, contour);

Input data:

filename – file name, string

contour – where data from the file is read, Contour2D.

Returned value:

boolean

ImportExport

Import-export

Properties:

SVG – Import-export in SVG.

Properties

SVG

Import-export in SVG.

Syntax:

SVG;

Data type:

ImportExportSVG

InfFurniture

Information about accessories

Properties:

DatumModeFilter – Filter by accessories/fragment mounting type.

DatumMode – Type of selected accessories/fragment mounting.

Methods:

Mount – Install fasteners between two panels.

Mount1 – Install fasteners on the panel plane.

MountScheme – Install the mounting map of panel joint.

MountBox – Install a section.

Make – Create accessory object.

EncodeToString – Encrypt accessories settings in the line.

DecodeFromString – Restore accessories settings from the line.

Choose – Open accessories selection window.

Properties

DatumModeFilter

Filter by accessories/fragment mounting type.

Syntax:

DatumModeFilter;

Data type:

DatumMode

DatumMode

Type of selected accessories/fragment mounting.

Syntax:

DatumMode;

Data type:

DatumMode

Methods

Mount

Install fasteners between two panels.

Syntax:

Mount(panel1, panel2, x, y, z);

Input data:

panel1, panel2 – panels, Panel.

x, y, z – coordinates, number

Returned value:

Object3

Mount1

Install fasteners on the panel plane.

Syntax:

Mount1(panel, x, y, z, angle);

Input data:

panel1 – panel, Panel.

x, y, z – coordinates, number

angle – angle in degrees, number

Returned value:

Object3

MountScheme

Install the mounting map of the panel joint.

Syntax:

MountScheme(panel1, panel2, FurniturePosition, BasisPoint);

Input data:

panel1 – panel, Panel.

FurniturePosition – accessories position, FurniturePosition

BasisPoint – base point, Vector

Returned value:

Object3

MountBox

Mount section.

Syntax:

MountBox(Position, Size, axisZ, axisY);

Input data:

Position – Section position, Vector

Size – Section size, Vector

axisZ - Direction of Z axis of the section, Vector

axisY - Direction of Y axis of the section, Vector

Returned value:

Object3

Make

Create accessories object.

Syntax:

Make(Thickness1, Thickness2);

Input data:

Thickness1, Thickness2 – panel thickness, number

Returned value:

Object3

EncodeToString

Create accessories object.

Syntax:

EncodeToString();

Returned value:

string

DecodeFromString

Create accessories object.

Syntax:

DecodeFromString(str);

Input data:

str – a string of accessories parameters saved by EncodeToString method, string

Returned value:

no

Choose

Open accessories selection window.

Syntax:

Choose();

Returned value:

boolean (true – the selection is made, false – the selection is canceled)

RootProperties

Root properties

Methods:

NewFurnitureValue – Create an object with information about accessories.

Methods

NewFurnitureValue

Create an object with information about accessories.

Syntax:

NewFurnitureValue();

Returned value:

InfFurniture

Elem2D

2D element.

Properties:

ElType – a type of 2D element.

Methods:

IsLine – Whether the element is a line.

AsLine – Get an element as a line.

IsCircle – Whether the element is a circle.

AsCircle – Get an element as a circle.

IsEllipse – Whether the element is an ellipse.

AsEllipse – Get an element as an ellipse.

IsArc – Whether the element is an arc.

AsArc – Get an element as an arc.

IsList – Whether the element is a list of elements.

AsList – Get an element as a list of elements.

ObjLength – Element length.

Properties

ElType

2D element type.

Syntax:

ElType;

Data type:

ElementType

Methods

IsLine

Whether the element is a line.

Syntax:

IsLine();

Returned value:

boolean

AsLine

Get an element as a line.

Syntax:

AsLine();

Returned value:

Line2D

IsCircle

Whether the element is a circle.

Syntax:

IsCircle();

Returned value:

boolean

AsCircle

Get an element as a circle.

Syntax:

AsCircle();

Returned value:

Circle2D

IsEllipse

Whether the element is an ellipse.

Syntax:

IsEllipse();

Returned value:

boolean

AsEllipse

Get an element as an ellipse.

Syntax:

AsEllipse();

Returned value:

Ellipse2D

IsArc

Whether the element is an arc.

Syntax:

IsArc();

Returned value:

boolean

AsArc

Get an element as an arc.

Syntax:

AsArc();

Returned value:

Arc2D

IsList

Whether the element is a list of elements.

Syntax:

IsList();

Returned value:

boolean

AsList

Get an element as a list of elements.

Syntax:

AsList();

Returned value:

Contour2D

ObjLength

Element length.

Syntax:

ObjLength();

Returned value:

number

Line2D

2D line

Properties:

Pos1 – Begin.

Pos2 – End.

Properties

Pos1

Begin.

Syntax:

Pos1;

Data type:

Point

Pos2

End.

Syntax:

Pos2;

Data type:

Point

Arc2D

2D arc

Properties:

Pos1 – Begin.

Pos2 – End.

Center – Center.

ArcDir – Direction.

Properties

Pos1

Begin.

Syntax:

Pos1;

Data type:

Point

Pos2

End.

Syntax:

Pos2;

Data type:

Point

Center

Center.

Syntax:

Center;

Data type:

Point

ArcDir

Direction.

Syntax:

ArcDir;

Data type:

boolean

Taken values:

True – counterclockwise;

False – clockwise;

Circle2D

2D circle

Properties:

Center – Center.

CirRadius – Radius.

Dir – Direction.

Properties

Center

Center.

Syntax:

Center;

Data type:

Point

CirRadius

Radius.

Syntax:

CirRadius;

Data type:

number

Dir

Direction.

Syntax:

ArcDir;

Data type:

boolean

Taken values:

True – counterclockwise;

False – clockwise;

Ellipse2D

2D ellipse

Properties:

Center – Center.

MajorRadius – Large radius - has number type

MinorRadius – Small radius - has number type

MajorAxisAngle – Angle - has number type

Dir – Direction

Properties

Center

Center.

Syntax:

Center;

Data type:

Point

MajorRadius

Large radius.

Syntax:

MajorRadius;

Data type:

number

MinorRadius

Small radius.

Syntax:

MinorRadius;

Data type:

number

MajorAxisAngle

Angle.

Syntax:

MajorAxisAngle;

Data type:

number

Dir

Direction.

Syntax:

Dir;

Data type:

boolean

Taken values:

True – counterclockwise;

False – clockwise;

FurnArticle

Model article

Properties:

Name – Model name.

OrderCode – Order code.

OrderName – Order name.

DatumMode – Type of accessories/fragment installation.

Method:

NameWithCode – Get name and article.

Properties

Name

Model name.

Syntax:

Name;

Data type:

string

OrderCode

Order code.

Syntax:

OrderCode;

Data type:

string

OrderName

Order name.

Syntax:

OrderName;

Data type:

string

DatumMode

Type of accessories/fragment installation.

Syntax:

DatumMode;

Data type:

DatumMode

Methods

NameWithCode

Get name and article.

Syntax:

NameWithCode();

Returned value:

string

BAZIS-Woodworker

Global constants of BAZIS-Woodworker

p3dFront

Front view.

Data type:

Number

p3dLeft

Left-hand view.

Data type:

Number

p3dRight

Right-hand view.

Data type:

Number

p3dTop

Top view.

Data type:

Number

p3dBottom

Bottom view.

Data type:

Number

p3dIsometric

Axonometry.

Data type:

Number

BAZIS-Estimate

Global properties of BAZIS-Estimate

Panel

Access to panel parameters.

Syntax:

Panel;

Data type:

Panel

Holes

List of panel holes

Syntax:

Holes;

Data type:

PanelHoles

Holes: property of PanelHoles type - 'List of panel holes

Material

Current material

Syntax:

Material;

Data type:

String

Material: property of String type- "Current material"

Result

Result of parameter calculation

Syntax:

Result;

Data type:

Number

Result: property of Number type - 'Result of parameter calculation'

Objects

Script objects

Syntax:

Objects;

Data type:

EstimateObjectList

EstimateObject

Estimate object

Methods:

AsPanel – Get an object as a panel.

IsPanel – Check whether the object is a panel.

AsButt – Get an object as an edge.

IsButt – Check whether the object is an edge.

AsPlastic – Get an object as plastic.

IsPlastic – Check whether the object is plastic.

AsExtrusionBody – Get an object as a profile.

IsExtrusionBody – Check whether the object is a profile.

AsTrajectoryBody – Get an object as a body along the trajectory.

IsTrajectoryBody – Check whether the object is a body along the trajectory.

Methods

AsPanel

Get an object as a panel.

Syntax:

AsPanel();

Returned value:

Panel

IsPanel

Check whether the object is a panel.

Syntax:

IsLine();

Returned value:

Boolean

AsButt

Get an object as an edge.

Syntax:

AsButt();

Returned value:

PanelButt

IsButt

Check whether the object is an edge.

Syntax:

IsButt();

Returned value:

Boolean

AsPlastic

Get an object as plastic.

Syntax:

AsPlastic();

Returned value:

PanelPlastic

IsPlastic

Check whether the object is plastic.

Syntax:

IsPlastic();

Returned value:

Boolean

AsExtrusionBody

Get an object as a profile.

Syntax:

AsExtrusionBody();

Returned value:

Extrusion

IsExtrusionBody

Check whether the object is a profile.

Syntax:

IsExtrusionBody();

Returned value:

Boolean

AsTrajectoryBody

Get an object as a body along the trajectory.

Syntax:

AsTrajectoryBody();

Returned value:

Trajectory

IsTrajectoryBody

Check whether the object is a body along the trajectory.

Syntax:

IsTrajectoryBody();

Returned value:

Boolean

EstimateObjectList

A list of estimate objects

Properties:

Count – Number of objects.

Items – Objects array.

Properties

Count

Number of objects.

Syntax:

Count;

Data type:

Number

Items

Objects array.

Syntax:

Items;

Data type:

EstimateObject

PanelHoles

A list of panel holes

Properties:

Count – Number of holes.

Holes – Objects array.

Properties

Count

Number of holes.

Syntax:

Count;

Data type:

Number

Holes

Objects array.

Syntax:

Holes;

Data type:

PanelHole

PanelHole

Panel hole

Properties:

Depth – Hole depth.

Diameter – Hole diameter.

Properties

Depth

Hole depth.

Syntax:

Depth;

Data type:

Number

Diameter

Hole diameter.

Syntax:

Diameter;

Data type:

Number

Door design wizard

DoorsMaker

Door design wizard

Methods:

Silent – Construct doors without template selection form opening.

ShowErrors – Show a list of errors after construction.

Save – Save door installation parameters to a file.

Load – Download door installation parameters from a file.

Setup – Install doors into section.

Methods

Silent

Construct doors without template selection form opening.

Syntax:

Silent();

Input parameters:

Boolean

ShowErrors

Show a list of errors after construction.

Syntax:

ShowErrors();

Input parameters:

Boolean

Save

Save door installation parameters to a file.

Syntax:

Save(filename);

Input parameters:

filename (string)

Load

Download door installation parameters from a file.

Syntax:

Load(filename);

Input parameters:

filename (string)

Setup

Install doors into section.

Syntax:

Setup(LeftObject,RightObject,TopObject,BottomObject);

Input parameters:

LeftObject – left bound, Panel or Edge

RightObject – right bound, Panel or Edge

TopObject – top bound, Panel or Edge

BottomObject – lower bound, Panel or Edge

Drawer design wizard

BoxesMaker

Drawer design wizard

Methods:

ShowErrors – Show a list of errors after construction.

Save – Save drawer installation parameters to a file.

Load – Download drawer installation parameters from a file.

Setup – Install drawers into section.

Methods

ShowErrors

Show a list of errors after construction.

Syntax:

ShowErrors();

Input parameters:

Boolean

Save

Save drawer installation parameters to a file.

Syntax:

Save(filename);

Input parameters:

filename (string)

Load

Download drawer installation parameters from a file.

Syntax:

Load(filename);

Input parameters:

filename (string)

Setup

Install drawers into section.

Syntax:

Setup(LeftObject,RightObject,TopObject,BottomObject);

Input parameters:

LeftObject – left bound, Panel

RightObject – right bound, Panel

TopObject – top bound, Panel or Edge

BottomObject – lower bound, Panel or Edge

BAZIS-Salon

TovarItems

Information about item

Properties:

TovarName – model name (item).

TovarModel – a list of all model objects (item).

Price – product price.

IsNotStandart – standard or nonstandard product.

KodTovID – additional item code.

Metka – show tag and dimensions.

Discount – product discount.

Markup – product markup.

Methods:

FindByName – element search by name.

Properties

TovarName

Model name (item).

Syntax:

TovarName;

Data type:

string

TovarModel

Model 3D (all model objects list).

Syntax:

TovarModel;

Data type:

List3D

Price

Item price.

Syntax:

Price;

Data type:

double

IsNotStandart

Standard or nonstandard item.

Syntax:

IsNotStandart;

Data type:

boolean

KodTovID

Additional item code.

Syntax:

KodTovID;

Data type:

string

Metka

Show tag and dimensions.

Syntax:

Metka;

Data type:

boolean

Discount

Product discount.

Syntax:

Discount;

Data type:

number

Note:

Integer value.

Markup

Product markup.

Syntax:

Markup;

Data type:

number

Note:

Integer value.

Methods

FindByName

Search element by name.

Syntax:

FindByName(name, CaseSensitive);

Input parameters:

name – element name, string

CaseSensitive – case-sensitive search, boolean (true – take into account, false – not)

Returned value:

TScItemTovar

TScItemTovar

Item element properties

Editable properties:

Article – element article.

Name – element name.

Count – Item elements count.

Material – item current material.

Price – element price.

Properties only for reading:

GroupMaterial – element article.

TypeElement – element type.

ObjList – a list of 3D item objects.

Properties

Article

element article.

Syntax:

Article;

Data type:

string

Name

Element name.

Syntax:

Name;

Data type:

string

Count

Item elements count.

Syntax:

Count;

Data type:

double

Material

Element current material.

Syntax:

Material;

Data type:

string

Price

Element price.

Syntax:

Price;

Data type:

double

GroupMaterial

Group of material.

Syntax:

GroupMaterial;

Data type:

string

Note:

Property only for reading

TypeElement

Element type.

Syntax:

TypeElement;

Data type:

string

Note:

Property only for reading

Note:

Property only for reading

ObjList

A list of element 3D objects.

Syntax:

ObjList;

Data type:

List3D

Note:

Property only for reading

SalonUtils

Internal functions of BAZIS-Salon

Properties:

PathAttachments – folder path with attached files.

SalonId – Salon ID.

Methods:

GetFullPathAttachment – get file path.

Properties

PathAttachments

Folder path with attached files.

Syntax:

PathAttachments;

Data type:

string

Note:

Property only for reading

SalonId

Salon ID.

Syntax:

SalonId;

Data type:

string

Methods

GetFullPathAttachment

Element search by name.

Syntax:

FindByName(name, CaseSensitive);

Input parameters:

fileName – file name, string

Returned value:

If file is in the folder with attached files, file full path will be returned, otherwise only its name.

UserProperties

Custom properties

Properties:

Value – value by property name.

Name – get property name by index.

Count – properties count.

Methods:

Remove – delete property by name.

Properties

Value

Value by property name.

Syntax:

Value;

Data type:

Array

Name

Get property name by index.

Syntax:

Name;

Data type:

Array.<String>

Count

Properties count.

Syntax:

Count;

Data type:

Number

Methods

Remove

Delete property by name.

Syntax:

Remove(name);

Input parameters:

name – property name, string

Appendix I. Screen form creation in visual mode

Screen form can be created in visual mode. Script text will be generated automatically, which will ensure that this form is created at launch. To perform the constructions call **Scripts** – **Script editor** command. **Script editor** dialogue will appear on the screen (fig. 1).

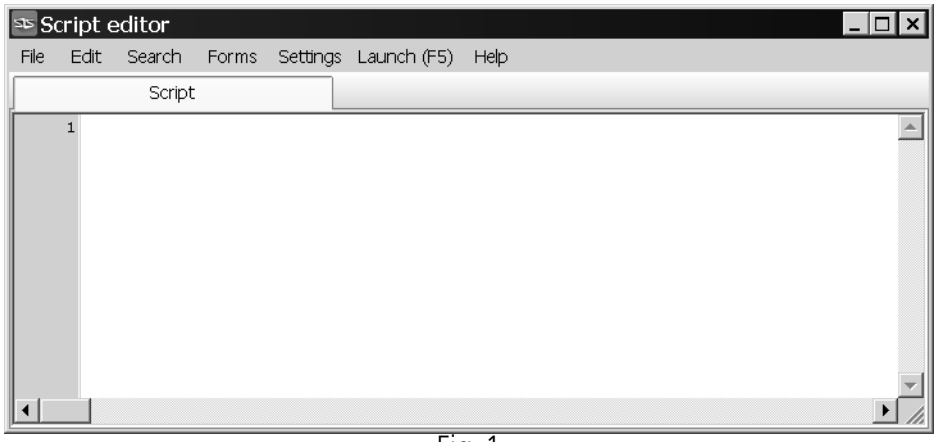


Fig. 1.

In this dialogue call **Forms** – **Add form** command. **Form editor** dialogue will appear on the screen (fig. 2).

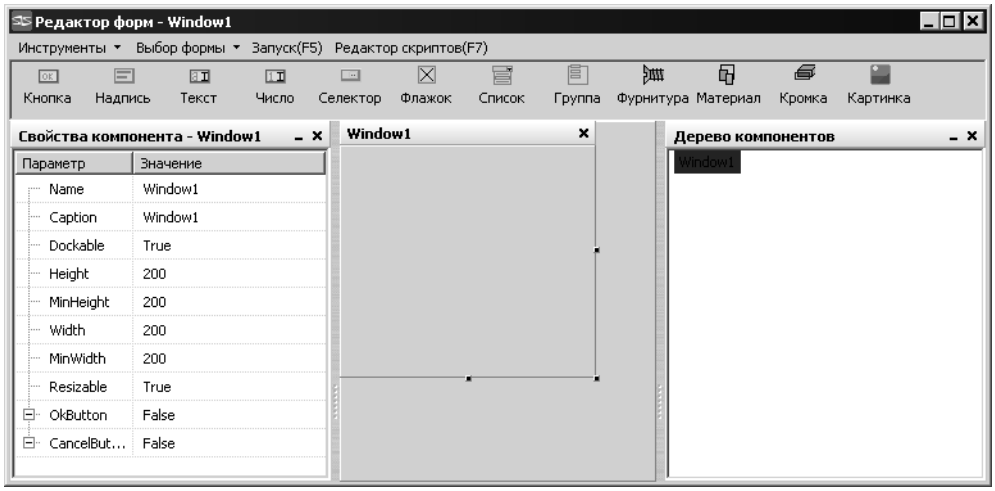


Fig. 2.

In this dialogue screen form with default parameters is created automatically. Properties of this component, currently the only one, are shown on **Component properties** panel. **Component tree** panel also contains the only component designation, form window. These panels availability in **Form editor** window is specified with **Tools** menu commands. Component geometric dimensions can be edited via mouse cursor or its property setting. Control elements on **Component properties** panel allow to change other properties. To check component functioning call **Launch** command. Created form will appear on the screen. To continue its editing, close the form.

The results of visual design lead to automatic generation of script text. It is created in **Script editor** dialogue, that remains open during visual design. **Script editor** command allows to make this dialogue active.

Component panel contains control elements designations, that can be used for screen form creation. *Button, Title, Text, Digit, Selector, Checkmark, List* and *Image* components are standard control elements. *Accessories, Material, Edge* components are specific for Material DB. *Group* component provides the ability to create hierarchical structure of screen form components. To place the component to the form, use standard Drag-and-Drop technology. When the component is placed in the window, it will appear in the structure of form objects on **Component tree** panel.

Several forms can be opened in editor simultaneously. Commands of **Form choice** menu allow to choose a required one for editing. To create a new form, call **Add form** command of this menu.

To finish visual design of screen forms, close **Form editor** window.

A
Action *48*
Action3D *69*
ArrangePositions *77*
AsyncExec *74*
BeginOrtho3 *75*
BlinkHint *72*
Cancel *74*
ChooseMaterial *78*
Commit *74*
Continue *74*
CursorToClosestLine *76*
CursorToClosestPoint *75*
CursorToMiddleOfLine *76*
EndOrtho3 *75*
ErrorHint *72*
Find3DPoint *76*
Find3DPointXZPlane *76*
Finish *74*
Get3Dobject *76*
Hint *72*
Interactive *70*
LoadModel *76*
ModelFilename *74*
MouseX *70*
MouseY *70*
NewFragment *77*
NewFurniture *77*
NewModel *77*
OnClick *72*
OnFinish *73*
OnMove *73*
OnRayTraceFinished *73*
OnStart *73*
Pos3 *71*
Properties *73*
RayTraceScene *76*
ReplaceFurniture *78*
Revert *77*
RightDir *71*
SaveModel *77*
ShowEdges *72*
ShowPoints *71*
UpDir *71*
ViewDir *71*
ActiveMaterial *48*
AddAssembly *55*
AddBlock *55*
AddCopy *56*
AddDraftBlock *56*
AddExtrusion *54*
AddFrontPanel *52*
AddHorizPanel *53*
AddPanel *52*
AddTrajectory *54*
AddVertPanel *53*
alert *49*
AnimationType *48*
Arc2D *164*
ArcDir *165*
Center *165*
Pos1 *164*
Pos2 *164*
Assembly *125*
AnimType *125*
Axis_X *43*
Axis_Y *44*
Axis_Z *44*
AxisX *43*
AxisY *43*
AxisZ *43*

B
BeginBlock *57*
Block *125*
IsFastener *125*
BoxesMaker *176*
Load *176*
Save *176*
Setup *176*
ShowErrors *176*

C
Circle2D *165*
Center *165*

CirRadius *165*
Dir *166*
Clear *83*
confirm *49*
Contour2D *134*
AddArc *139*
AddArc3 *139*
AddCircle *138*
AddEquidistant *143*
AddEquidistantRecursive *139*
Addition *140*
AddLine *138*
AddRectangle *138*
AddRoundRect *138*
Clear *137*
Count *136*
Elastic *142*
Facet *141*
FacetEx *140*
Find *141*
Fit *141*
Height *137*
IsClosedContour *143*
IsContourRectangle *144*
IsInContour *143*
IsPointInside *143*
Load *142*
Max *137*
Min *137*
Move *137*
Objects *136*
OrderContours *142*
Rotate *138*
Rounding *141*
RoundingEx *140*
Subtraction *140*
Symmetry *142*
Width *136*
Count *68*

D
DatumMode *46*
DeleteNewObjects *56*
DeleteObject *56*
DoorsMaker *174*
Load *175*
Save *175*
Setup *175*
ShowErrors *175*
Silent *175*

E
Edge3 *104*
First *104*
GFirst *104*
GLast *104*
Last *104*
Elem2D *161*
AsArc *163*
AsCircle *162*
AsEllipse *162*
AsLine *162*
AsList *163*
ElType *161*
IsArc *162*
IsCircle *162*
IsEllipse *162*
IsLine *161*
IsList *163*
ObjLength *163*
ElementType *47*
Ellipse2D *166*
Center *166*
Dir *167*
MajorAxisAngle *167*
MajorRadius *166*
MinorRadius *166*
EndBlock *57*
ErrorType *46*
EstimateObject *170*
AsButt *171*
AsExtrusionBody *172*
AsPanel *171*
AsPlastic *171*

AsTrajectoryBody *172*
IsButt *171*
IsExtrusionBody *172*
IsPanel *171*
IsPlastic *172*
IsTrajectoryBody *172*
EstimateObjectList *173*
Count *173*
Items *173*
ExtractMatCode *61*
ExtractMatName *60*
Extrusion *122*
Contour *123*
MaterialName *123*
Thickness *123*

F
FormatMatName *60*
FurnArticle *167*
DatumMode *168*
Name *167*
NameWithCode *168*
OrderCode *168*
OrderName *168*
FurniturePosition *47*
FurnMaterial *101*
Make *102*
Name *101*
Thickness *102*
Width *102*
FurnPositionMode *47*
FurnSchemeType *45*

G
Geometry2D *61*
Area *62*
Compare *61*
Distance *62*
Intersect *61*
GetEdge *52*
GetObject *51*
GetPanel *51*
GetPoint *51*

H
Holes *169*

I
ImportExport
SVG *157*
ImportExportSVG *156*
CurveQuality *156*
GroupElems *156*
Load *157*
Save *156*
InButtMaterial *149*
Allowance *150*
ClipPanel *150*
Name *149*
Overhung *150*
Sign *149*
Thickness *149*
Width *150*
InButton *145*
Caption *145*
Combo *145*
NewSubMenu *146*
InControl *144*
Enabled *144*
Hint *145*
id *144*
OnChange *145*
Visible *144*
InfFurniture *157*
Choose *160*
DatumMode *158*
DatumModeFilter *158*
DecodeFromString *160*
EncodeToString *160*
Make *159*
Mount *158*
Mount1 *158*
MountBox *159*
MountScheme *159*
InFloat *146*
Fixed *146*

ReadOnly *146*
Value *146*
InFurniture *150*
DatumMode *151*
DatumModeFilter *150*
Mount *151*
Mount1 *151*
MountBox *152*
MountScheme *151*
InMateria
Apply *148*
Name *148*
SetActive *148*
Thickness *148*
Width *148*
InMaterial *147*
InNumber *147*
Fixed *147*
ReadOnly *147*
Value *147*
InspectorError *155*
ErrorMessage *155*
ErrorObjects *155*
ErrorObjectsCount *155*
ErrorType *155*
ObjectsNames *156*
InspectorOptions *153*
FastIncorrectAnalyze *154*
FastIntersectionAnalyze *153*
MatNotExistsAnalyze *154*
MatOutOfStockAnalyze *154*
ObjIntersectionAnalyze *153*
PanelNotFixedAnalyze *154*
PanelTooLargeAnalyze *154*
PlasticTooLargeAnalyze *154*
Items *80*

L

Line2D *163*
Pos1 *164*
Pos2 *164*
List3D *115*
Count *116*
ElasticResize *117*
FurnType *117*
IsElastic *116*
Load *117*
Objects *116*
SalonType *117*

M

Material *170*
MaxValue *89*
MinValue *90*
Model *48*
Model3D *67*
Objects *68*
Selected *68*
SelectionCount *68*
Selections *68*
ModelInspector *152*
ErrorList *152*
Options *152*
Run *153*

N

NewButtMaterialInput *59*
NewButtonInput *57*
NewCOMObject *42*
NewContour *42*
NewFloatInput *58*
NewForm *43*
NewFurnitureInput *60*
newlink ScriptForm
ShowModal *101*
NewMaterialInput *59*
NewNumberInput *58*
NewPoint *42*
NewVector *41*

O

Object3 *105*
ArtPos *106*
AsList *107*
AsPanel *107*

Build *115*
GabMax *109*
GabMin *109*
GMax *108*
GMin *109*
GSize *109*
IsOwner *115*
List *107*
Name *106*
NToGlobal *115*
NToObject *114*
Orient *112*
OrientGCS *113*
Owner *106*
Position *108*
PositionX *108*
PositionY *108*
PositionZ *108*
Reflect *113*
Rotate *111*
RotateGCS *111*
RotateX *111*
RotateY *112*
RotateZ *112*
Selected *107*
SetDefaultTransform *110*
ToGlobal *114*
ToObject *114*
Translate *110*
TranslateGCS *111*
UserPropCount *109*
UserProperty *110*
UserPropertyName *110*
Visible *107*
Objects *170*
OpenFurniture *50*
OrientCamera *60*

P

p3dBottom *169*
p3dFront *168*
p3dIsometric *169*
p3dLeft *169*
p3dRight *169*
p3dTop *169*
Pane
Butts *118*
Panel *117*, *169*
AddButt *121*
AddCut *121*
AddPlastic *122*
Contour *118*
ContourHeight *119*
ContourWidth *119*
Cuts *119*
IsButtVisible *122*
MaterialName *120*
MaterialWidth *120*
Plastics *119*
Thickness *119*
PanelButt *127*
Allowance *129*
ClipPanel *129*
CutIndex *129*
ElemIndex *128*
Material *128*
Overhung *129*
Profile *129*
Sign *128*
Thickness *128*
Width *128*
PanelButts *126*
Add *127*
Butts *127*
Count *126*
PanelCut *133*
Contour *134*
Name *133*
Sign *133*
Trajectory *133*
PanelCuts *132*
Add *132*
Count *132*

Cuts *133*
PanelHole *174*
Depth *174*
Diameter *174*
PanelHoles *173*
Count *173*
Holes *173*
PanelPlastic *131*
Material *131*
TextureOrientation *131*
Thickness *131*
PanelPlastics *130*
Add *131*
Count *130*
PanelSideEnum *44*
Plastics *130*
Point *103*
x *103*
y *103*
prompt *49*

R

Result *170*
RootProperties *160*
NewFurnitureValue *160*

S

SalonAnimationType *44*
SalonUtils *181*
GetFullPathAttachment *182*
PathAttachments *182*
SalonId *182*
SchemeFurnPos *46*
ScriptBooleanProperty *89*
ScriptButtonProperty *90*
OnClick *91*
ScriptButtProperty *92*
SetActive *92*
Thickness *92*
Width *92*
ScriptColorProperty *93*
Value *93*
ScriptForm *96*
CancelButton *99*
CancelButtonCaption *99*
Caption *97*
Close *101*
Dockable *100*
Height *98*
Left *98*
MinHeight *98*
MinWidth *98*
OKButton *99*
OKButtonCaption *99*
OnCancelButtonClick *100*
OnClose *99*
OnShow *100*
Properties *97*
Resizable *100*
Show *101*
Top *99*
Visible *98*
Width *97*
ScriptForm_OnOkButtonClick *100*
ScriptFurnitureProperty *92*
Value *93*
ScriptGroupProperty *88*
Image *88*
MaxHeight *88*
ScriptBooleanProperty *89*
ScriptNumberProperty *89*
Scrollable *88*
Value *89*
ScriptMaterialProperty *91*
SetActive *91*
Width *91*
ScriptMenu *93*
Clear *95*
Count *94*
Items *94*
Load *95*
Name *94*
NewBool *96*
NewButton *96*

NewGroup *95*
NewNumber *96*
NewString *96*
OnChange *94*
Save *95*
Store *94*
ScriptProperty *78*
BackColor *82*
ChildrenEnabled *80*
Count *80*
DropDownMenu *82*
Enabled *80*
Expanded *81*
Load *83*
Name *79*
NameEditable *81*
NewBool *85*
NewButt *86*
NewButton *86*
NewCOMObject *83*
NewFurniture *87*
NewGroup *84*
NewImage *84*
NewMaterial *86*
NewNumber *85*
NewSeparator *87*
NewString *85*
OnActivate *82*
OnChange *80*
OnDeactivate *81*
OnValueChange *82*
OnValueValidate *81*
PopupMenu *82*
Save *83*
Tag *83*
Validate *87*
ValueValid *81*
Visible *80*
ScriptStringProperty *89*
SelectAll *50*
SetCamera *51*
StartEditing *57*
System *62*
apiVersion *67*
askFileNameSave *66*
askFolder *66*
askReadTextFile *64*
askWriteTextFile *64*
developerApiVersion *67*
exec *65*
fileExists *63*
getFileName *66*
getFileNameWithoutExtension *66*
include *63*
log *63*
readTextFile *64*
secureExec *64*
writeTextFile *63*
system *48*

T

TextureDir *45*
TovarItems *177*
Discount *178*
FindByName *179*
IsNotStandart *178*
KodTovID *178*
Markup *179*
Metka *178*
Price *178*
TovarModel *177*
TovarName *177*
TProjection3dEnum *44*
Trajectory *124*
Contour2D *124*
MaterialName *124*
Trajectory2D *124*
TScItemTovar *179*
Article *180*
Count *180*
GroupMaterial *181*
Material *180*
Name *180*

ObjList *181*
Price *180*
TypeElement *181*

U
UnSelectAll *50*
UserProperties *182*
Count *183*
Name *183*
Remove *183*
Value *183*

V
Value *89*, *90*, *91*
ValueStep *90*
Vector *102*
x *102*
y *103*
z *103*
ViewAll *50*

W
windowposition *45*

X
x *103*