

# **Autodesk Robot Structural Analysis Professional 2022**

## **Robot Object Model**

© 2021 Autodesk, Inc. All Rights Reserved. Except as otherwise permitted by Autodesk, Inc., this publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose. Certain materials included in this publication are reprinted with the permission of the copyright holder.

#### Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

#### Trademarks

The following are registered trademarks of Autodesk, Inc., in the USA and/or other countries:

Autodesk Robot Structural Analysis, Autodesk Concrete Building Structures, Spreadsheet Calculator, ATC, AutoCAD, Autodesk, Autodesk Inventor, Autodesk (logo), Buzzsaw,

Design Web Format, DWF, ViewCube, SteeringWheels, and Autodesk Revit.

All other brand names, product names or trademarks belong to their respective holders.

#### Third Party Software Program Credits

ACIS Copyright© 1989-2001 Spatial Corp. Portions Copyright© 2002 Autodesk, Inc.

Copyright© 1997 Microsoft Corporation. All rights reserved.

International CorrectSpell™ Spelling Correction System© 1995 by Lernout & Hauspie Speech Products, N.V. All rights reserved.

InstallShield™ 3.0. Copyright© 1997 InstallShield Software Corporation. All rights reserved.

PANTONE® and other Pantone, Inc. trademarks are the property of Pantone, Inc.© Pantone, Inc.,2002.

Portions Copyright© 1991-1996 Arthur D. Applegate. All rights reserved.

Portions relating to JPEG © Copyright 1991-1998 Thomas G. Lane. All rights reserved. Portions of this software are based on the work of the Independent JPEG Group.

Portions relating to TIFF © Copyright 1997-1998 Sam Leffler. © Copyright 1991-1997 Silicon Graphics, Inc. All rights reserved.

#### Government Use

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR 12.212 (Commercial Computer Software-Restricted Rights) and DFAR 227.7202 (Rights in Technical Data and Computer Software), as applicable.

# Table of Contents

<b>RobotOM Namespace</b>	<b>1</b>
<b>Robot Object Model</b>	<b>1</b>
Data Server	1
Basic definitions	2
Querying mechanism	2
Labels - complex attributes of objects	16
IRobotPredefinedLabel	28
Uniform access to structure components	29
Structure components	36
Load cases	37
Load records	38
IRobotLoadRecordCommon	80
IRobotLoadRecordCommon Members	80
IRobotLoadRecordCommon Fields	80
IsAutoGenerated	80
Snow/wind loads	81
Generating 3D loads	81
Code combinations	147
Modal analysis parameters	187
Modal analysis parameters recognizing static forces	187
Seismic analysis parameters	207
IRobotSeismicResidualModeDefinitionType	291
IRobotSeismicResidualMode	292
IRobotSeismicResidualMode Members	292
IRobotSeismicResidualMode Fields	292
AugmentationFactor	292
DefinitionMethod	293
IsActive	293
LimitFrequency	293
IRobotSeismicAnalysis_EC8_General_Params	293
IRobotSeismicAnalysis_EC8_General_Params Members	294
IRobotSeismicAnalysis_EC8_General_Params Fields	294
Ag	294
B	295
BehaviorFactor	295
Direction	295
DirectionType	296
ExcitationDir	296

Filter	296
ResidualMode	296
S	297
Spectrum	297
Tb	297
Tc	297
Td	298
IRobotSeismicAnalysis_EC8_SpectrumType	298
IRobotSeismicAnalysis_EC8_GroundType	298
IRobotSeismicAnalysis_EC8_Params	299
IRobotSeismicAnalysis_EC8_Params Members	299
IRobotSeismicAnalysis_EC8_Params Fields	299
Ag	300
BehaviorFactor	300
Direction	300
DirectionType	301
ExcitationDir	301
Filter	301
GroundType	301
ResidualMode	302
Spectrum	302
SpectrumType	302
Spectral analysis parameters	302
Non-linear analysis parameters	314
Moving loads	323
Buckling analysis parameters	356
Time history analysis parameters	360
IRobotTimeHistoryHHTParams	382
IRobotTimeHistoryHHTParams Members	383
IRobotTimeHistoryHHTParams Fields	383
Alpha	383
Beta	383
CoeffAlpha	384
Nonlinearity	384
NonlinearParams	384
IRobotTimeHistoryNewmarkAccelParams	385
IRobotTimeHistoryNewmarkAccelParams Members	385
IRobotTimeHistoryNewmarkAccelParams Fields	385
Alpha	385
Beta	386
Nonlinearity	386
NonlinearParams	386

Harmonic analysis parameters	387
Push-over analysis parameters	388
FRF analysis parameters	392
Footfall analysis parameters	395
Parameters of simplified seismic analysis	443
IRobotSELFSeismicEngine	444
IRobotSELFSeismicEngine Members	444
IRobotSELFSeismicEngine Fields	444
GenerationParams	444
IRobotSELFSeismicEngine Methods	445
GenerateLoadCases	445
IRobotSELFSeismicGenerationParams	445
IRobotSELFSeismicGenerationParams Members	446
IRobotSELFSeismicGenerationParams Fields	446
CodeName	446
CodeNumber	446
ExcitationDir	447
ModalCaseParams	447
SeismicParams	447
TBaseMethod	447
IRobotSELFSeismicStructureParams	448
IRobotSELFSeismicStructureParams Members	448
IRobotSELFSeismicStructureParams Fields	448
BaseLevelCoordZ	448
BaseLevelDefMethod	449
BaseLevelStorey	449
TopLevelCoordZ	449
TopLevelDefMethod	450
TopLevelStorey	450
IRobotSELFSeismicLevelDefinitionMethod	450
IRobotSELFSeismicTBaseMethod	451
IRobotSELFSeismicAnalysisParams	451
IRobotSELFSeismicAnalysisParams Members	451
IRobotSELFSeismicAnalysisParams Fields	452
CodeName	452
CodeNumber	452
CodeParams	452
Eccentricities	453
StructureParams	453
TBaseMethod	453
IRobotSELFSeismicAnalysisParams Methods	453
GetExcitationDir	454

IRobotSELFS seismic_ASCE_7_10_SiteClassType	454
IRobotSELFS seismic_ASCE_7_10_Params	454
IRobotSELFS seismic_ASCE_7_10_Params Members	455
IRobotSELFS seismic_ASCE_7_10_Params Fields	455
I	455
R	456
S1	456
SiteClass	456
Ss	456
StructureTypeX	457
StructureTypeY	457
TBaseValueX	457
TBaseValueY	457
TL	458
IRobotSELFS seismic_ASCE_7_10_StructureType	458
IRobotSELFS seismic_EC_8_Params	458
IRobotSELFS seismic_EC_8_Params Members	459
IRobotSELFS seismic_EC_8_Params Fields	459
Ag	459
Beta	460
Q	460
SiteClass	460
SpectrumType	460
StructureTypeX	461
StructureTypeY	461
IRobotSELFS seismic_EC_8_Params Methods	461
SiteClassEnvelopeCheck	461
SiteClassEnvelopesChecked	462
IRobotSELFS seismic_EC_8_StructureType	462
IRobotSELFS seismic_EC_8_SiteClass	462
IRobotSELFS seismic_EC_8_SpectrumType	463
IRobotSELFS seismic_AS_1170_4_Params	463
IRobotSELFS seismic_AS_1170_4_Params Members	464
IRobotSELFS seismic_AS_1170_4_Params Fields	464
Kp	464
Mi	464
Probability	465
SoilCategory	465
Sp	465
StructureTypeX	465
StructureTypeY	466
Z	466

IRobotSELFS seismic_AS_1170_4_SoilCategoryType	466
IRobotSELFS seismic_AS_1170_4_StructureType	467
IRobotSELFS seismic_AS_1170_4_ProbabilityType	467
Wind loads simulation	468
IRobotWindLoadsSimulationEngine	468
IRobotWindLoadsSimulationEngine Members	468
IRobotWindLoadsSimulationEngine Fields	468
Params	468
IRobotWindLoadsSimulationEngine Methods	469
Generate	469
IRobotWindLoadsSimulationParams	469
IRobotWindLoadsSimulationParams Members	469
IRobotWindLoadsSimulationParams Fields	470
DeviationPercent	470
DirectionXNE nabled	471
DirectionXNYN E nabled	471
DirectionXNYP E nabled	471
DirectionXP E nabled	471
DirectionXPYNE nabled	472
DirectionXPYPE nabled	472
DirectionYNE nabled	472
DirectionYPE nabled	472
Elements	473
OpeningsClosed	473
TerrainLevel	473
Velocity	474
IRobotCasePredefinedNumber	474
Nodes	474
Complex attributes of a node	475
Rigid links	475
Compatible nodes	485
Bars	533
Complex attributes of a bar	534
Cables	534
Elastic ground	537
IRobotUpliftDirection	540
IRobotUpliftSense	541
Sections	541
Releases	575
Geometrical imperfections	583
Non-linear hinges	587
Claddings	609

Complex attributes shared by objects of different types	654
Material	654
Non-linear constraints	671
Objects	696
Complex attributes of an object	697
Linear releases	697
Panel calculation model	712
IRobotObjEdgeSelection	801
IRobotObjEdgeSelection Members	801
IRobotObjEdgeSelection Fields	802
Count	802
IRobotObjEdgeSelection Methods	802
FromText	802
Get	803
ToText	803
Finite elements	803
FE mesh generator	804
Grouping of objects	843
IRobotObjectStructuralType	846
Results	847
Calculation results for nodes	847
Calculation results for bars	864
Data structures containing complex calculation results	879
Calculation results for dynamic analyses	919
Calculation results for finite elements	930
IRobotResultStatusType	1027
IRobotCalculationResume	1028
IRobotCalculationResume Members	1028
IRobotCalculationResume Fields	1029
DiagonalStiffnessMatrixMax	1029
DiagonalStiffnessMatrixMin	1029
DiagonalStiffnessMatrixPrecision	1029
EquationSolvingMethodUsed	1030
IRobotCalculationResume Methods	1030
GetEnergy	1030
GetEnergyPrecision	1031
Selections	1031
Edit operations	1042
Stories	1047
Quantity survey	1074
Project	1100
Project components	1100

Spread footing	1101
IRConcrFootingPierType	1123
RC Beam	1123
Continuous Footing	1163
RC Column	1179
IRConcrColumnBucklingModel	1196
IRConcrColumnBucklingModel Members	1196
IRConcrColumnBucklingModel Fields	1196
IsDirectionYOff	1197
IsDirectionZOff	1197
IsSwayY	1197
IsSwayZ	1198
IsTotalStructureHeight	1198
ky	1198
kz	1198
Ly	1199
Lz	1199
NumberOfStories	1199
TotalStructureHeight	1199
IRConcrColumnLoads	1200
IRConcrColumnLoads Members	1200
IRConcrColumnLoads Fields	1200
Count	1200
Item	1201
IRConcrColumnLoads Methods	1201
Add	1201
RemoveAll	1201
IRConcrColumnLoad	1202
IRConcrColumnLoad Members	1202
IRConcrColumnLoad Fields	1202
CaseName	1203
CaseType	1203
Fy	1203
Fz	1204
Gamma	1204
MnsY	1204
MnsZ	1204
MyA	1205
MyB	1205
MyC	1205
MzA	1205
MzB	1206

MzC	1206
N	1206
NdN	1206
IRConcrColumnLoadCaseType	1207
RC Slab	1207
RC Deep Beam	1221
RC Wall	1225
Reinforcement	1238
Drawing	1259
Support of external file formats	1266
Job preferences	1274
Units and formats	1274
IRobotDatabaseType	1317
IRobotCalculationModelCoherence	1318
Backgrounds	1318
IRobotBackgroundLayers	1319
IRobotBackgroundLayers Members	1319
IRobotBackgroundLayers Fields	1320
Count	1320
IRobotBackgroundLayers Methods	1320
FindName	1320
Get	1321
IsImported	1321
SetImported	1321
IRobotBackgroundInsertParams	1321
IRobotBackgroundInsertParams Members	1322
IRobotBackgroundInsertParams Fields	1322
InsertionPoint	1323
Layers	1323
Mirror	1323
Plane	1324
PositionOnNormalAxis	1324
ReferencePoint	1324
RotationAngle	1324
ScalingFactor	1325
Units	1325
IRobotBackground	1325
IRobotBackground Members	1326
IRobotBackground Fields	1326
Color	1326
FilePath	1327
InsertParams	1327

IsOn	1327
Name	1327
Number	1328
VisibilityRange	1328
IRobotBackground Methods	1328
Save	1328
IRobotBackgroundVisibilityRangeType	1329
IRobotBackgroundVisibilityRange	1329
IRobotBackgroundVisibilityRange Members	1330
IRobotBackgroundVisibilityRange Fields	1330
FromPos	1330
ToPos	1330
Type	1330
IRobotBackgroundServer	1331
IRobotBackgroundServer Members	1331
IRobotBackgroundServer Methods	1331
Create	1332
Get	1332
GetAllNumbers	1332
Remove	1333
RemoveAll	1333
IRobotActiveModelType	1351
Presentation of data	1351
Views and layouts	1352
Tables	1352
Graphical structure views	1372
Panel cuts	1374
Influence lines	1378
IRobotViewScreenCaptureResolution	1455
Views for RTF format files	1455
Screen captures	1458
Views for HTML format files	1460
IRobotDialogId	1469
Printouts	1469
Structural axes	1512
Application - main object of the model	1527
IRobotLicenseEntitlement	1539
IRobotLicenseEntitlementStatus	1539
IRobotCalculationsType	1540
Add-ins Manager	1540
Calculation module	1552
IRobotCalcEngineEvents	1584

IRobotCalcEngineEvents Members	1584
IRobotCalcEngineEvents Methods	1584
CalcMessage	1584
IRobotCalcMessageSource	1585
IRobotCalcMessageSeverityLevel	1585
IRobotCalculationStatus	1586
IRobotCalculationMode	1586
Direct Analysis Method	1587
IRobotDAMCalcModule	1587
IRobotDAMCalcModule Members	1588
IRobotDAMCalcModule Fields	1588
IsActive	1588
Params	1588
IRobotDAMCalcModule Methods	1589
DeleteModel	1589
Run	1589
IRobotDAMAnalysisType	1589
IRobotDAMNotionalLoads	1590
IRobotDAMNotionalLoads Members	1590
IRobotDAMNotionalLoads Fields	1591
ActiveXN	1591
ActiveXP	1591
ActiveYN	1592
ActiveYP	1592
Coefficient	1592
GravityLoadCombEnabled	1593
LateralLoadCombEnabled	1593
LateralLoadCombType	1593
IRobotDAMLateralLoadCombType	1594
IRobotDAMReducedStiffness	1594
IRobotDAMReducedStiffness Members	1594
IRobotDAMReducedStiffness Fields	1595
RCBeamsValue	1595
RCColumnsAndWallsValue	1595
RCSlabsValue	1596
SteelMembersReductionType	1596
SteelMembersTauBValue	1596
SteelMembersValue	1597
IRobotDAMSteelMembersReductionType	1597
IRobotDAMParams	1597
IRobotDAMParams Members	1598
IRobotDAMParams Fields	1598

Analysis	1598
NotionalLoads	1599
ReducedStiffness	1599
IRobotDAMPParams Methods	1599
GetNLPDParams	1600
SetNLPDParams	1600
Update	1600
Global data types	1600
Geometric data types	1601
IRobotGeoPlane	1627
Concrete	1656
Supplementing the required reinforcement module with new codes	1656
IRBestForceDataIntegerValue	1690
IRBestForceDataSLSCombType	1691
IRBestCodeService2	1691
IRBestCodeService2 Members	1691
IRBestCodeService2 Fields	1692
RobotVersion	1692
ShowBucklingButton	1692
Plate and shell reinforcement	1692
IRConcrPlateCodeService2	1778
IRConcrPlateCodeService2 Members	1779
IRConcrPlateCodeService2 Fields	1779
RobotVersion	1779
ShowLongTermCracking	1779
Design of RC members	1779
Supplementing RC modules with new codes	1791
IRConcrCodeService	1792
IRConcrCodeService Members	1792
IRConcrCodeService Fields	1792
Beam	1792
Column	1793
IRConcrCodeService Methods	1793
IsConcrComponentServed	1793
IRConcrCodeColumn	1793
IRConcrCodeColumn Members	1794
IRConcrCodeColumn Methods	1794
IsCommandEnabled	1794
Verification	1794
IRConcrCodeColumnCommand	1795
IRConcrCodeReport	1795
IRConcrCodeReport Members	1795

IRConcrCodeReport Methods	1795
AddError	1796
AddMessage	1796
AddWarning	1796
IRConcrCodeBeam	1796
IRConcrCodeBeam Members	1797
IRConcrCodeBeam Methods	1797
IsCommandEnabled	1797
Verification	1797
IRConcrCodeBeamCommand	1798
Steel and timber design	1798
EC3 Code	1800
CB71 Code	1844
Member section definition module	1928
Connection module	1929
Knee connection	1930
Column base connections	1960
Pinned column base	1961
Fixed column base	1970
Concrete column base	1979
Angle connection	2019
Tube connection	2031
Gusset plate connections	2048
Beam-principal beam connection	2111
External parameters	2168
Robot Kernel	2195
STR files analyzer	2206
Structure correction	2209

## Index

ee

# RobotOM Namespace

This is namespace RobotOM.

---

## Robot Object Model

When modeling a structure, one uses a certain set of concepts that facilitate the process of defining and analyzing the structure. Here belong node, bar, load case and many others. While working in Robot, one encounters such concepts as project – represented by the RTD file, graphical viewers and tables – presenting different aspects of data describing a structure, as well as many other abstractions. In order to make it possible to use the above-mentioned concepts at the level of a programming language, they have been formalized and defined according to COM standards. Program members and data types used to define structures are described in Robot Object Model by means of appropriate interfaces. An interface is understood to mean a set of data, mutually linked in a logical manner, and a set of operations that can be performed on the data. The data are called attributes or members, and operations that can be performed are called functions or methods of the interface. The set of interface functions defines its functionality. Each interface has got a unique name. For instance, a node of a structure is represented by means of the IRobotNode interface. Among the data of the interface one finds, among others, three real numbers corresponding to node coordinates. In order to take advantage of the functionality of a given interface, one should create an object that realizes the interface. The manner of creating an object depends on the currently used programming language. In Visual Basic, for instance, a new object is created by means of the New operator or CreateObject function. Additionally, there is a certain interface type that describes the set of constant values or identifiers. Such interface represents enumeration type. The IRobotProjectType interface is a good example here. Identifiers referring to different types of projects available in Robot are its members. In order to use a member of such an interface, it suffices to provide its name. There is no necessity to create an object to realize it. Robot Object Model is a set of such interfaces. It is contained in the robotom.tlb file in the standard binary format, so that interfaces available in the model could be recognized by the compilers of programming languages and by the system software. The TLB standard extension comes from the Type Library name, as the interface is often, especially at the level of a programming language, identified with data type. Therefore, one can refer to an object realizing the interface called InterfaceName by calling it an object of the InterfaceName type.

### Naming conventions

All names begin with a capital letter. If a name consists of more than one word, each name element begins with a capital letter. Additionally, an interface name begins with the IRobot prefix, but for interfaces associated with a specialized module the prefix corresponds to the module name (eg IRJoint for interfaces defining the connection module of Robot). The convention has been adopted to minimize the possibility of conflicting names in programming languages that cannot use many independent name spaces. The names of members of enumeration interface consist, in turn, of capital letters, and particular segments are separated by the underline sign ("\_"). Additionally, each name begins with I\_FLIN\_, where FLIN refers to the first letters of interface name, with the prefix omitted. For instance, names of all elements of IRobotProjectType interface begin with the I\_PT\_ prefix (e.g. I\_PT\_FRAME\_2D). Such convention facilitates the creation of unique names. The uniqueness of names of members of enumeration interfaces is necessary, for they are often interpreted in programming languages as global constants that should have unique names.

Units All functions of Robot Object Model use the standard SI units, independently of the current settings in Robot options. .

---

## Data Server

All types of data used to define and model a structure (nodes, bars, load cases) are managed by the Robot Object Model

and the Data Server. The server also provides access to the results of calculations carried out for the defined structures. .

## Interfaces

	Name	Description
↪	IRobotStructure ( <a href="#">see page 1085</a> )	RobotStructure represents an entire structure. Particular structure components are represented by appropriate interface components. .
↪	IRobotStructureCache ( <a href="#">see page 1093</a> )	Interface enabling buffered access to a structure. It enables adding quickly a whole component set to a structure. To add a whole data set to a structure, ApplyCache function of RobotStructure interface should be called up. .
↪	IRobotStructureApplyInfo ( <a href="#">see page 1096</a> )	Interface providing access to information about structure components generated as a result of inserting a new data set to a structure. .
↪	IRobotStructureMergeData ( <a href="#">see page 1097</a> )	Interface defining data that may be merged into a structure by means of the Merge function.
↪	IRobotStructureEvents ( <a href="#">see page 1099</a> )	Events of RobotStructure object.

## I Basic definitions

Data Server manages a large amount of different types of data describing structures. In order to simplify and unify the manipulation of the data, certain basic abstractions have been defined. They allow one to classify the data. Each structure component is represented in the Data Server by means of an object of the appropriate type. All objects of the same type have the same set of attributes and the same functionality described with the set of operations that may be performed on the object. The set of attributes and operations is described formally by means of the appropriate interface. For instance, attributes and complete functionality of a structure node is described by the RobotNode interface. The object type determines univocally how it is possible to work with the object in question, and it defines the access interface for object attributes and functionalities. .

## Enumerations

	Name	Description
⇨	IRobotObjectType ( <a href="#">see page 36</a> )	RobotObjectType defines identifiers for all types of objects managed by the Data Server .

## I.1 Querying mechanism

Data Server defines a simple querying mechanism that allows one to select only the relevant structure elements from among all the structure elements managed by the server. The selection process consists of two phases: determining the structure elements that are of interest for the designer, and taking them from the server and performing an operation on the elements. Selection defined by means of the object of RobotSelection type is used to determine the relevant sub-set of objects - structure components. Thus, a new selection is to be defined by creating a new object of RobotSelection type. By means of the operations available for an object of RobotSelection type, one may describe the selection of the relevant structure components. On the basis of the RobotSelection object, one may obtain a set of objects that meet the criteria determined in Selection. The mode of access to such a set is defined by means of the Collection described formally by the RobotCollection type. The collection simplifies the access to all its elements, orders and indexes them with numbers from 1 to the number of elements in the collection. Thus, it is easy to browse through the objects belonging to the collection. A collection created on the basis of the formerly defined selection allows one to get access to all the object that meet the selection criteria. .

## Interfaces

	Name	Description
↪	IRobotSelection ( <a href="#">see page 3</a> )	RobotSelection allows one to define selections of particular structure components .

	IRobotCollection ( <a href="#">see page 9</a> )	The RobotCollection type is defined to unify the access mode to the sub-set of objects of a given type. Each object belonging to a collection is linked with a number - the index of the object in the collection. In order to obtain an object, one should provide its index. Indexes assume values from 1 to Count ( <a href="#">see page 10</a> ), where Count ( <a href="#">see page 10</a> ) is the number of all objects in the collection. One should differentiate between the object index and the user-defined object number (e.g. node number). The user-defined number of an object is ascribed permanently to the object, unless the user changes... more ( <a href="#">see page 9</a> )
	IRobotMultiSelection ( <a href="#">see page 11</a> )	RobotMultiSelection interface describes a set of many selections of different types. Multi-selection may be useful to describe complex selections of objects of more than one type..
	IRobotMultiCollection ( <a href="#">see page 13</a> )	The RobotMultiCollection groups many collections of objects of various types..
	IRobotModeSelection ( <a href="#">see page 15</a> )	Interface describing selection of a mode for a modal load case..

## I.1.1 IRobotSelection

### Class Hierarchy

#### C++

```
interface IRobotSelection : IDispatch;
```

#### C#

```
public interface IRobotSelection;
```

### Visual Basic

```
Public Interface IRobotSelection
```

### Description

RobotSelection allows one to define selections of particular structure components .

#### I.1.1.1 IRobotSelection Members

The following tables list the members exposed by IRobotSelection.

### Public Fields

	Name	Description
	Count ( <a href="#">see page 4</a> )	Number of objects in the selection.
	Type ( <a href="#">see page 4</a> )	Types of objects described by selections (e.g. I_OT_NODE for node selection). The object type is applied when a selection is created and it may not be changed. .

### Public Methods

	Name	Description
	Add ( <a href="#">see page 5</a> )	Adding the indicated selection to the current one..
	AddOne ( <a href="#">see page 5</a> )	Function adds to the selection a single object with the user specified number.
	AddText ( <a href="#">see page 6</a> )	Function adds to the selection a list of objects described with the specified text.
	And ( <a href="#">see page 6</a> )	Operation of logical product with the selection provided as the argument .
	AndText ( <a href="#">see page 6</a> )	Function performs the logical product operation of the selection and the object list described with the specified text. .
	Clear ( <a href="#">see page 7</a> )	Resetting selection - selection is empty after this operation .
	Contains ( <a href="#">see page 7</a> )	Function returns the True value if the selection includes an object with the user-specified number.

	Exclude ( <a href="#">see page 7</a> )	Logical complement of the current selection and the selection indicated as the argument .
	ExcludeOne ( <a href="#">see page 7</a> )	Function deletes from the selection a single object with the user specified number.
	ExcludeText ( <a href="#">see page 8</a> )	Function deletes from the selection a list of objects described with the specified text.
	FromText ( <a href="#">see page 8</a> )	Robot allows one to define a selection of particular structure components by introducing a text - list describing the selection in the appropriate syntax. Such text may also be used as a parameter of the FromText method that creates on the basis of the text the relevant selection. A text provided as the parameter should describe the selection in the currently used interface language of Robot. .
	Get ( <a href="#">see page 8</a> )	Function returns the user number for the successive object included in the selection.
	ToText ( <a href="#">see page 9</a> )	Returns a selection description in the form of the appropriate text .

### I.1.1.2 IRobotSelection Fields

The fields of the IRobotSelection class are listed here.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 4</a> )	Number of objects in the selection.
	Type ( <a href="#">see page 4</a> )	Types of objects described by selections (e.g. I_OT_NODE for node selection). The object type is applied when a selection is created and it may not be changed. .

#### I.1.1.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As long
```

##### Description

Number of objects in the selection.

##### Version

Available since version 8.2.

#### I.1.1.2.2 Type

##### C++

```
HRESULT get_Type(IRobotObjectType*);
```

##### C#

```
public IRobotObjectType Type { get; }
```

##### Visual Basic

```
Public ReadOnly Type As IRobotObjectType
```

##### Description

Types of objects described by selections (e.g. I\_OT\_NODE for node selection). The object type is applied when a selection is created and it may not be changed. .

### I.1.1.3 IRobotSelection Methods

The methods of the IRobotSelection class are listed here.

#### Public Methods

	Name	Description
⊕	Add ( [ see page 5 )	Adding the indicated selection to the current one. .
⊕	AddOne ( [ see page 5 )	Function adds to the selection a single object with the user specified number.
⊕	AddText ( [ see page 6 )	Function adds to the selection a list of objects described with the specified text.
⊕	And ( [ see page 6 )	Operation of logical product with the selection provided as the argument .
⊕	AndText ( [ see page 6 )	Function performs the logical product operation of the selection and the object list described with the specified text. .
⊕	Clear ( [ see page 7 )	Resetting selection - selection is empty after this operation .
⊕	Contains ( [ see page 7 )	Function returns the True value if the selection includes an object with the user-specified number.
⊕	Exclude ( [ see page 7 )	Logical complement of the current selection and the selection indicated as the argument .
⊕	ExcludeOne ( [ see page 7 )	Function deletes from the selection a single object with the user specified number.
⊕	ExcludeText ( [ see page 8 )	Function deletes from the selection a list of objects described with the specified text.
⊕	FromText ( [ see page 8 )	Robot allows one to define a selection of particular structure components by introducing a text - list describing the selection in the appropriate syntax. Such text may also be used as a parameter of the FromText method that creates on the basis of the text the relevant selection. A text provided as the parameter should describe the selection in the currently used interface language of Robot. .
⊕	Get ( [ see page 8 )	Function returns the user number for the successive object included in the selection.
⊕	ToText ( [ see page 9 )	Returns a selection description in the form of the appropriate text .

#### I.1.1.3.1 Add

##### C++

```
HRESULT Add(IRobotSelection* _sel);
```

##### C#

```
public void Add(IRobotSelection _sel);
```

##### Visual Basic

```
Public Sub Add(ByRef _sel As IRobotSelection)
```

##### Description

Adding the indicated selection to the current one. .

#### I.1.1.3.2 AddOne

##### C++

```
HRESULT AddOne(long _obj_num);
```

##### C#

```
public void AddOne(long _obj_num);
```

**Visual Basic**

```
Public Sub AddOne(_obj_num As long)
```

**Description**

Function adds to the selection a single object with the user specified number.

**Version**

Available since version 8.2.

**I.1.1.3.3 AddText****C++**

```
HRESULT AddText(BSTR _sel_text);
```

**C#**

```
public void AddText(String _sel_text);
```

**Visual Basic**

```
Public Sub AddText(_sel_text As String)
```

**Description**

Function adds to the selection a list of objects described with the specified text.

**Version**

Available since version 8.2.

**I.1.1.3.4 And****C++**

```
HRESULT And(IRobotSelection* _sel);
```

**C#**

```
public void And(IRobotSelection _sel);
```

**Visual Basic**

```
Public Sub And(ByRef _sel As IRobotSelection)
```

**Description**

Operation of logical product with the selection provided as the argument .

**I.1.1.3.5 AndText****C++**

```
HRESULT AndText(BSTR _sel_text);
```

**C#**

```
public void AndText(String _sel_text);
```

**Visual Basic**

```
Public Sub AndText(_sel_text As String)
```

**Description**

Function performs the logical product operation of the selection and the object list described with the specified text. .

**Version**

Available since version 8.2.

**I.1.1.3.6 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

Resetting selection - selection is empty after this operation .

**I.1.1.3.7 Contains****C++**

```
HRESULT Contains(long _obj_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool Contains(long _obj_num);
```

**Visual Basic**

```
Public Function Contains(_obj_num As long) As Boolean
```

**Description**

Function returns the True value if the selection includes an object with the user-specified number.

**Version**

Available since version 8.2.

**I.1.1.3.8 Exclude****C++**

```
HRESULT Exclude(IRobotSelection* _sel);
```

**C#**

```
public void Exclude(IRobotSelection _sel);
```

**Visual Basic**

```
Public Sub Exclude(ByRef _sel As IRobotSelection)
```

**Description**

Logical complement of the current selection and the selection indicated as the argument .

**I.1.1.3.9 ExcludeOne****C++**

```
HRESULT ExcludeOne(long _obj_num);
```

**C#**

```
public void ExcludeOne(long _obj_num);
```

**Visual Basic**

```
Public Sub ExcludeOne(_obj_num As long)
```

**Description**

Function deletes from the selection a single object with the user specified number.

**Version**

Available since version 8.2.

**I.1.1.3.10 ExcludeText****C++**

```
HRESULT ExcludeText(BSTR _sel_text);
```

**C#**

```
public void ExcludeText(String _sel_text);
```

**Visual Basic**

```
Public Sub ExcludeText(_sel_text As String)
```

**Description**

Function deletes from the selection a list of objects described with the specified text.

**Version**

Available since version 8.2.

**I.1.1.3.11 FromText****C++**

```
HRESULT FromText(BSTR _sel_text);
```

**C#**

```
public void FromText(String _sel_text);
```

**Visual Basic**

```
Public Sub FromText(_sel_text As String)
```

**Description**

Robot allows one to define a selection of particular structure components by introducing a text - list describing the selection in the appropriate syntax. Such text may also be used as a parameter of the FromText method that creates on the basis of the text the relevant selection. A text provided as the parameter should describe the selection in the currently used interface language of Robot. .

**I.1.1.3.12 Get****C++**

```
HRESULT Get(long _idx, long* ret);
```

**C#**

```
public long Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As long
```

**Description**

Function returns the user number for the successive object included in the selection.

**Version**

Available since version 8.2.

**I.1.1.3.13 ToText****C++**

```
HRESULT ToText(BSTR* ret);
```

**C#**

```
public String ToText();
```

**Visual Basic**

```
Public Function ToText() As String
```

**Description**

Returns a selection description in the form of the appropriate text .

**I.1.2 IRobotCollection****Class Hierarchy****C++**

```
interface IRobotCollection : IDispatch;
```

**C#**

```
public interface IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotCollection
```

**Description**

The RobotCollection type is defined to unify the access mode to the sub-set of objects of a given type. Each object belonging to a collection is linked with a number - the index of the object in the collection. In order to obtain an object, one should provide its index. Indexes assume values from 1 to Count ( see page 10), where Count ( see page 10) is the number of all objects in the collection. One should differentiate between the object index and the user-defined object number (e.g. node number). The user-defined number of an object is ascribed permanently to the object, unless the user changes the number. The index, in turn, corresponds to the position of the object in the given collection - the same object may have different indexes in different collections. Indexes are ascribed to objects in a given collection in such a way that the increasing index values correspond to the increasing values of user-defined numbers ascribed to the objects. For instance, if a collection consists of three objects with the user-defined numbers 24, 105 and 214, the objects obtain the following indexes in the same collection: 1, 2, 3 ..

**I.1.2.1 IRobotCollection Members**

The following tables list the members exposed by IRobotCollection.

## Public Fields

	Name	Description
◆	Count (see page 10)	Number of all objects in a collection (it is also the index of the last element of the collection) .

## Public Methods

	Name	Description
◆	Get (see page 10)	The function returns an object with the indicated index in the collection. The type of returned object depends on the collection type - for a collection of nodes, the Get method returns an object of the RobotNode type. .

### I.1.2.2 IRobotCollection Fields

The fields of the IRobotCollection class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 10)	Number of all objects in a collection (it is also the index of the last element of the collection) .

### I.1.2.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of all objects in a collection (it is also the index of the last element of the collection) .

### I.1.2.3 IRobotCollection Methods

The methods of the IRobotCollection class are listed here.

## Public Methods

	Name	Description
◆	Get (see page 10)	The function returns an object with the indicated index in the collection. The type of returned object depends on the collection type - for a collection of nodes, the Get method returns an object of the RobotNode type. .

### I.1.2.3.1 Get

#### C++

```
HRESULT Get(long _index, IDispatch* ret);
```

#### C#

```
public IDispatch Get(long _index);
```

#### Visual Basic

```
Public Function Get(_index As long) As IDispatch
```

## Description

The function returns an object with the indicated index in the collection. The type of returned object depends on the collection type - for a collection of nodes, the Get method returns an object of the RobotNode type. .

### I.1.3 IRobotMultiSelection

#### Class Hierarchy

#### C++

```
interface IRobotMultiSelection : IDispatch;
```

#### C#

```
public interface IRobotMultiSelection;
```

#### Visual Basic

```
Public Interface IRobotMultiSelection
```

#### Description

RobotMultiSelection interface describes a set of many selections of different types. Multi-selection may be useful to describe complex selections of objects of more than one type. .

#### I.1.3.1 IRobotMultiSelection Members

The following tables list the members exposed by IRobotMultiSelection.

#### Public Fields

	Name	Description
❖	CaseCmpnt (see page 11)	Selected number of a load case component (it refers only to code combinations); zero value indicates selection of all components (selection is not limited to a specific component).
❖	Modes (see page 12)	Mode selection is treated as a supplement to load case selection .

#### Public Methods

	Name	Description
❖	Exist (see page 12)	The function returns True (non-zero value) if the multi-selection contains the selection of objects of the indicated type. .
❖	Get (see page 13)	The function returns selection of objects of the indicated type .
❖	Set (see page 13)	The function sets the indicated selection as the selection of objects of a given type within the multi-selection .

#### I.1.3.2 IRobotMultiSelection Fields

The fields of the IRobotMultiSelection class are listed here.

#### Public Fields

	Name	Description
❖	CaseCmpnt (see page 11)	Selected number of a load case component (it refers only to code combinations); zero value indicates selection of all components (selection is not limited to a specific component).
❖	Modes (see page 12)	Mode selection is treated as a supplement to load case selection .

#### I.1.3.2.1 CaseCmpnt

#### C++

```
HRESULT get_CaseCmpnt(long* );
HRESULT put_CaseCmpnt(long);
```

**C#**

```
public long CaseCmpnt { get; set; }
```

**Visual Basic**

```
Public CaseCmpnt As long
```

**Description**

Selected number of a load case component (it refers only to code combinations); zero value indicates selection of all components (selection is not limited to a specific component).

**Version**

Available since version 3.5.

**I.1.3.2.2 Modes****C++**

```
HRESULT get_Modes(IRobotModeSelection**);
```

**C#**

```
public IRobotModeSelection Modes { get; }
```

**Visual Basic**

```
Public ReadOnly Modes As IRobotModeSelection
```

**Description**

Mode selection is treated as a supplement to load case selection .

**Version**

Available since version 3.5.

**I.1.3.3 IRobotMultiSelection Methods**

The methods of the IRobotMultiSelection class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Exist (see page 12)	The function returns True (non-zero value) if the multi-selection contains the selection of objects of the indicated type. .
💡	Get (see page 13)	The function returns selection of objects of the indicated type .
💡	Set (see page 13)	The function sets the indicated selection as the selection of objects of a given type within the multi-selection .

**I.1.3.3.1 Exist****C++**

```
HRESULT Exist(IRobotObjectType _obj_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist(IRobotObjectType _obj_type);
```

**Visual Basic**

```
Public Function Exist(_obj_type As IRobotObjectType) As Boolean
```

**Description**

The function returns True (non-zero value) if the multi-selection contains the selection of objects of the indicated type. .

### I.1.3.3.2 Get

**C++**

```
HRESULT Get(IRobotObjectType _obj_type, IRobotSelection** ret);
```

**C#**

```
public IRobotSelection Get(IRobotObjectType _obj_type);
```

**Visual Basic**

```
Public Function Get(_obj_type As IRobotObjectType) As IRobotSelection
```

**Description**

The function returns selection of objects of the indicated type .

### I.1.3.3.3 Set

**C++**

```
HRESULT Set(IRobotObjectType _obj_type, IRobotSelection* _sel);
```

**C#**

```
public void Set(IRobotObjectType _obj_type, IRobotSelection _sel);
```

**Visual Basic**

```
Public Sub Set(_obj_type As IRobotObjectType, ByRef _sel As IRobotSelection)
```

**Description**

The function sets the indicated selection as the selection of objects of a given type within the multi-selection .

## I.1.4 IRobotMultiCollection

**Class Hierarchy**

**C++**

```
interface IRobotMultiCollection : IDispatch;
```

**C#**

```
public interface IRobotMultiCollection;
```

**Visual Basic**

```
Public Interface IRobotMultiCollection
```

**Description**

The RobotMultiCollection groups many collections of objects of various types. .

### I.1.4.1 IRobotMultiCollection Members

The following tables list the members exposed by IRobotMultiCollection.

**Public Methods**

	Name	Description
	Exist (see page 14)	The function returns True (non-zero value) if there is a collection of objects of the indicated type within the multi-collection .

	Get (see page 14)	The function returns a collection of objects of a similar type, contained in the given multi-collection. Before selecting a collection, one should check by means of the Exist (see page 14) function if it has been defined within the framework of the multi-collection. .
	Set (see page 14)	The function sets the indicated collection as the collection of objects of the indicated type within the multi-collection .

### I.1.4.2 IRobotMultiCollection Methods

The methods of the IRobotMultiCollection class are listed here.

#### Public Methods

	Name	Description
	Exist (see page 14)	The function returns True (non-zero value) if there is a collection of objects of the indicated type within the multi-collection .
	Get (see page 14)	The function returns a collection of objects of a similar type, contained in the given multi-collection. Before selecting a collection, one should check by means of the Exist (see page 14) function if it has been defined within the framework of the multi-collection. .
	Set (see page 14)	The function sets the indicated collection as the collection of objects of the indicated type within the multi-collection .

#### I.1.4.2.1 Exist

##### C++

```
HRESULT Exist(IRobotObjectType _obj_type, VARIANT_BOOL* ret);
```

##### C#

```
public bool Exist(IRobotObjectType _obj_type);
```

##### Visual Basic

```
Public Function Exist(_obj_type As IRobotObjectType) As Boolean
```

##### Description

The function returns True (non-zero value) if there is a collection of objects of the indicated type within the multi-collection .

#### I.1.4.2.2 Get

##### C++

```
HRESULT Get(IRobotObjectType _obj_type, IRobotCollection** ret);
```

##### C#

```
public IRobotCollection Get(IRobotObjectType _obj_type);
```

##### Visual Basic

```
Public Function Get(_obj_type As IRobotObjectType) As IRobotCollection
```

##### Description

The function returns a collection of objects of a similar type, contained in the given multi-collection. Before selecting a collection, one should check by means of the Exist (see page 14) function if it has been defined within the framework of the multi-collection. .

#### I.1.4.2.3 Set

##### C++

```
HRESULT Set(IRobotObjectType _obj_type, IRobotCollection* _collection);
```

**C#**

```
public void Set(IRobotObjectType _obj_type, IRobotCollection _collection);
```

**Visual Basic**

```
Public Sub Set(_obj_type As IRobotObjectType, ByRef _collection As IRobotCollection)
```

**Description**

The function sets the indicated collection as the collection of objects of the indicated type within the multi-collection .

**I.1.5 IRobotModeSelection****Class Hierarchy****C++**

```
interface IRobotModeSelection : IDispatch;
```

**C#**

```
public interface IRobotModeSelection;
```

**Visual Basic**

```
Public Interface IRobotModeSelection
```

**Description**

Interface describing selection of a mode for a modal load case. .

**Version**

Available since version 3.5.

**I.1.5.1 IRobotModeSelection Members**

The following tables list the members exposed by IRobotModeSelection.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Combination (see page 15)	Selected type of a combination which includes all load case modes .
❖	Mode (see page 16)	Mode number - its interpretation depends on a chosen type of mode selection .
❖	Type (see page 16)	Type of mode selection.

**I.1.5.2 IRobotModeSelection Fields**

The fields of the IRobotModeSelection class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Combination (see page 15)	Selected type of a combination which includes all load case modes .
❖	Mode (see page 16)	Mode number - its interpretation depends on a chosen type of mode selection .
❖	Type (see page 16)	Type of mode selection.

**I.1.5.2.1 Combination****C++**

```
HRESULT get_Combination(IRobotModeCombinationType* );
HRESULT put_Combination(IRobotModeCombinationType);
```

**C#**

```
public IRobotModeCombinationType Combination { get; set; }
```

**Visual Basic**

```
Public Combination As IRobotModeCombinationType
```

**Description**

Selected type of a combination which includes all load case modes .

**Version**

Available since version 3.5.

**I.1.5.2.2 Mode****C++**

```
HRESULT get_Mode(long* );
HRESULT put_Mode(long);
```

**C#**

```
public long Mode { get; set; }
```

**Visual Basic**

```
Public Mode As long
```

**Description**

Mode number - its interpretation depends on a chosen type of mode selection .

**Version**

Available since version 3.5.

**I.1.5.2.3 Type****C++**

```
HRESULT get_Type(IRobotModeSelectionType* );
HRESULT put_Type(IRobotModeSelectionType);
```

**C#**

```
public IRobotModeSelectionType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotModeSelectionType
```

**Description**

Type of mode selection.

**Version**

Available since version 3.5.

**I.2 Labels - complex attributes of objects**

The notion of Label has been introduced to simplify the description of complex object attributes - structure components. A complex attribute is an attribute for the description of which a set of values is required, as opposed to a simple attribute described by one value. A support is an example of a complex attribute. In order to define a support, one should provide an entire set of values describing particular support parameters. On the other hand, the entire support is an attribute of a supported node. The set of all values describing particular quantities-values of a complex attribute becomes in a sense the value of this attribute. A label is used to name and identify such a complex value. It suffices to provide once the values of all

attribute parameters (e.g. a support) and associate a name with this set (a character string), to make it possible to use the name only instead of the entire set of values. Thus, in order to apply an appropriate support to a node, it suffices to provide the name of a support (the label associated with the set of values describing all support parameters) and it is not necessary to reinforce-introduce the values of particular support parameters. Since there are many different types of complex attributes, the notion of a complex attribute i.e. a label type (RobotLabelType) has been introduced. The name of a label is unique within the set of labels of the same type. .

## Enumerations

	Name	Description
	IRobotLabelType (see page 17)	A set of Label types available in the Data Server is defined. .
	IRobotPredefinedLabel (see page 28)	Identifiers of labels automatically generated by program.

## Interfaces

	Name	Description
	IRobotLabel (see page 18)	The RobotLabel interface describes the complex attribute referred to as Label in the Robot Object Model. The complex attribute is defined by an entire set of values and not by one value, as in the case of a simple attribute. The set of values is associated with a name - a character string. The set of values describing complex attribute parameters consists the data of the label, while the character string associated with the set consists the label name.
	IRobotLabelServer (see page 20)	All labels are managed (created, made accessible, deleted) by the label server. Irrespective of label type, a label must be defined by means of the server to be used in structure definition later on. .

## I.2.1 IRobotLabelType

### C++

```
enum IRobotLabelType;
```

### C#

```
public enum IRobotLabelType;
```

### Visual Basic

```
Public Enum IRobotLabelType
```

### Members

Members	Description
I_LT_NODE_RELEASE = 1	Label describing a node release .
I_LT_NODE_COMPATIBILITY = 2	Label describing a compatibility .
I_LT_BAR_SECTION = 3	Label describing a bar section .
I_LT_BAR_RELEASE = 4	Label describing a bar release .
I_LT_BAR_OFFSET = 5	Label describing a bar offset .
I_LT_BAR_CABLE = 6	Label describing a cable .
I_LT_BAREND_BRACKET = 7	Label describing a bracket at one of bar ends.
I_LT_PANEL_THICKNESS = 11	Label describing a plate thickness.
I_LT_PANEL_REINFORCEMENT = 12	Label describing a plate reinforcement.
I_LT_UNKNOWN = -1	Unknown label type Available since version 1.7.
I_LT_SUPPORT = 0	Label describing a support Available since version 2.0.
I_LT_MATERIAL = 8	Label describing a material Available since version 2.0.
I_LT_LINEAR_RELEASE = 13	Linear release between panels. Available since version 2.5.

I_LT_BAR_ELASTIC_GROUND = 14	Elastic ground (bars). Available since version 2.5.
I_LT_NODE_RIGID_LINK = 15	Rigid link. Available since version 2.5.
I_LT_MEMBER_TYPE = 16	Member type for steel/aluminum and timber design. Available since version 2.5.
I_LT_VEHICLE = 17	Vehicle definition. Available since version 3.
I_LT_SOLID_PROPERTIES = 18	Solid properties. Available since version 3.
I_LT_BAR_GEO_IMPERFECTIONS = 19	Geometrical imperfections. Available since version 3.
I_LT_BAR_NONLINEAR_HINGE = 20	Non-linear hinge. Available since version 3.
I_LT_CLADDING = 21	Available since version 3.5.
I_LT_PANEL_CALC_MODEL = 22	Label describing a panel calculation model. Available since version 9.7.
I_LT_MEMBER_REINFORCEMENT_PARAMS = 25	Available since version 12.

**Description**

A set of Label types available in the Data Server is defined.

**I.2.2 IRobotLabel****Class Hierarchy****C++**

```
interface IRobotLabel : IDispatch;
```

**C#**

```
public interface IRobotLabel;
```

**Visual Basic**

```
Public Interface IRobotLabel
```

**Description**

The RobotLabel interface describes the complex attribute referred to as Label in the Robot Object Model. The complex attribute is defined by an entire set of values and not by one value, as in the case of a simple attribute. The set of values is associated with a name - a character string. The set of values describing complex attribute parameters consists the data of the label, while the character string associated with the set consists the label name.

**I.2.2.1 IRobotLabel Members**

The following tables list the members exposed by IRobotLabel.

**Public Fields**

	Name	Description
◆	Data (see page 19)	Object containing the label data - set of values describing the complex attribute represented by the label. The object type depends on the label type. For instance, the data of a label describing a node support (i.e. labels of the I_LT_NODE_SUPPORT type) are represented by the object of the RobotNodeSupportData type..
◆	Name (see page 19)	Name associated with the label - the name us unique among all the labels of the same type and therefore it may be used to identify the label. .
◆	Type (see page 19)	Label type .

	Uniqueld ( <a href="#">see page 20</a> )	Unique label identifier Available since version 1.7.
-----------------------------------------------------------------------------------	------------------------------------------	------------------------------------------------------

### I.2.2.2 IRobotLabel Fields

The fields of the IRobotLabel class are listed here.

#### Public Fields

	Name	Description
	Data ( <a href="#">see page 19</a> )	Object containing the label data - set of values describing the complex attribute represented by the label. The object type depends on the label type. For instance, the data of a label describing a node support (i.e. labels of the I_LT_NODE_SUPPORT type) are represented by the object of the RobotNodeSupportData type. .
	Name ( <a href="#">see page 19</a> )	Name associated with the label - the name us unique among all the labels of the same type and therefore it may be used to identify the label. .
	Type ( <a href="#">see page 19</a> )	Label type .
	Uniqueld ( <a href="#">see page 20</a> )	Unique label identifier Available since version 1.7.

#### I.2.2.2.1 Data

##### C++

```
HRESULT get_Data(IDispatch*);
```

##### C#

```
public IDispatch Data { get; }
```

##### Visual Basic

```
Public ReadOnly Data As IDispatch
```

##### Description

Object containing the label data - set of values describing the complex attribute represented by the label. The object type depends on the label type. For instance, the data of a label describing a node support (i.e. labels of the I\_LT\_NODE\_SUPPORT type) are represented by the object of the RobotNodeSupportData type. .

#### I.2.2.2.2 Name

##### C++

```
HRESULT get_Name(BSTR*);
```

##### C#

```
public String Name { get; }
```

##### Visual Basic

```
Public ReadOnly Name As String
```

##### Description

Name associated with the label - the name us unique among all the labels of the same type and therefore it may be used to identify the label. .

#### I.2.2.2.3 Type

##### C++

```
HRESULT get_Type(IRobotLabelTextType*);
```

##### C#

```
public IRobotLabelTextType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotLabelType
```

**Description**

Label type .

**I.2.2.2.4 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique label identifier Available since version 1.7.

**I.2.3 IRobotLabelServer****Class Hierarchy****C++**

```
interface IRobotLabelServer : IDispatch;
```

**C#**

```
public interface IRobotLabelServer;
```

**Visual Basic**

```
Public Interface IRobotLabelServer
```

**Description**

All labels are managed (created, made accessible, deleted) by the label server. Irrespective of label type, a label must be defined by means of the server to be used in structure definition later on. .

**I.2.3.1 IRobotLabelServer Members**

The following tables list the members exposed by IRobotLabelServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✿	Create ( <a href="#">see page 23</a> )	The function creates and returns a new label of the defined type and with the defined name. if labels of the indicated type are stored in a database (as in the case foundation bar section type labels - I_LT_BAR_SECTION), then, the function Create looks through the currently available databases to find a label with the defined name. If the label is not found in the database, all parameter values describing a label will be initialized with appropriate values from the database. If the user defines a label that is absent from the database, then, he should define appropriate values of... more ( <a href="#">see page 23</a> )
✿	CreateLike ( <a href="#">see page 23</a> )	Function creates and returns a new label of the indicated type with the indicated name. A data set defining the label will be copied from the label named the "_like_name" parameter.

	Delete ( <a href="#">see page 23</a> )	The function deletes a label of the indicated type and name. If the label has been used as an attribute value of a structure component, then, the value of the attribute is set as undefined after deleting the label. For instance, if a bar had STEEL defined as its material, then, after deleting the STEEL label of the I_LT_BAR_MATERIAL type from the label server, the bar will have no material defined. .
	Exist ( <a href="#">see page 24</a> )	The function checks if there exist the label of the indicated type and name, i.e. if it has been saved in the structure. .
	FindWithId ( <a href="#">see page 24</a> )	Function finds and returns a label with the specified identifier. If such a label is not found, then an "empty reference" is returned (Nothing in the languages such as Visual Basic, NULL in the languages such as C++). Available since version 1.7.
	Get ( <a href="#">see page 24</a> )	The function returns the label of the indicated type and name. If such label already exists, running the function leads to critical error. .
	GetAll ( <a href="#">see page 24</a> )	The function returns all labels of all types that have been saved in the structure. .
	GetAvailableNames ( <a href="#">see page 25</a> )	Function returns the table with names of available labels of a given type. The available labels are those that may be assigned to structure elements by means of the SetLabel function. They include all the labels used earlier in the current structure and global labels - available in each structure.
	GetDefault ( <a href="#">see page 25</a> )	Function returns the name of a default label of the specified type.
	GetMany ( <a href="#">see page 25</a> )	The function returns a collection containing all label of the indicated type saved in the structure .
	GetPredefinedName ( <a href="#">see page 26</a> )	Function returns the current name of the given predefined label. .
	GetUniqueId ( <a href="#">see page 26</a> )	Function returns a unique identifier for the specified label.
	IsAvailable ( <a href="#">see page 26</a> )	Function returns a non-zero value (True) if the indicated label is available - it may be assigned to a structure element by means of the SetLabel function..
	IsPredefinedName ( <a href="#">see page 27</a> )	Function returns True if the given type and label name correspond with one of the predefined label. The identifier of the found predefined label will be added to the initial parameter _ret_predef_lab.
	IsUsed ( <a href="#">see page 27</a> )	Function checks if the label of the specified type with the specified name is currently used in the structure.
	SetDefault ( <a href="#">see page 27</a> )	Function sets a name of a default label of the specified type. Default labels are applied during automatic generation of structure elements e.g. as a result of edit operations performed on a structure.
	Store ( <a href="#">see page 28</a> )	Once a label is created and the values of its particular parameters are defined, one should save it in the structure so that it could be applied to appropriate structure component. If a label of the same type and name has already been saved in the structure, the values of all its parameters will be up-dated on the basis of the newly-defined values. .
	StoreWithName ( <a href="#">see page 28</a> )	Function saves the indicated label under the indicated name.

### I.2.3.2 IRobotLabelServer Methods

The methods of the IRobotLabelServer class are listed here.

## Public Methods

	<b>Name</b>	<b>Description</b>
≡	Create ( <a href="#">see page 23</a> )	The function creates and returns a new label of the defined type and with the defined name. If labels of the indicated type are stored in a database (as in the case foundation bar section type labels - I_LT_BAR_SECTION), then, the function Create looks through the currently available databases to find a label with the defined name. If the label is not found in the database, all parameter values describing a label will be initialized with appropriate values from the database. If the user defines a label that is absent from the database, then, he should define appropriate values of... more ( <a href="#">see page 23</a> )
≡	CreateLike ( <a href="#">see page 23</a> )	Function creates and returns a new label of the indicated type with the indicated name. A data set defining the label will be copied from the label named the "_like_name" parameter.
≡	Delete ( <a href="#">see page 23</a> )	The function deletes a label of the indicated type and name. If the label has been used as an attribute value of a structure component, then, the value of the attribute is set as undefined after deleting the label. For instance, if a bar had STEEL defined as its material, then, after deleting the STEEL label of the I_LT_BAR_MATERIAL type from the label server, the bar will have no material defined. .
≡	Exist ( <a href="#">see page 24</a> )	The function checks if there exist the label of the indicated type and name, i.e. if it has been saved in the structure. .
≡	FindWithId ( <a href="#">see page 24</a> )	Function finds and returns a label with the specified identifier. If such a label is not found, then an "empty reference" is returned (Nothing in the languages such as Visual Basic, NULL in the languages such as C++). Available since version 1.7.
≡	Get ( <a href="#">see page 24</a> )	The function returns the label of the indicated type and name. If such label already exists, running the function leads to critical error. .
≡	GetAll ( <a href="#">see page 24</a> )	The function returns all labels of all types that have been saved in the structure. .
≡	GetAvailableNames ( <a href="#">see page 25</a> )	Function returns the table with names of available labels of a given type. The available labels are those that may be assigned to structure elements by means of the SetLabel function. They include all the labels used earlier in the current structure and global labels - available in each structure.
≡	GetDefault ( <a href="#">see page 25</a> )	Function returns the name of a default label of the specified type.
≡	GetMany ( <a href="#">see page 25</a> )	The function returns a collection containing all label of the indicated type saved in the structure. .
≡	GetPredefinedName ( <a href="#">see page 26</a> )	Function returns the current name of the given predefined label. .
≡	GetUniqueId ( <a href="#">see page 26</a> )	Function returns a unique identifier for the specified label.
≡	IsAvailable ( <a href="#">see page 26</a> )	Function returns a non-zero value (True) if the indicated label is available - it may be assigned to a structure element by means of the SetLabel function. .
≡	IsPredefinedName ( <a href="#">see page 27</a> )	Function returns True if the given type and label name correspond with one of the predefined label. The identifier of the found predefined label will be added to the initial parameter _ret_predef_lab.
≡	IsUsed ( <a href="#">see page 27</a> )	Function checks if the label of the specified type with the specified name is currently used in the structure.
≡	SetDefault ( <a href="#">see page 27</a> )	Function sets a name of a default label of the specified type. Default labels are applied during automatic generation of structure elements e.g. as a result of edit operations performed on a structure.
≡	Store ( <a href="#">see page 28</a> )	Once a label is created and the values of its particular parameters are defined, one should save it in the structure so that it could be applied to appropriate structure component. If a label of the same type and name has already been saved in the structure, the values of all its parameters will be up-dated on the basis of the newly-defined values. .
≡	StoreWithName ( <a href="#">see page 28</a> )	Function saves the indicated label under the indicated name.

### I.2.3.2.1 Create

#### C++

```
HRESULT Create(IRobotLabelType _label_type, BSTR _label_name, IRobotLabel** ret);
```

#### C#

```
public IRobotLabel Create(IRobotLabelType _label_type, String _label_name);
```

#### Visual Basic

```
Public Function Create(_label_type As IRobotLabelType, _label_name As String) As IRobotLabel
```

#### Description

The function creates and returns a new label of the defined type and with the defined name. If labels of the indicated type are stored in a database (as in the case foundation bar section type labels - I\_LT\_BAR\_SECTION), then, the function Create looks through the currently available databases to find a label with the defined name. If the label is not found in the database, all parameter values describing a label will be initialized with appropriate values from the database. If the user defines a label that is absent from the database, then, he should define appropriate values of all the parameters describing the complex attribute. .

### I.2.3.2.2 CreateLike

#### C++

```
HRESULT CreateLike(IRobotLabelType _lab_type, BSTR _lab_name, BSTR _like_name,
IRobotLabel** ret);
```

#### C#

```
public IRobotLabel CreateLike(IRobotLabelType _lab_type, String _lab_name, String
_like_name);
```

#### Visual Basic

```
Public Function CreateLike(_lab_type As IRobotLabelType, _lab_name As String, _like_name As
String) As IRobotLabel
```

#### Description

Function creates and returns a new label of the indicated type with the indicated name. A data set defining the label will be copied from the label named the "\_like\_name" parameter.

#### Version

Available since version 3.

### I.2.3.2.3 Delete

#### C++

```
HRESULT Delete(IRobotLabelType _label_type, BSTR _label_name);
```

#### C#

```
public void Delete(IRobotLabelType _label_type, String _label_name);
```

#### Visual Basic

```
Public Sub Delete(_label_type As IRobotLabelType, _label_name As String)
```

#### Description

The function deletes a label of the indicated type and name. If the label has been used as an attribute value of a structure component, then, the value of the attribute is set as undefined after deleting the label. For instance, if a bar had STEEL defined as its material, then, after deleting the STEEL label of the I\_LT\_BAR\_MATERIAL type from the label server, the bar

will have no material defined. .

#### I.2.3.2.4 Exist

##### C++

```
HRESULT Exist(IRobotLabelType _label_type, BSTR _label_name, VARIANT_BOOL* ret);
```

##### C#

```
public bool Exist(IRobotLabelType _label_type, String _label_name);
```

##### Visual Basic

```
Public Function Exist(_label_type As IRobotLabelType, _label_name As String) As Boolean
```

##### Description

The function checks if there exist the label of the indicated type and name, i.e. if it has been saved in the structure. .

#### I.2.3.2.5 FindWithId

##### C++

```
HRESULT FindWithId(long _unique_id, IRobotLabel** ret);
```

##### C#

```
public IRobotLabel FindWithId(long _unique_id);
```

##### Visual Basic

```
Public Function FindWithId(_unique_id As long) As IRobotLabel
```

##### Description

Function finds and returns a label with the specified identifier. If such a label is not found, then an "empty reference" is returned (Nothing in the languages such as Visual Basic, NULL in the languages such as C++). Available since version 1.7.

#### I.2.3.2.6 Get

##### C++

```
HRESULT Get(IRobotLabelType _label_type, BSTR _label_name, IRobotLabel** ret);
```

##### C#

```
public IRobotLabel Get(IRobotLabelType _label_type, String _label_name);
```

##### Visual Basic

```
Public Function Get(_label_type As IRobotLabelType, _label_name As String) As IRobotLabel
```

##### Description

The function returns the label of the indicated type and name. If such label already exists, running the function leads to critical error. .

#### I.2.3.2.7 GetAll

##### C++

```
HRESULT GetAll(IRobotCollection** ret);
```

##### C#

```
public IRobotCollection GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotCollection
```

**Description**

The function returns all labels of all types that have been saved in the structure. .

**I.2.3.2.8 GetAvailableNames****C++**

```
HRESULT GetAvailableNames(IRobotLabelType _label_type, IRobotNamesArray** ret);
```

**C#**

```
public IRobotNamesArray GetAvailableNames(IRobotLabelType _label_type);
```

**Visual Basic**

```
Public Function GetAvailableNames(_label_type As IRobotLabelType) As IRobotNamesArray
```

**Description**

Function returns the table with names of available labels of a given type. The available labels are those that may be assigned to structure elements by means of the SetLabel function. They include all the labels used earlier in the current structure and global labels - available in each structure.

**Version**

Available since version 3.

**I.2.3.2.9 GetDefault****C++**

```
HRESULT GetDefault(IRobotLabelType _lab_type, BSTR* ret);
```

**C#**

```
public String GetDefault(IRobotLabelType _lab_type);
```

**Visual Basic**

```
Public Function GetDefault(_lab_type As IRobotLabelType) As String
```

**Description**

Function returns the name of a default label of the specified type.

**Version**

Available since version 3.5.

**I.2.3.2.10 GetMany****C++**

```
HRESULT GetMany(IRobotLabelType _label_type, IRobotCollection** ret);
```

**C#**

```
public IRobotCollection GetMany(IRobotLabelType _label_type);
```

**Visual Basic**

```
Public Function GetMany(_label_type As IRobotLabelType) As IRobotCollection
```

## Description

The function returns a collection containing all label of the indicated type saved in the structure .

### I.2.3.2.11 GetPredefinedName

#### C++

```
HRESULT GetPredefinedName(IRobotPredefinedLabel _predef_label, BSTR* ret);
```

#### C#

```
public String GetPredefinedName(IRobotPredefinedLabel _predef_label);
```

#### Visual Basic

```
Public Function GetPredefinedName(_predef_label As IRobotPredefinedLabel) As String
```

#### Description

Function returns the current name of the given predefined label. .

#### Version

Available since version 11.

### I.2.3.2.12 GetUniqueId

#### C++

```
HRESULT GetUniqueId(IRobotLabelType _lab_type, BSTR _lab_name, long* ret);
```

#### C#

```
public long GetUniqueId(IRobotLabelType _lab_type, String _lab_name);
```

#### Visual Basic

```
Public Function GetUniqueId(_lab_type As IRobotLabelType, _lab_name As String) As long
```

#### Description

Function returns a unique identifier for the specified label.

#### Version

Available since version 8.2.

### I.2.3.2.13 IsAvailable

#### C++

```
HRESULT IsAvailable(IRobotLabelType _lab_type, BSTR _lab_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsAvailable(IRobotLabelType _lab_type, String _lab_name);
```

#### Visual Basic

```
Public Function IsAvailable(_lab_type As IRobotLabelType, _lab_name As String) As Boolean
```

#### Description

Function returns a non-zero value (True) if the indicated label is available - it may be assigned to a structure element by means of the SetLabel function. .

#### Version

Available since version 3.

### I.2.3.2.14 IsPredefinedName

#### C++

```
HRESULT IsPredefinedName(IRobotLabelType _lab_type, BSTR _lab_name, RobotPredefinedLabel* _ret_predef_lab, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsPredefinedName(IRobotLabelType _lab_type, String _lab_name, RobotPredefinedLabel* _ret_predef_lab);
```

#### Visual Basic

```
Public Function IsPredefinedName(_lab_type As IRobotLabelType, _lab_name As String, ByRef _ret_predef_lab As RobotPredefinedLabel*) As Boolean
```

#### Description

Function returns True if the given type and labe name correspond with one of the predefined label. The identifier of the found predefined lable will be added to the initial parameter \_ret\_predef\_lab.

#### Version

Available since version 11.

### I.2.3.2.15 IsUsed

#### C++

```
HRESULT IsUsed(IRobotLabelType _lab_type, BSTR _lab_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsUsed(IRobotLabelType _lab_type, String _lab_name);
```

#### Visual Basic

```
Public Function IsUsed(_lab_type As IRobotLabelType, _lab_name As String) As Boolean
```

#### Description

Function checks if the label of the specified type with the specified name is currently used in th structure.

#### Version

Available since version 8.1.

### I.2.3.2.16 SetDefault

#### C++

```
HRESULT SetDefault(IRobotLabelType _lab_type, BSTR _lab_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool SetDefault(IRobotLabelType _lab_type, String _lab_name);
```

#### Visual Basic

```
Public Function SetDefault(_lab_type As IRobotLabelType, _lab_name As String) As Boolean
```

#### Description

Function sets a name of a default label of the specified type. Default labels are applied during automatic generation of structure elements e.g. as a result of edit operations performed on a structure.

#### Version

Available since version 3.5.

### I.2.3.2.17 Store

#### C++

```
HRESULT Store(IRobotLabel* _label);
```

#### C#

```
public void Store(IRobotLabel _label);
```

#### Visual Basic

```
Public Sub Store(ByRef _label As IRobotLabel)
```

#### Description

Once a label is created and the values of its particular parameters are defined, one should save it in the structure so that it could be applied to appropriate structure component. If a label of the same type and name has already been saved in the structure, the values of all its parameters will be up-dated on the basis of the newly-defined values. .

### I.2.3.2.18 StoreWithName

#### C++

```
HRESULT StoreWithName(IRobotLabel* _label, BSTR _label_name);
```

#### C#

```
public void StoreWithName(IRobotLabel _label, String _label_name);
```

#### Visual Basic

```
Public Sub StoreWithName(ByRef _label As IRobotLabel, _label_name As String)
```

#### Description

Function saves the indicated label under the indicated name.

#### Version

Available since version 3.

## I.2.4 IRobotPredefinedLabel

#### C++

```
enum IRobotPredefinedLabel;
```

#### C#

```
public enum IRobotPredefinedLabel;
```

#### Visual Basic

```
Public Enum IRobotPredefinedLabel
```

#### Members

Members	Description
I_PL_MEMBER_TYPE_TIMBER_MEMBER = 1	Available since version 11.
I_PL_MEMBER_TYPE_TIMBER_COLUMN = 2	Available since version 11.
I_PL_MEMBER_TYPE_TIMBER_BEAM = 3	Available since version 11.
I_PL_MEMBER_TYPE_SIMPLE_BAR = 4	Available since version 11.
I_PL_MEMBER_TYPE_COLUMN = 5	Available since version 11.
I_PL_MEMBER_TYPE_BEAM = 6	Available since version 11.
I_PL_PANEL_CALC_MODEL_SHELL = 11	Available since version 11.

I_PL_PANEL_CALC_MODEL_SLAB_FLEXIBLE_DIAPHRAGM = 12	Available since version 11.
I_PL_PANEL_CALC_MODEL_SLAB_RIGID_DIAPHRAGM = 13	Available since version 11.
I_PL_PANEL_CALC_MODEL_CURTAIN_WALL = 14	Available since version 11.
I_PL_PANEL_CALC_MODEL_DECK_SLAB = 15	Available since version 11.
I_PL_PANEL_REINFORCEMENT_RC_FLOOR = 21	Available since version 11.
I_PL_PANEL_REINFORCEMENT_RC_WALL = 22	Available since version 11.
I_PL_PANEL_REINFORCEMENT_RC_SHELL = 23	Available since version 11.
I_PL_MEMBER_TYPE_RC_BEAM = 7	Available since version 11.
I_PL_MEMBER_TYPE_RC_COLUMN = 8	Available since version 11.
I_PL_MEMBER_REINFORCEMENT_PARAMS_STANDARD = 31	Available since version 12.
I_PL_PANEL_CALC_MODEL_SLAB_XY_DIAPHRAGM = 16	Available since version 12.

**Description**

Identifiers of labels automatically generated by program.

**Version**

Available since version 11.

## I.3 Uniform access to structure components

In order to unify access mode to different structure components, the following two notions have been defined: Object (RobotDataObject) and Object Server (RobotDataObjectServer). An Object represents any structure component. The Object Server manages all objects of the same type. For instance, a structure node is represented in the Robot Object Model as an object of the RobotNode type, and all nodes are managed (created, made accessible, deleted) by the RobotNodeServer type. .

**Interfaces**

	Name	Description
»	IRobotDataObject ( <a href="#">see page 29</a> )	RobotDataObject defines a common interface for structure components. Each structure component has its own user-defined number that is unique among all the components of the same type. The user may use the number to identify objects. There is one uniform manner of ascribing and accessing complex attributes (labels) defined for all structure components. .
»	IRobotDataObjectServer ( <a href="#">see page 33</a> )	All objects of the same type are managed by the same object server. RobotDataObjectServer defines a common interface for such servers. .

### I.3.1 IRobotDataObject

**Class Hierarchy****C++**

```
interface IRobotDataObject : IDispatch;
```

**C#**

```
public interface IRobotDataObject;
```

**Visual Basic**

```
Public Interface IRobotDataObject
```

**Description**

RobotDataObject defines a common interface for structure components. Each structure component has its own user-defined

number that is unique among all the components of the same type. The user may use the number to identify objects. There is one uniform manner of ascribing and accessing complex attributes (labels) defined for all structure components. .

### I.3.1.1 IRobotDataObject Members

The following tables list the members exposed by IRobotDataObject.

#### Public Fields

	Name	Description
◆	Number (see page 30)	Object number is assigned by the user; it is unique among all the components of the same type..

#### Public Methods

	Name	Description
◆	GetLabel (see page 31)	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabelName (see page 31)	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabels (see page 31)	The function returns a collection containing all labels (of different types) defined for a given object. .
◆	HasLabel (see page 32)	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
◆	RemoveLabel (see page 32)	The function deletes a label of the indicated type. .
◆	SetLabel (see page 32)	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .

### I.3.1.2 IRobotDataObject Fields

The fields of the IRobotDataObject class are listed here.

#### Public Fields

	Name	Description
◆	Number (see page 30)	Object number is assigned by the user; it is unique among all the components of the same type..

#### I.3.1.2.1 Number

##### C++

```
HRESULT get_Number(long* );
HRESULT put_Number(long );
```

##### C#

```
public long Number { get; set; }
```

##### Visual Basic

```
Public Number As long
```

##### Description

Object number is assigned by the user; it is unique among all the components of the same type. .

### I.3.1.3 IRobotDataObject Methods

The methods of the IRobotDataObject class are listed here.

## Public Methods

	Name	Description
ESHOW	GetLabel (see page 31)	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
ESHOW	GetLabelName (see page 31)	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
ESHOW	GetLabels (see page 31)	The function returns a collection containing all labels (of different types) defined for a given object. .
ESHOW	HasLabel (see page 32)	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
ESHOW	RemoveLabel (see page 32)	The function deletes a label of the indicated type. .
ESHOW	SetLabel (see page 32)	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .

### I.3.1.3.1 GetLabel

#### C++

```
HRESULT GetLabel(IRobotLabelType _label_type, IRobotLabel** ret);
```

#### C#

```
public IRobotLabel GetLabel(IRobotLabelType _label_type);
```

#### Visual Basic

```
Public Function GetLabel(_label_type As IRobotLabelType) As IRobotLabel
```

#### Description

The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .

### I.3.1.3.2 GetLabelName

#### C++

```
HRESULT GetLabelName(IRobotLabelType _label_type, BSTR* ret);
```

#### C#

```
public String GetLabelName(IRobotLabelType _label_type);
```

#### Visual Basic

```
Public Function GetLabelName(_label_type As IRobotLabelType) As String
```

#### Description

The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .

### I.3.1.3.3 GetLabels

#### C++

```
HRESULT GetLabels(IRobotCollection** ret);
```

**C#**

```
public IRobotCollection GetLabels();
```

**Visual Basic**

```
Public Function GetLabels() As IRobotCollection
```

**Description**

The function returns a collection containing all labels (of different types) defined for a given object. .

**I.3.1.3.4 HasLabel****C++**

```
HRESULT HasLabel(IRobotLabelType _label_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool HasLabel(IRobotLabelType _label_type);
```

**Visual Basic**

```
Public Function HasLabel(_label_type As IRobotLabelType) As Boolean
```

**Description**

The function returns True (non-zero value) if a label of the type has already been defined for the object. .

**I.3.1.3.5 RemoveLabel****C++**

```
HRESULT RemoveLabel(IRobotLabelType _label_type);
```

**C#**

```
public void RemoveLabel(IRobotLabelType _label_type);
```

**Visual Basic**

```
Public Sub RemoveLabel(_label_type As IRobotLabelType)
```

**Description**

The function deletes a label of the indicated type. .

**I.3.1.3.6 SetLabel****C++**

```
HRESULT SetLabel(IRobotLabelType _label_type, BSTR _label_name);
```

**C#**

```
public void SetLabel(IRobotLabelType _label_type, String _label_name);
```

**Visual Basic**

```
Public Sub SetLabel(_label_type As IRobotLabelType, _label_name As String)
```

**Description**

The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .

## I.3.2 IRobotDataObjectServer

### Class Hierarchy

#### C++

```
interface IRobotDataObjectServer : IDispatch;
```

#### C#

```
public interface IRobotDataObjectServer;
```

#### Visual Basic

```
Public Interface IRobotDataObjectServer
```

### Description

All objects of the same type are managed by the same object server. RobotDataObjectServer defines a common interface for such servers. .

### I.3.2.1 IRobotDataObjectServer Members

The following tables list the members exposed by IRobotDataObjectServer.

#### Public Methods

	Name	Description
≡	Delete ( [ see page 34) )	The function deletes the object of the indicated number. .
≡	DeleteMany ( [ see page 34) )	The function deletes all objects that meet the criteria of the indicated selection. .
≡	Exist ( [ see page 34) )	The function returns True (non-zero value) if the object of the indicated name already exists. .
≡	Get ( [ see page 34) )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .
≡	GetAll ( [ see page 35) )	The function returns a collection containing all objects managed by the server. .
≡	GetMany ( [ see page 35) )	The function returns a collection of objects that meet the criteria of the indicated selection. .
≡	RemoveLabel ( [ see page 35) )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .
≡	SetLabel ( [ see page 36) )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. .

### I.3.2.2 IRobotDataObjectServer Methods

The methods of the IRobotDataObjectServer class are listed here.

#### Public Methods

	Name	Description
≡	Delete ( [ see page 34) )	The function deletes the object of the indicated number. .
≡	DeleteMany ( [ see page 34) )	The function deletes all objects that meet the criteria of the indicated selection. .
≡	Exist ( [ see page 34) )	The function returns True (non-zero value) if the object of the indicated name already exists. .
≡	Get ( [ see page 34) )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .

	GetAll ( <a href="#">see page 35</a> )	The function returns a collection containing all objects managed by the server. .
	GetMany ( <a href="#">see page 35</a> )	The function returns a collection of objects that meet the criteria of the indicated selection. .
	RemoveLabel ( <a href="#">see page 35</a> )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .
	SetLabel ( <a href="#">see page 36</a> )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. .

### I.3.2.2.1 Delete

**C++**

```
HRESULT Delete(long _number);
```

**C#**

```
public void Delete(long _number);
```

**Visual Basic**

```
Public Sub Delete(_number As long)
```

**Description**

The function deletes the object of the indicated number. .

### I.3.2.2.2 DeleteMany

**C++**

```
HRESULT DeleteMany(IRobotSelection* _selection);
```

**C#**

```
public void DeleteMany(IRobotSelection _selection);
```

**Visual Basic**

```
Public Sub DeleteMany(ByRef _selection As IRobotSelection)
```

**Description**

The function deletes all objects that meet the criteria of the indicated selection. .

### I.3.2.2.3 Exist

**C++**

```
HRESULT Exist(long _number, VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist(long _number);
```

**Visual Basic**

```
Public Function Exist(_number As long) As Boolean
```

**Description**

The function returns True (non-zero value) if the object of the indicated name already exists. .

### I.3.2.2.4 Get

**C++**

```
HRESULT Get(long _number, IRobotDataObject** ret);
```

**C#**

```
public IRobotDataObject Get(long _number);
```

**Visual Basic**

```
Public Function Get(_number As long) As IRobotDataObject
```

**Description**

The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .

**I.3.2.2.5 GetAll****C++**

```
HRESULT GetAll(IRobotCollection** ret);
```

**C#**

```
public IRobotCollection GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotCollection
```

**Description**

The function returns a collection containing all objects managed by the server. .

**I.3.2.2.6 GetMany****C++**

```
HRESULT GetMany(IRobotSelection* _selection, IRobotCollection** ret);
```

**C#**

```
public IRobotCollection GetMany(IRobotSelection _selection);
```

**Visual Basic**

```
Public Function GetMany(ByRef _selection As IRobotSelection) As IRobotCollection
```

**Description**

The function returns a collection of objects that meet the criteria of the indicated selection. .

**I.3.2.2.7 RemoveLabel****C++**

```
HRESULT RemoveLabel(IRobotSelection* _object_selection, IRobotLabelType _label_type);
```

**C#**

```
public void RemoveLabel(IRobotSelection _object_selection, IRobotLabelType _label_type);
```

**Visual Basic**

```
Public Sub RemoveLabel(ByRef _object_selection As IRobotSelection, _label_type As IRobotLabelType)
```

**Description**

The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .

### I.3.2.2.8 SetLabel

#### C++

```
HRESULT SetLabel(IRobotSelection* _object_selection, IRobotLabelType _label_type, BSTR _label_name);
```

#### C#

```
public void SetLabel(IRobotSelection _object_selection, IRobotLabelType _label_type, String _label_name);
```

#### Visual Basic

```
Public Sub SetLabel(ByRef _object_selection As IRobotSelection, _label_type As IRobotLabelType, _label_name As String)
```

#### Description

The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection.

## I.4 IRobotObjectType

#### C++

```
enum IRobotObjectType;
```

#### C#

```
public enum IRobotObjectType;
```

#### Visual Basic

```
Public Enum IRobotObjectType
```

#### Members

Members	Description
I_OT_NODE = 0	Node .
I_OT_BAR = 1	Bar .
I_OT_PANEL = 4	Panel.
I_OTFINITE_ELEMENT = 5	Finite element.
I_OT_GEOMETRY = 6	Geometrical object .
I_OT_FAMILY = 3	Group.
I_OT_CASE = 2	Load case .
I_OT_OBJECT = -2	Shared auxiliary identifier for the following types of structure components: bar, slab, volumetric object and geometric object Available since version 2.0.
I_OT_UNDEFINED = -1	Auxiliary identifier denoting undefined or unknown object type Available since version 2.0.
I_OT_VOLUME = 7	Volumetric object. Available since version 3.

#### Description

RobotObjectType defines identifiers for all types of objects managed by the Data Server .

## II Structure components

All structure components are represented by the objects of relevant data types. .

## Enumerations

	Name	Description
	IRobotObjectStructuralType ( <a href="#">see page 846</a> )	Type of object as a structure component.

## II.1 Load cases

Robot Object Model makes it possible to perform operations on load cases available in the Robot program. There are two basic types of load cases: a simple case and combinations. A simple case definition consists of a list of load records, while a combination contains a list of simple cases with the relevant coefficients. .

## Enumerations

	Name	Description
	IRobotCaseNature ( <a href="#">see page 406</a> )	The set of case natures accepted by Robot is determined. .
	IRobotCaseAnalyzeType ( <a href="#">see page 407</a> )	Robot Object Model defines a set of analysis types for load cases accepted by Robot. .
	IRobotCombinationType ( <a href="#">see page 429</a> )	Type of load case combinations available in Robot. .
	IRobotCaseType ( <a href="#">see page 429</a> )	Among load cases, one distinguishes three basic types: a simple case, a combination and a code combination. .
	IRobotLimitState ( <a href="#">see page 431</a> )	Limit state type.
	IRobotCaseAnalysisModesFilterType ( <a href="#">see page 433</a> )	Available methods of mode filtering.
	IRobotDynamicAnalysisDampingType ( <a href="#">see page 442</a> )	
	IRobotCaseRelatedValueType ( <a href="#">see page 442</a> )	A set of identifiers for types of values that can be associated with the load case.
	IRobotCasePredefinedNumber ( <a href="#">see page 474</a> )	A list of predefined values which can be used when calling method Get of RobotCaseServer in order to obtain a specific RobotCase object.

## Interfaces

	Name	Description
	IRobotCase ( <a href="#">see page 404</a> )	The part of the load case interface that is shared by simple cases and combinations has been defined using RobotCase. .
	IRobotCaseServer ( <a href="#">see page 407</a> )	Server of load cases .
	IRobotSimpleCase ( <a href="#">see page 417</a> )	A simple load case is defined using the list of load records. .
	IRobotCaseCombination ( <a href="#">see page 422</a> )	A combination is defined by means of simple load cases with appropriate factors. .
	IRobotCaseFactorMngr ( <a href="#">see page 427</a> )	Interface granting access to the list of pairs (load case, coefficient). All pairs from the list are numbered (indexed) with numbers from 1 to the number of items (Count ( <a href="#">see page 428</a> )). .
	IRobotCaseFactor ( <a href="#">see page 430</a> )	Interface describing a pair (load case, coefficient) used in the definition of load case combination. .
	IRobotCaseCollection ( <a href="#">see page 431</a> )	Collection of load cases .
	IRobotCaseAnalysisModesFilter ( <a href="#">see page 432</a> )	Definition of modes taken into account during structure dynamic analysis. .
	IRobotDynamicAnalysisExcitationDirection ( <a href="#">see page 434</a> )	
	IRobotDynamicAnalysisDamping ( <a href="#">see page 439</a> )	Damping parameters for the dynamic analysis.

## II.1.1 Load records

A load record is used to model loads applied to a structure .

### Enumerations

	Name	Description
	IRobotLoadRecordType ( <a href="#">see page 54</a> )	Load record types available in the Robot Object Model .

### Interfaces

	Name	Description
	IRobotLoadRecord ( <a href="#">see page 55</a> )	The load record is used to model a load applied to a structure. RobotLoadRecord is described by an interface shared by all load record types available in Robot . .
	IRobotLoadRecordMngr ( <a href="#">see page 58</a> )	Load record manager manages logical lists of inter-linked records (e.g. a list of records linked with the same load case). The records are numbered from 1 to the number of records in the list .
	IRobotLoadRecordLinear3D ( <a href="#">see page 61</a> )	Load record of the I_LRT_LINEAR_3D type. .
	IRobotLoadRecordIn3Points ( <a href="#">see page 63</a> )	Load record for a load defined by three points .
	IRobotLoadRecordThermalIn3Points ( <a href="#">see page 65</a> )	Thermal load in three points. .
	IRobotLoadRecordInContour ( <a href="#">see page 68</a> )	Planar load on the contour. For a uniform load only the values for the first point of the load should be specified (I_ICRV_PX1, I_ICRV_PY1 and I_ICRV_PZ1) without providing the point coordinates. For a variable load all the load values and coordinates of the appropriate points should be specified applying the SetPoint ( <a href="#">see page 71</a> )() function.
	IRobotLoadRecordLinear ( <a href="#">see page 72</a> )	
	IRobotLoadRecord2 ( <a href="#">see page 74</a> )	Extended interface of the record that describes a load. .
	IRobotLoadRecordBarTrapezoidal ( <a href="#">see page 76</a> )	Trapezoidal load on bars.
	IRobotLoadRecordDead ( <a href="#">see page 78</a> )	Load record describing self-weight.
	IRobotLoadRecordCommon ( <a href="#">see page 80</a> )	The interface providing access to the extended set of common attributes for load records of different types.

### II.1.1.1 Values describing load records

Each load record is defined by means of several numerical values. The significance of particular values depends on the load record type. A set of identifiers is defined to refer to particular values in the relevant load records. In order to define correctly a load record, the appropriate set of values (defined for this record type) should be recognized first. The value identifiers are used as parameters of the SetValue() and GetValue() functions for the load record. .

### Enumerations

	Name	Description
	IRobotNodeForceRecordValues ( <a href="#">see page 39</a> )	Identifiers of the values describing load records of the following types: I_LRT_NODE_FORCE and I_LRT_NODE_FORCE_MASS. .
	IRobotNodeDisplacementRecordValues ( <a href="#">see page 40</a> )	The identifier of values for the records of the I_LRT_NODE_DISPLACEMENT type .
	IRobotNodeAuxiliaryRecordValues ( <a href="#">see page 40</a> )	Identifiers of values describing load records of the I_LRT_NODE_AUXILIARY type.
	IRobotBarDilatationRecordValues ( <a href="#">see page 40</a> )	Identifiers of the values describing load records of the following type: .

	IRobotBarConcentrateRecordValues (see page 41)	Identifier of the values describing load records of the following types: I_LRT_BAR_FORCE_CONCENTRATE, I_LRT_BAR_FORCE_CONCENTRATE_MASS .
	IRobotBarMomentDistributedRecordValues (see page 41)	Identifiers of the values describing load records of the following type: I_LRT_BAR_MOMENT_DISTRIBUTED .
	IRobotBarUniformRecordValues (see page 41)	Identifiers of the values describing load records of the following type: I_LRT_BAR_UNIFORM and I_LRT_BAR_UNIFORM_MASS .
	IRobotBarTrapezoidalRecordValues (see page 42)	Identifiers of the values describing load records of the following type: I_LRT_BAR_TRAPEZOIDALE and I_LRT_BAR_TRAPEZOIDALE_MASS .
	IRobotBarThermalRecordValues (see page 43)	Identifiers of the values describing load records of the following type: I_LRT_BAR_THERMAL .
	IRobotBarDeadRecordValues (see page 43)	Identifiers of the values describing load records of the following type: I_LRT_BAR_DEAD .
	IRobotPointAuxiliaryRecordValues (see page 44)	Identifiers of the values describing load records of the following type: I_LRT_POINT_AUXILIARY..
	IRobotIn3PointsRecordValues (see page 44)	Identifiers of the values describing load records of the following type: I_LRT_IN_3_POINTS..
	IRobotPressureRecordValues (see page 45)	Identifiers of the values describing load records of the following type: I_LRT_PRESSURE..
	IRobotLinearRecordValues (see page 45)	Identifiers of the values describing load records of the following type: I_LRT_LINEAR .
	IRobotMassActivationRecordValues (see page 46)	Identifiers of the values describing load records of the following type: I_LRT_MASS_ACTIVATION .
	IRobotLinear3DRecordValues (see page 46)	Identifiers of the values describing load records of the following type: I_LRT_LINEAR_3D .
	IRobotThermalIn3PointsRecordValues (see page 47)	Identifiers of the values describing load records of the following type: I_LRT_THERMAL_IN_3_POINTS (planar thermal load 3P)..
	IRobotInContourRecordValues (see page 48)	Identifiers of the values describing load records of the following type: I_LRT_IN_CONTOUR (planar load on contour - uniform or variable) .
	IRobotUniformRecordValues (see page 48)	Identifiers of the values describing load records of the following type: I_LRT_UNIFORM .
	IRobotThermalRecordValues (see page 49)	Identifiers of the values describing load records of the following type: I_LRT_THERMAL (thermal load: uniform, linear, at three points) .
	IRobotNodeForceInPointRecordValues (see page 49)	Values that define a load resulting from a force acting at a point (load record type : I_LRT_NODE_FORCE_IN_POINT).
	IRobotLinearOnEdgesRecordValues (see page 50)	Parameters of linear load on edges.
	IRobotDeadRecordValues (see page 51)	
	IRobotSurfaceOnObjectRecordValues (see page 51)	Identifiers of values for records of I_LRT_SURFACE_ON_OBJECT type.
	IRobotMobilePointForceRecordValues (see page 52)	Values describing a load record of the I_LLRT_MOBILE_POINT_FORCE type. .
	IRobotMobileDistributedRecordValues (see page 52)	Values describing a load of the I_LRT_MOBILE_DISTRIBUTED type. .
	IRobotNodeVelocityRecordValues (see page 53)	Identifiers of values for the load records of the I_LRT_NODE_VELOCITY type.
	IRobotNodeAccelerationRecordValues (see page 54)	Identifiers of values for the load records of the I_LRT_NODE_ACCELERATION type.

### II.1.1.1 IRobotNodeForceRecordValues

C++

```
enum IRobotNodeForceRecordValues;
```

**C#**

```
public enum IRobotNodeForceRecordValues;
```

**Visual Basic**

```
Public Enum IRobotNodeForceRecordValues
```

**Description**

Identifiers of the values describing load records of the following types: I\_LRT\_NODE\_FORCE and I\_LRT\_NODE\_FORCE\_MASS. .

**II.1.1.2 IRobotNodeDisplacementRecordValues****C++**

```
enum IRobotNodeDisplacementRecordValues;
```

**C#**

```
public enum IRobotNodeDisplacementRecordValues;
```

**Visual Basic**

```
Public Enum IRobotNodeDisplacementRecordValues
```

**Description**

The identifier of values for the records of the I\_LRT\_NODE\_DISPLACEMENT type .

**II.1.1.3 IRobotNodeAuxiliaryRecordValues****C++**

```
enum IRobotNodeAuxiliaryRecordValues;
```

**C#**

```
public enum IRobotNodeAuxiliaryRecordValues;
```

**Visual Basic**

```
Public Enum IRobotNodeAuxiliaryRecordValues
```

**Description**

Identifiers of values describing load records of the I\_LRT\_NODE\_AUXILIARY type.

**II.1.1.4 IRobotBarDilatationRecordValues****C++**

```
enum IRobotBarDilatationRecordValues;
```

**C#**

```
public enum IRobotBarDilatationRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarDilatationRecordValues
```

**Description**

Identifiers of the values describing load records of the following type: .

### II.1.1.5 IRobotBarForceConcentrateRecordValues

**C++**

```
enum IRobotBarForceConcentrateRecordValues;
```

**C#**

```
public enum IRobotBarForceConcentrateRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarForceConcentrateRecordValues
```

**Members**

Members	Description
I_BFCRV_OFFSET_Y = 21	Offset value of the force application point (Y local direction). Available since version 3.
I_BFCRV_OFFSET_Z = 22	Offset value of the force application point (Z local direction). Available since version 3.
I_BFCRV_GENERATE_CALC_NODE = 12	1 - generation of calculation node 0 - no calculation node generated. Available since version 11.

**Description**

Identifier of the values describing load records of the following types: I\_LRT\_BAR\_FORCE\_CONCENTRATE, I\_LRT\_BAR\_FORCE\_CONCENTRATE\_MASS .

### II.1.1.6 IRobotBarMomentDistributedRecordValues

**C++**

```
enum IRobotBarMomentDistributedRecordValues;
```

**C#**

```
public enum IRobotBarMomentDistributedRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarMomentDistributedRecordValues
```

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_BAR\_MOMENT\_DISTRIBUTED .

### II.1.1.7 IRobotBarUniformRecordValues

**C++**

```
enum IRobotBarUniformRecordValues;
```

**C#**

```
public enum IRobotBarUniformRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarUniformRecordValues
```

**Members**

Members	Description
I_BURV_PX = 0	Linear pressure (or mass value) in X direction.

I_BURV_PY = 1	Linear pressure (or mass value) in Y direction.
I_BURV_PZ = 2	Linear pressure (or mass value) in Z direction.
I_BURV_ALPHA = 8	Alpha angle of the force vector (if additional vector rotation is defined).
I_BURV_BETA = 9	Beta angle of the force vector (if additional vector rotation is defined).
I_BURV_GAMMA = 10	Gamma angle of the force vector (if additional vector rotation is defined).
I_BURV_LOCAL = 11	Zero value if loads values refer to global coordinate system. Otherwise - to local.
I_BURV_PROJECTION = 12	Flag indicating whether load is to be projected (value different from zero) or not (zero).
I_BURV_RELATIVE = 13	Understanding of the distance values measured along the bar length: relative - different from zero, absolute - zero.
I_BURV_OFFSET_Y = 21	Offset value of the uniform load application point (Y local direction). Available since version 3.
I_BURV_OFFSET_Z = 22	Offset value of the uniform load application point (Z local direction). Available since version 3.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_BAR\_UNIFORM and I\_LRT\_BAR\_UNIFORM\_MASS .

**II.1.1.8 IRobotBarTrapezoidalRecordValues****C++**

```
enum IRobotBarTrapezoidalRecordValues;
```

**C#**

```
public enum IRobotBarTrapezoidalRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarTrapezoidalRecordValues
```

**Members**

Members	Description
I_BTRV_PX2 = 0	A value of the X-component of the linear bar pressure in ending point.
I_BTRV_PY2 = 1	A value of the Y-component of the linear bar pressure in ending point.
I_BTRV_PZ2 = 2	A value of the Z-component of the linear bar pressure in ending point.
I_BTRV_PX1 = 3	A value of the X-component of the linear bar pressure in starting point.
I_BTRV_PY1 = 4	A value of the Y-component of the linear bar pressure in starting point.
I_BTRV_PZ1 = 5	A value of the Z-component of the linear bar pressure in starting point.
I_BTRV_X2 = 6	Distance of the ending point from the bar origin.
I_BTRV_X1 = 7	Distance of the starting point from the bar origin.
I_BTRV_ALPHA = 8	Alpha angle of the force vector (if additional vector rotation is defined).

I_BTRV_BETA = 9	Beta angle of the force vector (if additional vector rotation is defined).
I_BTRV_GAMMA = 10	Gamma angle of the force vector (if additional vector rotation is defined).
I_BTRV_PROJECTION = 12	Flag indicating whether the load is to be projected (value different from zero) or not (zero value).
I_BTRV_RELATIVE = 13	Understanding of the distance values measured along the bar length: relative - different from zero, absolute - zero.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_BAR\_TRAPEZOIDALE and I\_LRT\_BAR\_TRAPEZOIDALE\_MASS.

**II.1.1.9 IRobotBarThermalRecordValues****C++**

```
enum IRobotBarThermalRecordValues;
```

**C#**

```
public enum IRobotBarThermalRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarThermalRecordValues
```

**Members**

Members	Description
I_BTRV_TX = 0	Uniform temperature change (heating or cooling) of the bar.
I_BTRV TY = 1	Difference in the temperature between the extreme cross section fibers along local y bar axis.
I_BTRV_TZ = 2	Difference in the temperature between the extreme cross section fibers along local z bar axis.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_BAR\_THERMAL .

**II.1.1.10 IRobotBarDeadRecordValues****C++**

```
enum IRobotBarDeadRecordValues;
```

**C#**

```
public enum IRobotBarDeadRecordValues;
```

**Visual Basic**

```
Public Enum IRobotBarDeadRecordValues
```

**Members**

Members	Description
I_BDRV_ENTIRE_STRUCTURE = 15	Flag indicating that load is applied to the whole structure. Available since version 3.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_BAR\_DEAD. .

### II.1.1.11 IRobotPointAuxiliaryRecordValues

#### C++

```
enum IRobotPointAuxiliaryRecordValues;
```

#### C#

```
public enum IRobotPointAuxiliaryRecordValues;
```

#### Visual Basic

```
Public Enum IRobotPointAuxiliaryRecordValues
```

#### Members

Members	Description
I_PARV_FX = 0	X component value of the force load.
I_PARV_FY = 1	Y component value of the force load.
I_PARV_FZ = 2	Z component value of the force load.
I_PARV_CX = 3	X component value of the moment load.
I_PARV_CY = 4	Y component value of the moment load.
I_PARV_CZ = 5	Z component value of the moment load.
I_PARV_ALPHA = 8	Alpha angle of the force vector (if additional vector rotation is defined).
I_PARV_BETA = 9	Beta angle of the force vector (if additional vector rotation is defined).
I_PARV_GAMMA = 10	Gamma angle of the force vector (if additional vector rotation is defined).
I_PARV_X = 11	X coordinate of a point in which force load is applied.
I_PARV_Y = 12	Y coordinate of a point in which force load is applied.
I_PARV_Z = 13	Z coordinate of a point in which force load is applied.

#### Description

Identifiers of the values describing load records of the following type: I\_LRT\_POINT\_AUXILIARY. .

### II.1.1.12 IRobotIn3PointsRecordValues

#### C++

```
enum IRobotIn3PointsRecordValues;
```

#### C#

```
public enum IRobotIn3PointsRecordValues;
```

#### Visual Basic

```
Public Enum IRobotIn3PointsRecordValues
```

#### Members

Members	Description
I_3PRV_PX1 = 0	A value of the pressure PX intensity acting in point 1.
I_3PRV_PY1 = 1	A value of the pressure PY intensity acting in point 2.
I_3PRV_PZ1 = 2	A value of the pressure PZ intensity acting in point 3.
I_3PRV_PX2 = 3	A value of the pressure PX intensity acting in point 2.
I_3PRV_PY2 = 4	A value of the pressure PY intensity acting in point 2.
I_3PRV_PZ2 = 5	A value of the pressure PZ intensity acting in point 3.
I_3PRV_PX3 = 6	A value of the pressure PX intensity acting in point 3.

I_3PRV_PY3 = 7	A value of the pressure PY intensity acting in point 2.
I_3PRV_PZ3 = 8	A value of the pressure PZ intensity acting in point 3.
I_3PRV_PROJECTION = 12	Projected/not projected load value (0/-1).
I_3PRV_N1 = 9	The first node number in which load pressure values are defined.
I_3PRV_N2 = 10	The second node number in which load pressure values are defined.
I_3PRV_N3 = 11	The third node number in which load pressure values are defined.
I_3PRV_LOCAL_SYSTEM = 13	Local/global loads.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_IN\_3\_POINTS. .

**II.1.1.13 IRobotPressureRecordValues****C++**

```
enum IRobotPressureRecordValues;
```

**C#**

```
public enum IRobotPressureRecordValues;
```

**Visual Basic**

```
Public Enum IRobotPressureRecordValues
```

**Members**

Members	Description
I_PRV_P = 0	Constant pressure value (independent from liquid).
I_PRV_RO = 1	Unit weight of liquid Available since version 2.0.
I_PRV_H = 2	Liquid level Available since version 2.0.
I_PRV_DIRECTION = 6	Direction of gravity for liquid (0 : -X, 1 : -Y, 2 : -Z, 3 : X, 4 : Y, 5 : Z).

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_PRESSURE. .

**II.1.1.14 IRobotLinearRecordValues****C++**

```
enum IRobotLinearRecordValues;
```

**C#**

```
public enum IRobotLinearRecordValues;
```

**Visual Basic**

```
Public Enum IRobotLinearRecordValues
```

**Members**

Members	Description
I_LRV_MX1 = 0	A value of the moment MX acting at point 1.
I_LRV_MX2 = 1	A value of the moment MX acting at point 2.
I_LRV_PZ1 = 2	A value of the force PZ acting at point 1.
I_LRV_PZ2 = 3	A value of the force PZ acting at point 2.

I_LRV_MY1 = 4	A value of the moment MY acting at point 1.
I_LRV_MY2 = 5	A value of the moment MY acting at point 2.
I_LRV_NODE_1 = 6	Node number defined at point 1.
I_LRV_NODE_2 = 7	Node number defined at point 1.
I_LRV_LOCAL = 11	Zero value if loads values refer to global coordinate system. Otherwise - to local.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_LINEAR .

**II.1.1.15 IRobotMassActivationRecordValues****C++**

```
enum IRobotMassActivationRecordValues;
```

**C#**

```
public enum IRobotMassActivationRecordValues;
```

**Visual Basic**

```
Public Enum IRobotMassActivationRecordValues
```

**Members**

Members	Description
I_MARV_INPUT_DIR_X = 0	Flag to add/activate mass in global X-direction.
I_MARV_INPUT_DIR_Y = 1	Flag to add/activate mass in global Y-direction.
I_MARV_INPUT_DIR_Z = 2	Flag to add/activate mass in global Z-direction.
I_MARV_FACTOR = 3	Load to mass conversion coefficient (standard = 1.0).
I_MARV_CASE_NUM = 4	Load case which is to be converted.
I_MARV_ACTIVATION_DIR = 6	Direction of mass conversion (load direction which components are to be converted to masses: 0 denotes X, 1 - Y, 2 - Z).
I_MARV_SIGN = 7	Conversion sign (direction of loads supplementary to ACTIVATION_DIR) +1 or -1.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_MASS\_ACTIVATION .

**II.1.1.16 IRobotLinear3DRecordValues****C++**

```
enum IRobotLinear3DRecordValues;
```

**C#**

```
public enum IRobotLinear3DRecordValues;
```

**Visual Basic**

```
Public Enum IRobotLinear3DRecordValues
```

**Members**

Members	Description
I_L3DRV_PX1 = 0	A value of the intensity of the linearly changing load PX acting in point 1 (linear load is defined in 2 points).
I_L3DRV_PY1 = 1	A value of the intensity of the linearly changing load PY acting in point 1 (linear load is defined in 2 points).

I_L3DRV_PZ1 = 2	A value of the intensity of the linearly changing load PZ acting in point 1 (linear load is defined in 2 points).
I_L3DRV_MX1 = 3	A value of the intensity of the linearly changing moment MX acting in point 1 (linear load is defined in 2 points).
I_L3DRV_MY1 = 4	A value of the intensity of the linearly changing moment MY acting in point 1 (linear load is defined in 2 points).
I_L3DRV_MZ1 = 5	A value of the intensity of the linearly changing moment MZ acting in point 1 (linear load is defined in 2 points).
I_L3DRV_PX2 = 6	A value of the intensity of the linearly changing load PX acting in point 2 (linear load is defined in 2 points).
I_L3DRV_PY2 = 7	A value of the intensity of the linearly changing load PY acting in point 2 (linear load is defined in 2 points).
I_L3DRV_PZ2 = 8	A value of the intensity of the linearly changing load PZ acting in point 2 (linear load is defined in 2 points).
I_L3DRV_MX2 = 9	A value of the intensity of the linearly changing moment MX acting in point 2 (linear load is defined in 2 points).
I_L3DRV_MY2 = 10	A value of the intensity of the linearly changing moment MY acting in point 2 (linear load is defined in 2 points).
I_L3DRV_MZ2 = 11	A value of the intensity of the linearly changing moment MZ acting in point 2 (linear load is defined in 2 points).
I_L3DRV_LOCAL = 13	Zero value if loads values refer to global coordinate system. Otherwise - to local.
I_L3DRV_GAMMA = 15	Rotation angle of the defined linear loads measured around the line of action.

### Description

Identifiers of the values describing load records of the following type: I\_LRT\_LINEAR\_3D .

#### II.1.1.17 IRobotThermalIn3PointsRecordValues

##### C++

```
enum IRobotThermalIn3PointsRecordValues;
```

##### C#

```
public enum IRobotThermalIn3PointsRecordValues;
```

##### Visual Basic

```
Public Enum IRobotThermalIn3PointsRecordValues
```

##### Members

Members	Description
I_3PRV_TX1 = 0	Change of the middle surface temperature in the 1st point of surface for uniform or variable temperature change definition.
I_3PRV_TZ1 = 2	Change of the temperature gradient (along local z-axis) in the 1st point of surface for uniform or variable temperature change definition.
I_3PRV_TX2 = 3	Change of the middle surface temperature in the 2nd point of surface for uniform or variable temperature change definition.
I_3PRV_TZ2 = 5	Change of the temperature gradient (along local z-axis) in the 2nd point of surface for uniform or variable temperature change definition.
I_3PRV_TX3 = 6	Change of the middle surface temperature in the 3rd point of surface for uniform or variable temperature change definition.
I_3PRV_TZ3 = 8	Change of the temperature gradient (along local z-axis) in the 3rd point of surface for uniform or variable temperature change definition.

## Description

Identifiers of the values describing load records of the following type: I\_LRT\_THERMAL\_IN\_3\_POINTS (planar thermal load 3P). .

### II.1.1.18 IRobotInContourRecordValues

#### C++

```
enum IRobotInContourRecordValues;
```

#### C#

```
public enum IRobotInContourRecordValues;
```

#### Visual Basic

```
Public Enum IRobotInContourRecordValues
```

#### Members

Members	Description
I_ICRV_PX1 = 0	Value of the X component of the pressure load in point 1 (for planar load defined in 3 points).
I_ICRV_PY1 = 1	Value of the Y component of the pressure load in point 1 (for planar load defined in 3 points).
I_ICRV_PZ1 = 2	Value of the Z component of the pressure load in point 1 (for planar load defined in 3 points).
I_ICRV_PX2 = 3	Value of the X component of the pressure load in point 2 (for planar load defined in 3 points).
I_ICRV_PY2 = 4	Value of the Y component of the pressure load in point 2 (for planar load defined in 3 points).
I_ICRV_PZ2 = 5	Value of the Z component of the pressure load in point 2 (for planar load defined in 3 points).
I_ICRV_PX3 = 6	Value of the X component of the pressure load in point 3 (for planar load defined in 3 points).
I_ICRV_PY3 = 7	Value of the Y component of the pressure load in point 3 (for planar load defined in 3 points).
I_ICRV_PZ3 = 8	Value of the Z component of the pressure load in point 3 (for planar load defined in 3 points).
I_ICRV_PROJECTION = 12	Flag indicating whether the load is to be projected (value different from zero) or not (zero).
I_ICRV_NPOINTS = 13	Number of points defining a contour.
I_ICRV_LOCAL = 15	Variable describing coordinate system: local (the value not equal zero) or global (zero).

#### Description

Identifiers of the values describing load records of the following type: I\_LRT\_IN\_CONTOUR (planar load on contour - uniform or variable) .

### II.1.1.19 IRobotUniformRecordValues

#### C++

```
enum IRobotUniformRecordValues;
```

#### C#

```
public enum IRobotUniformRecordValues;
```

**Visual Basic**

```
Public Enum IRobotUniformRecordValues
```

**Members**

Members	Description
I_URV_PX = 0	Value of the X component of the pressure load acting on shell element.
I_URV_PY = 1	Value of the Y component of the pressure load acting on shell element.
I_URV_PZ = 2	Value of the Z component of the pressure load acting on shell element.
I_URV_LOCAL_SYSTEM = 11	Parameter assumes the value 0 (zero) when the load acts in the global system and the value different from zero when the load acts in the local system .
I_URV_PROJECTED = 12	Flag indicating whether the load is to be projected (value different from zero) or not (zero).

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_UNIFORM .

**II.1.1.20 IRobotThermalRecordValues****C++**

```
enum IRobotThermalRecordValues;
```

**C#**

```
public enum IRobotThermalRecordValues;
```

**Visual Basic**

```
Public Enum IRobotThermalRecordValues
```

**Members**

Members	Description
I_TRV_T_1 = 0	Uniform temperature change (heating or cooling) of the bar.
I_TRV_T_2 = 3	Difference in the temperature between the extreme cross section fibers along local y bar axis.
I_TRV_T_3 = 6	Difference in the temperature between the extreme cross section fibers along local z bar axis.

**Description**

Identifiers of the values describing load records of the following type: I\_LRT\_THERMAL (thermal load: uniform, linear, at three points) .

**II.1.1.21 IRobotNodeForceInPointRecordValues****C++**

```
enum IRobotNodeForceInPointRecordValues;
```

**C#**

```
public enum IRobotNodeForceInPointRecordValues;
```

**Visual Basic**

```
Public Enum IRobotNodeForceInPointRecordValues
```

## Members

Members	Description
I_NFIPRV_FX = 0	A value of concentrated force acting along the X direction Available since version 1.7.
I_NFIPRV_FY = 1	A value of concentrated force acting along the Y direction Available since version 1.7.
I_NFIPRV_FZ = 2	A value of concentrated force acting along the Z direction Available since version 1.7.
I_NFIPRV_MX = 3	A value of moment (about X axis) Available since version 1.7.
I_NFIPRV_MY = 4	A value of moment (about Y axis) Available since version 1.7.
I_NFIPRV_MZ = 5	A value of moment (about Z axis) Available since version 1.7.
I_NFIPRV_ALPHA = 8	Alpha angle of the force vector (if additional vector rotation defined).
I_NFIPRV_BETA = 9	Beta angle of the force vector (if additional vector rotation defined).
I_NFIPRV_GAMMA = 10	Gamma angle of the force vector (if additional vector rotation defined).
I_NFIPRV_POINT_X = 11	X coordinate of the point of load application Available since version 1.7.
I_NFIPRV_POINT_Y = 12	Y coordinate of the point of load application Available since version 1.7.
I_NFIPRV_POINT_Z = 13	Z coordinate of the point of load application Available since version 1.7.

## Description

Values that define a load resulting from a force acting at a point (load record type : I\_LRT\_NODE\_FORCE\_IN\_POINT).

### II.1.1.22 IRobotLinearOnEdgesRecordValues

#### C++

```
enum IRobotLinearOnEdgesRecordValues;
```

#### C#

```
public enum IRobotLinearOnEdgesRecordValues;
```

#### Visual Basic

```
Public Enum IRobotLinearOnEdgesRecordValues
```

## Members

Members	Description
I_LOERV_PX = 0	Available since version 2.0.
I_LOERV_PY = 1	Available since version 2.0.
I_LOERV_PZ = 2	Available since version 2.0.
I_LOERV_MX = 3	Available since version 2.0.
I_LOERV_MY = 4	Available since version 2.0.
I_LOERV_MZ = 5	Available since version 2.0.
I_LOERV_GAMMA = 6	Available since version 2.0.
I_LOERV_LOCAL_SYSTEM = 11	Available since version 2.0.

## Description

Parameters of linear load on edges.

### II.1.1.23 IRobotDeadRecordValues

**C++**

```
enum IRobotDeadRecordValues;
```

**C#**

```
public enum IRobotDeadRecordValues;
```

**Visual Basic**

```
Public Enum IRobotDeadRecordValues
```

**Members**

Members	Description
I_DRV_X = 0	Weight multiplier for X-direction (if gravity load acts along this direction). Available since version 3.
I_DRV_Y = 1	Weight multiplier for Y-direction. Available since version 3.
I_DRV_Z = 2	Weight multiplier for Z-direction. Available since version 3.
I_DRV_ENTIRE_STRUCTURE = 15	Flag which shows if weight of all structural elements is taken into account. Available since version 3.
I_DRV_COEFF = 3	Available since version 12.

**Version**

Available since version 3.

### II.1.1.24 IRobotSurfaceOnObjectRecordValues

**C++**

```
enum IRobotSurfaceOnObjectRecordValues;
```

**C#**

```
public enum IRobotSurfaceOnObjectRecordValues;
```

**Visual Basic**

```
Public Enum IRobotSurfaceOnObjectRecordValues
```

**Members**

Members	Description
I_SOORV_PX = 0	Pressure value (on X-direction). Available since version 3.
I_SOORV_PY = 1	Pressure value (on Y-direction). Available since version 3.
I_SOORV_PZ = 2	Pressure value (on Z-direction). Available since version 3.
I_SOORV_LOCAL = 11	If equal zero - loads are defined in global coordinate system. Available since version 3.

**Description**

Identifiers of values for records of I\_LRT\_SURFACE\_ON\_OBJECT type.

**Version**

Available since version 3.

**II.1.1.1.25 IRobotMobilePointForceRecordValues****C++**

```
enum IRobotMobilePointForceRecordValues;
```

**C#**

```
public enum IRobotMobilePointForceRecordValues;
```

**Visual Basic**

```
Public Enum IRobotMobilePointForceRecordValues
```

**Members**

Members	Description
I_MPFRV_X = 0	X coordinate of load application point. Available since version 3.
I_MPFRV_Y = 1	Y coordinate of load application point. Available since version 3.
I_MPFRV_Z = 2	Z coordinate of load application point. Available since version 3.
I_MPFRV_FX = 3	Concentrated load value towards route local direction X. Available since version 3.
I_MPFRV_FY = 4	Concentrated load value towards route local direction Y. Available since version 3.
I_MPFRV_FZ = 5	Concentrated load value towards route local direction Z (downwards). Available since version 3.

**Description**

Values describing a load record of the I\_LRT\_MOBILE\_POINT\_FORCE type. . .

**Version**

Available since version 3.

**II.1.1.1.26 IRobotMobileDistributedRecordValues****C++**

```
enum IRobotMobileDistributedRecordValues;
```

**C#**

```
public enum IRobotMobileDistributedRecordValues;
```

**Visual Basic**

```
Public Enum IRobotMobileDistributedRecordValues
```

**Members**

Members	Description
I_MDRV_X = 0	X coordinate of load application point. Available since version 3.
I_MDRV_Y = 1	Y coordinate of load application point. Available since version 3.

I_MDRV_Z = 2	Z coordinate of load application point. Available since version 3.
I_MDRV_PX = 3	Available since version 3.
I_MDRV_PY = 4	Available since version 3.
I_MDRV_PZ = 5	Surface load intensity towards route local direction Z (downwards). Available since version 3.
I_MDRV_DX = 6	Length of a segment/vehicle side along which the load is acting (along the vehicle axis). Available since version 3.
I_MDRV_DY = 7	Length of a segment/vehicle side along which the load is acting (perpendicular to the vehicle axis). Available since version 3.

**Description**

Values describing a load of the I\_LRT\_MOBILE\_DISTRIBUTED type. .

**Version**

Available since version 3.

**II.1.1.27 IRobotNodeVelocityRecordValues****C++**

```
enum IRobotNodeVelocityRecordValues;
```

**C#**

```
public enum IRobotNodeVelocityRecordValues;
```

**Visual Basic**

```
Public Enum IRobotNodeVelocityRecordValues
```

**Members**

Members	Description
I_NVRV_UX = 0	.
I_NVRV_UY = 1	. Available since version 15.
I_NVRV_UZ = 2	.
I_NVRV_ALPHA = 8	. Available since version 15.
I_NVRV_BETA = 9	.
I_NVRV_GAMMA = 10	. Available since version 15.

**Description**

Identifiers of values for the load records of the I\_LRT\_NODE\_VELOCITY type.

**Version**

Available since version 15.

### II.1.1.28 IRobotNodeAccelerationRecordValues

#### C++

```
enum IRobotNodeAccelerationRecordValues;
```

#### C#

```
public enum IRobotNodeAccelerationRecordValues;
```

#### Visual Basic

```
Public Enum IRobotNodeAccelerationRecordValues
```

#### Members

Members	Description
I_NACRV_UX = 0	.
I_NACRV_UY = 1	. Available since version 15.
I_NACRV_UZ = 2	.
I_NACRV_ALPHA = 8	.
I_NACRV_BETA = 9	.
I_NACRV_GAMMA = 10	. Available since version 15.

#### Description

Identifiers of values for the load records of the I\_LRT\_NODE\_ACCELERATION type.

#### Version

Available since version 15.

### II.1.2 IRobotLoadRecordType

#### C++

```
enum IRobotLoadRecordType;
```

#### C#

```
public enum IRobotLoadRecordType;
```

#### Visual Basic

```
Public Enum IRobotLoadRecordType
```

#### Members

Members	Description
I_LRT_NODE_FORCE = 0	Nodal force.
I_LRT_NODE_DISPLACEMENT = 1	Displacements of a support node.
I_LRT_BAR_DILATATION = 2	Shortening/extension of a bar.
I_LRT_BAR_FORCE_CONCENTRATED = 3	Concentrated force on the bar length.
I_LRT_BAR_MOMENT_DISTRIBUTED = 4	Distributed moment load on a bar.
I_LRT_BAR_UNIFORM = 5	Uniform linear load on a bar.
I_LRT_BAR_TRAPEZOIDALE = 6	Linear trapezoidal load on a bar.

I_LRT_BAR_THERMAL = 8	Thermal load on a bar.
I_LRT_LINEAR = 21	Linear load on FE, between two points (for slabs - 2D).
I_LRT_LINEAR_3D = 19	Linear load on FE, between two points (3D space).
I_LRT_POINT_AUXILIARY = 23	Concentrated load at a point.
I_LRT_IN_3_POINTS = 22	Planar load defined at three points.
I_LRT_PRESSURE = 24	Hydrostatic pressure.
I_LRT_THERMAL_IN_3_POINTS = 25	Planar thermal load defined at three points.
I_LRT_IN_CONTOUR = 28	Planar load defined on the contour (uniform or given at three points).
I_LRT_NODE_FORCE_MASS = 30	Concentrated nodal mass.
I_LRT_BAR_FORCE_CONCENTRATED_MASS = 33	Concentrated mass at a point on the bar length.
I_LRT_BAR_UNIFORM_MASS = 35	Uniform linear mass on a bar.
I_LRT_BAR_TRAPEZOIDALE_MASS = 36	Trapezoidal linear mass on a bar.
I_LRT_MASS_ACTIVATION = 39	Activation of a record for masses.
I_LRT_SPECTRUM_VALUE = 40	Spectrum value.
I_LRT_UNIFORM = 26	Uniform surface load.
I_LRT_THERMAL = 8	Planar thermal load.
I_LRT_NODE_FORCE_IN_POINT = 23	Concentrated force in the point Available since version 1.7.
I_LRT_LINEAR_ON_EDGES = 69	Linear load on edges Available since version 1.7.
I_LRT_DEAD = 7	Self-weight Available since version 2.0.
I_LRT_SURFACE_ON_OBJECT = 70	Surface load on object. Available since version 3.
I_LRT_MOBILE_POINT_FORCE = 53	Moving load - point force. Available since version 3.
I_LRT_MOBILE_DISTRIBUTED = 55	A moving distributed load: <ul style="list-style-type: none"><li>• linear, if value Dx or Dy equals zero</li><li>• planar, if Dx and Dy assume non-zero values</li></ul> Available since version 3.
I_LRT_NODE_VELOCITY = 10	. Available since version 15.
I_LRT_NODE_ACCELERATION = 11	. Available since version 15.

## Description

Load record types available in the Robot Object Model .

### II.1.1.3 IRobotLoadRecord

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecord : IDispatch;
```

#### C#

```
public interface IRobotLoadRecord;
```

#### Visual Basic

```
Public Interface IRobotLoadRecord
```

## Description

The load record is used to model a load applied to a structure. RobotLoadRecord is described by an interface shared by all load record types available in Robot. .

### II.1.1.3.1 IRobotLoadRecord Members

The following tables list the members exposed by IRobotLoadRecord.

#### Public Fields

	Name	Description
◆	Description (see page 56)	Load record description .
◆	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
◆	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
◆	Type (see page 57)	Load record type..

#### Public Methods

	Name	Description
◆	GetValue (see page 58)	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
◆	SetValue (see page 58)	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.3.2 IRobotLoadRecord Fields

The fields of the IRobotLoadRecord class are listed here.

#### Public Fields

	Name	Description
◆	Description (see page 56)	Load record description .
◆	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
◆	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
◆	Type (see page 57)	Load record type..

### II.1.1.3.2.1 Description

#### C++

```
HRESULT get_Description(BSTR* );
HRESULT put_Description(BSTR);
```

#### C#

```
public String Description { get; set; }
```

#### Visual Basic

```
Public Description As String
```

#### Description

Load record description .

### II.1.1.3.2.2 Objects

#### C++

```
HRESULT get_Objects(IRobotSelection**);
```

#### C#

```
public IRobotSelection Objects { get; }
```

#### Visual Basic

```
Public ReadOnly Objects As IRobotSelection
```

#### Description

Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .

### II.1.1.3.2.3 ObjectType

#### C++

```
HRESULT get_ObjectType(IRobotObjectType*);  
HRESULT put_ObjectType(IRobotObjectType);
```

#### C#

```
public IRobotObjectType ObjectType { get; set; }
```

#### Visual Basic

```
Public ObjectType As IRobotObjectType
```

#### Description

Type (see page 57) of the object where a load is applied. .

### II.1.1.3.2.4 Type

#### C++

```
HRESULT get_Type(IRobotLoadRecordType*);
```

#### C#

```
public IRobotLoadRecordType Type { get; }
```

#### Visual Basic

```
Public ReadOnly Type As IRobotLoadRecordType
```

#### Description

Load record type. .

### II.1.1.3.3 IRobotLoadRecord Methods

The methods of the IRobotLoadRecord class are listed here.

#### Public Methods

	Name	Description
💡	GetValue (see page 58)	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

	SetValue (see page 58)	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
-----------------------------------------------------------------------------------	------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### II.1.1.3.3.1 GetValue

C++

```
HRESULT GetValue(short _value_id, double* ret);
```

C#

```
public double GetValue(short _value_id);
```

Visual Basic

```
Public Function GetValue(_value_id As short) As double
```

Description

The function returns the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.3.3.2 SetValue

C++

```
HRESULT SetValue(short _value_id, double _value);
```

C#

```
public void SetValue(short _value_id, double _value);
```

Visual Basic

```
Public Sub SetValue(_value_id As short, _value As double)
```

Description

The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.4 IRobotLoadRecordMngr

Class Hierarchy

C++

```
interface IRobotLoadRecordMngr : IDispatch;
```

C#

```
public interface IRobotLoadRecordMngr;
```

Visual Basic

```
Public Interface IRobotLoadRecordMngr
```

Description

Load record manager manages logical lists of inter-linked records (e.g. a list of records linked with the same load case). The records are numbered from 1 to the number of records in the list .

### II.1.1.4.1 IRobotLoadRecordMngr Members

The following tables list the members exposed by IRobotLoadRecordMngr.

#### Public Fields

	Name	Description
◆	Count (see page 59)	Number of currently managed load records .

#### Public Methods

	Name	Description
◆	Create (see page 60)	
◆	Delete (see page 60)	The function deletes the record with the indicated index. The list of all records is automatically re-numbered after this operation. .
◆	Get (see page 60)	The function returns a record with the indicated index on the list of the managed load records. .
◆	New (see page 60)	The function creates a new load record of the indicated type. The newly-created record is added at the end of the managed list. The function gives the index of the created record back. .

### II.1.1.4.2 IRobotLoadRecordMngr Fields

The fields of the IRobotLoadRecordMngr class are listed here.

#### Public Fields

	Name	Description
◆	Count (see page 59)	Number of currently managed load records .

### II.1.1.4.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of currently managed load records .

### II.1.1.4.3 IRobotLoadRecordMngr Methods

The methods of the IRobotLoadRecordMngr class are listed here.

#### Public Methods

	Name	Description
◆	Create (see page 60)	
◆	Delete (see page 60)	The function deletes the record with the indicated index. The list of all records is automatically re-numbered after this operation. .
◆	Get (see page 60)	The function returns a record with the indicated index on the list of the managed load records. .
◆	New (see page 60)	The function creates a new load record of the indicated type. The newly-created record is added at the end of the managed list. The function gives the index of the created record back. .

### II.1.1.4.3.1 Create

**C++**

```
HRESULT Create(IRobotLoadRecordType _load_type, IRobotLoadRecord** ret);
```

**C#**

```
public IRobotLoadRecord Create(IRobotLoadRecordType _load_type);
```

**Visual Basic**

```
Public Function Create(_load_type As IRobotLoadRecordType) As IRobotLoadRecord
```

**Version**

Available since version 8.5.

### II.1.1.4.3.2 Delete

**C++**

```
HRESULT Delete(long _record_index);
```

**C#**

```
public void Delete(long _record_index);
```

**Visual Basic**

```
Public Sub Delete(_record_index As long)
```

**Description**

The function deletes the record with the indicated index. The list of all records is automatically re-numbered after this operation. .

### II.1.1.4.3.3 Get

**C++**

```
HRESULT Get(long _record_index, IRobotLoadRecord** ret);
```

**C#**

```
public IRobotLoadRecord Get(long _record_index);
```

**Visual Basic**

```
Public Function Get(_record_index As long) As IRobotLoadRecord
```

**Description**

The function returns a record with the indicated index on the list of the managed load records. .

### II.1.1.4.3.4 New

**C++**

```
HRESULT New(IRobotLoadRecordType _record_type, long* ret);
```

**C#**

```
public long New(IRobotLoadRecordType _record_type);
```

**Visual Basic**

```
Public Function New(_record_type As IRobotLoadRecordType) As long
```

## Description

The function creates a new load record of the indicated type. The newly-created record is added at the end of the managed list. The function gives the index of the created record back. .

### II.1.1.5 IRobotLoadRecordLinear3D

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecordLinear3D : IRobotLoadRecord;
```

#### C#

```
public interface IRobotLoadRecordLinear3D : IRobotLoadRecord;
```

#### Visual Basic

```
Public Interface IRobotLoadRecordLinear3D
```

#### Description

Load record of the I\_LRT\_LINEAR\_3D type. .

### II.1.1.5.1 IRobotLoadRecordLinear3D Members

The following tables list the members exposed by IRobotLoadRecordLinear3D.

#### Public Fields

	Name	Description
❖	Description ( [ see page 56) )	Load record description .
❖	Objects ( [ see page 57) )	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
❖	ObjectType ( [ see page 57) )	Type ( [ see page 57) ) of the object where a load is applied .
❖	Type ( [ see page 57) )	Load record type.. .
❖	UniqueId ( [ see page 62) )	Unique identifier of a load record Available since version 1.7. .

#### Public Methods

	Name	Description
❖	GetPoint ( [ see page 62) )	The function returns the coordinates of the indicated point of load application. .
❖	GetValue ( [ see page 58) )	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
❖	SetPoint ( [ see page 62) )	The function sets the coordinates of the indicated point of load application. .
❖	SetValue ( [ see page 58) )	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.5.2 IRobotLoadRecordLinear3D Fields

The fields of the IRobotLoadRecordLinear3D class are listed here.

## Public Fields

	Name	Description
◆	UniqueId ( [ see page 62) )	Unique identifier of a load record Available since version 1.7.

### II.1.1.5.2.1 UniqueId

#### C++

```
HRESULT get_UniqueId(long*);
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As long
```

#### Description

Unique identifier of a load record Available since version 1.7.

## II.1.1.5.3 IRobotLoadRecordLinear3D Methods

The methods of the IRobotLoadRecordLinear3D class are listed here.

### Public Methods

	Name	Description
◆	GetPoint ([ see page 62)	The function returns the coordinates of the indicated point of load application..
◆	SetPoint ([ see page 62)	The function sets the coordinates of the indicated point of load application..

### II.1.1.5.3.1 GetPoint

#### C++

```
HRESULT GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### C#

```
public void GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Sub GetPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

#### Description

The function returns the coordinates of the indicated point of load application. .

### II.1.1.5.3.2 SetPoint

#### C++

```
HRESULT SetPoint(int _which_point, double _x, double _y, double _z);
```

#### C#

```
public void SetPoint(int _which_point, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetPoint(_which_point As int, _x As double, _y As double, _z As double)
```

## Description

The function sets the coordinates of the indicated point of load application. .

### II.1.1.6 IRobotLoadRecordIn3Points

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecordIn3Points : IRobotLoadRecord;
```

#### C#

```
public interface IRobotLoadRecordIn3Points : IRobotLoadRecord;
```

#### Visual Basic

```
Public Interface IRobotLoadRecordIn3Points
```

#### Description

Load record for a load defined by three points .

### II.1.1.6.1 IRobotLoadRecordIn3Points Members

The following tables list the members exposed by IRobotLoadRecordIn3Points.

#### Public Fields

	Name	Description
❖	Description ( <a href="#">see page 56</a> )	Load record description .
❖	Objects ( <a href="#">see page 57</a> )	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
❖	ObjectType ( <a href="#">see page 57</a> )	Type ( <a href="#">see page 57</a> ) of the object where a load is applied .
❖	Type ( <a href="#">see page 57</a> )	Load record type..
❖	UniqueId ( <a href="#">see page 64</a> )	Unique identifier of a load record Available since version 1.7.

#### Public Methods

	Name	Description
❖	GetGeoLimits ( <a href="#">see page 64</a> )	Function returns geometrical limits applied to loads. Available since version 2.0.
❖	GetPoint ( <a href="#">see page 64</a> )	The function returns coordinates of the indicated point of load application. .
❖	GetValue ( <a href="#">see page 58</a> )	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
❖	SetGeoLimits ( <a href="#">see page 65</a> )	Function sets geometrical limits for loads. Available since version 2.0.
❖	SetPoint ( <a href="#">see page 65</a> )	The function sets the coordinates of the indicated point of load application. .
❖	SetValue ( <a href="#">see page 58</a> )	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.6.2 IRobotLoadRecordIn3Points Fields

The fields of the IRobotLoadRecordIn3Points class are listed here.

## Public Fields

	Name	Description
◆	UniqueId ( <a href="#">see page 64</a> )	Unique identifier of a load record Available since version 1.7.

### II.1.1.6.2.1 UniqueId

#### C++

```
HRESULT get_UniqueId(long*);
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As long
```

#### Description

Unique identifier of a load record Available since version 1.7.

## II.1.1.6.3 IRobotLoadRecordIn3Points Methods

The methods of the IRobotLoadRecordIn3Points class are listed here.

### Public Methods

	Name	Description
◆	GetGeoLimits ( <a href="#">see page 64</a> )	Function returns geometrical limits applied to loads. Available since version 2.0.
◆	GetPoint ( <a href="#">see page 64</a> )	The function returns coordinates of the indicated point of load application. .
◆	SetGeoLimits ( <a href="#">see page 65</a> )	Function sets geometrical limits for loads. Available since version 2.0.
◆	SetPoint ( <a href="#">see page 65</a> )	The function sets the coordinates of the indicated point of load application. .

### II.1.1.6.3.1 GetGeoLimits

#### C++

```
HRESULT GetGeoLimits(IRobotGeoLayer** ret);
```

#### C#

```
public IRobotGeoLayer GetGeoLimits();
```

#### Visual Basic

```
Public Function GetGeoLimits() As IRobotGeoLayer
```

#### Description

Function returns geometrical limits applied to loads. Available since version 2.0.

### II.1.1.6.3.2 GetPoint

#### C++

```
HRESULT GetPoint(int _which_point, double* _x, double* _y, double* _z, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Function GetPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*,  
ByRef _z As double*) As Boolean
```

**Description**

The function returns coordinates of the indicated point of load application. .

**II.1.6.3.3 SetGeoLimits****C++**

```
HRESULT SetGeoLimits(IRobotGeoLayer* _layer_def, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetGeoLimits(IRobotGeoLayer _layer_def);
```

**Visual Basic**

```
Public Function SetGeoLimits(ByRef _layer_def As IRobotGeoLayer) As Boolean
```

**Description**

Function sets geometrical limits for loads. Available since version 2.0.

**II.1.6.3.4 SetPoint****C++**

```
HRESULT SetPoint(int _which_point, double _x, double _y, double _z);
```

**C#**

```
public void SetPoint(int _which_point, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetPoint(_which_point As int, _x As double, _y As double, _z As double)
```

**Description**

The function sets the coordinates of the indicated point of load application. .

**II.1.7 IRobotLoadRecordThermalIn3Points****Class Hierarchy****C++**

```
interface IRobotLoadRecordThermalIn3Points : IRobotLoadRecord;
```

**C#**

```
public interface IRobotLoadRecordThermalIn3Points : IRobotLoadRecord;
```

**Visual Basic**

```
Public Interface IRobotLoadRecordThermalIn3Points
```

**Description**

Thermal load in three points. .

**II.1.7.1 IRobotLoadRecordThermalIn3Points Members**

The following tables list the members exposed by IRobotLoadRecordThermalIn3Points.

## Public Fields

	Name	Description
◆	Description (see page 56)	Load record description .
◆	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
◆	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
◆	Type (see page 57)	Load record type..
◆	Uniqueld (see page 66)	Unique identifier of a load record Available since version 1.7.

## Public Methods

	Name	Description
◆	GetGeoLimits (see page 67)	Function returns an object describing geometrical limitations of the load. If the load is not limited, then an empty reference to the object (NULL) will be returned. Available since version 2.0.
◆	GetPoint (see page 67)	The function sets the coordinates of the indicated point of load application. .
◆	GetValue (see page 58)	The function returns the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
◆	SetGeoLimits (see page 67)	Function sets geometrical limits for loads. Available since version 2.0.
◆	SetPoint (see page 68)	The function sets the coordinates of the indicated point of load application. .
◆	SetValue (see page 58)	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.7.2 IRobotLoadRecordThermalIn3Points Fields

The fields of the IRobotLoadRecordThermalIn3Points class are listed here.

## Public Fields

	Name	Description
◆	Uniqueld (see page 66)	Unique identifier of a load record Available since version 1.7.

### II.1.1.7.2.1 Uniqueld

#### C++

```
HRESULT get_Uniqueld(long*);
```

#### C#

```
public long Uniqueld { get; }
```

#### Visual Basic

```
Public ReadOnly Uniqueld As Long
```

#### Description

Unique identifier of a load record Available since version 1.7.

### II.1.1.7.3 IRobotLoadRecordThermalIn3Points Methods

The methods of the IRobotLoadRecordThermalIn3Points class are listed here.

## Public Methods

	Name	Description
⊕	GetGeoLimits (see page 67)	Function returns an object describing geometrical limitations of the load. If the load is not limited, then an empty reference to the object (NULL) will be returned. Available since version 2.0.
⊕	GetPoint (see page 67)	The function sets the coordinates of the indicated point of load application. .
⊕	SetGeoLimits (see page 67)	Function sets geometrical limits for loads. Available since version 2.0.
⊕	SetPoint (see page 68)	The function sets the coordinates of the indicated point of load application. .

### II.1.1.7.3.1 GetGeoLimits

#### C++

```
HRESULT GetGeoLimits(IRobotGeoLayer** ret);
```

#### C#

```
public IRobotGeoLayer GetGeoLimits();
```

#### Visual Basic

```
Public Function GetGeoLimits() As IRobotGeoLayer
```

#### Description

Function returns an object describing geometrical limitations of the load. If the load is not limited, then an empty reference to the object (NULL) will be returned. Available since version 2.0.

### II.1.1.7.3.2 GetPoint

#### C++

```
HRESULT GetPoint(int _which_point, double* _x, double* _y, double* _z, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As Boolean
```

#### Description

The function sets the coordinates of the indicated point of load application. .

### II.1.1.7.3.3 SetGeoLimits

#### C++

```
HRESULT SetGeoLimits(IRobotGeoLayer* _layer_def, VARIANT_BOOL* ret);
```

#### C#

```
public bool SetGeoLimits(IRobotGeoLayer _layer_def);
```

#### Visual Basic

```
Public Function SetGeoLimits(ByRef _layer_def As IRobotGeoLayer) As Boolean
```

#### Description

Function sets geometrical limits for loads. Available since version 2.0.

### II.1.7.3.4 SetPoint

#### C++

```
HRESULT SetPoint(int _which_point, double _x, double _y, double _z);
```

#### C#

```
public void SetPoint(int _which_point, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetPoint(_which_point As int, _x As double, _y As double, _z As double)
```

#### Description

The function sets the coordinates of the indicated point of load application. .

### II.1.8 IRobotLoadRecordInContour

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecordInContour : IRobotLoadRecord;
```

#### C#

```
public interface IRobotLoadRecordInContour : IRobotLoadRecord;
```

#### Visual Basic

```
Public Interface IRobotLoadRecordInContour
```

#### Description

Planar load on the contour. For a uniform load only the values for the first point of the load should be specified (I\_ICRV\_PX1, I\_ICRV\_PY1 and I\_ICRV\_PZ1) without providing the point coordinates. For a variable load all the load values and coordinates of the appropriate points should be specified applying the SetPoint (see page 71()) function.

### II.1.8.1 IRobotLoadRecordInContour Members

The following tables list the members exposed by IRobotLoadRecordInContour.

#### Public Fields

	Name	Description
❖	Description (see page 56)	Load record description .
❖	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
❖	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
❖	Type (see page 57)	Load record type. .
❖	Uniqueld (see page 69)	Unique identifier of a load record Available since version 1.7.

#### Public Methods

	Name	Description
❖	GetContourPoint (see page 70)	The function gets the coordinates of the indicated point of a contour. .
❖	GetGeoLimits (see page 70)	Function returns an object describing geometrical limitations for a load. If they have not been defined, then an empty reference to the object will be returned (NULL). Available since version 2.0.

	GetPoint ( <a href="#">see page 70</a> )	The function gets the coordinates of a point. If the indicated point has not been defined, the function returns False (zero value). .
	GetValue ( <a href="#">see page 58</a> )	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
	GetVector ( <a href="#">see page 71</a> )	Function takes coordinates of a load projection vector. .
	SetContourPoint ( <a href="#">see page 71</a> )	The function sets the coordinates of the indicated point of a contour. .
	SetGeoLimits ( <a href="#">see page 71</a> )	Function defines geometrical limitations for a load. If the limitations are defined successfully the function returns a value different from zero (True). Available since version 2.0.
	SetPoint ( <a href="#">see page 71</a> )	The function sets the coordinates of the indicated point of loading. Only for variable loads at three points. .
	SetValue ( <a href="#">see page 58</a> )	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
	SetVector ( <a href="#">see page 72</a> )	Function sets vector of load projection.

### II.1.1.8.2 IRobotLoadRecordInContour Fields

The fields of the IRobotLoadRecordInContour class are listed here.

#### Public Fields

	Name	Description
	UniqueId ( <a href="#">see page 69</a> )	Unique identifier of a load record Available since version 1.7.

### II.1.1.8.2.1 UniqueId

#### C++

```
HRESULT get_UniqueId(long*);
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As long
```

#### Description

Unique identifier of a load record Available since version 1.7.

### II.1.1.8.3 IRobotLoadRecordInContour Methods

The methods of the IRobotLoadRecordInContour class are listed here.

#### Public Methods

	Name	Description
	GetContourPoint ( <a href="#">see page 70</a> )	The function gets the coordinates of the indicated point of a contour. .
	GetGeoLimits ( <a href="#">see page 70</a> )	Function returns an object describing geometrical limitations for a load. If they have not been defined, then an empty reference to the object will be returned (NULL). Available since version 2.0.
	GetPoint ( <a href="#">see page 70</a> )	The function gets the coordinates of a point. If the indicated point has not been defined, the function returns False (zero value). .
	GetVector ( <a href="#">see page 71</a> )	Function takes coordinates of a load projection vector. .
	SetContourPoint ( <a href="#">see page 71</a> )	The function sets the coordinates of the indicated point of a contour. .

	SetGeoLimits (see page 71)	Function defines geometrical limitations for a load. If the limitations are defined successfully the function returns a value different from zero (True). Available since version 2.0.
	SetPoint (see page 71)	The function sets the coordinates of the indicated point of loading. Only for variable loads at three points. .
	SetVector (see page 72)	Function sets vector of load projection.

### II.1.1.8.3.1 GetContourPoint

#### C++

```
HRESULT GetContourPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### C#

```
public void GetContourPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Sub GetContourPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

#### Description

The function gets the coordinates of the indicated point of a contour. .

### II.1.1.8.3.2 GetGeoLimits

#### C++

```
HRESULT GetGeoLimits(IRobotGeoLayer** ret);
```

#### C#

```
public IRobotGeoLayer GetGeoLimits();
```

#### Visual Basic

```
Public Function GetGeoLimits() As IRobotGeoLayer
```

#### Description

Function returns an object describing geometrical limitations for a load. If they have not been defined, then an empty reference to the object will be returned (NULL). Available since version 2.0.

### II.1.1.8.3.3 GetPoint

#### C++

```
HRESULT GetPoint(int _which_point, double* _x, double* _y, double* _z, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As Boolean
```

#### Description

The function gets the coordinates of a point. If the indicated point has not been defined, the function returns False (zero value). .

### II.1.1.8.3.4 GetVector

**C++**

```
HRESULT GetVector(double* _vx, double* _vy, double* _vz);
```

**C#**

```
public void GetVector(double* _vx, double* _vy, double* _vz);
```

**Visual Basic**

```
Public Sub GetVector(ByRef _vx As double*, ByRef _vy As double*, ByRef _vz As double*)
```

**Description**

Function takes coordinates of a load projection vector. .

### II.1.1.8.3.5 SetContourPoint

**C++**

```
HRESULT SetContourPoint(int _which_point, double _x, double _y, double _z);
```

**C#**

```
public void SetContourPoint(int _which_point, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetContourPoint(_which_point As int, _x As double, _y As double, _z As double)
```

**Description**

The function sets the coordinates of the indicated point of a contour. .

### II.1.1.8.3.6 SetGeoLimits

**C++**

```
HRESULT SetGeoLimits(IRobotGeoLayer* _layer_def, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetGeoLimits(IRobotGeoLayer _layer_def);
```

**Visual Basic**

```
Public Function SetGeoLimits(ByRef _layer_def As IRobotGeoLayer) As Boolean
```

**Description**

Function defines geometrical limitations for a load. If the limitations are defined successfully the function returns a value different from zero (True). Available since version 2.0.

### II.1.1.8.3.7 SetPoint

**C++**

```
HRESULT SetPoint(int _which_point, double _x, double _y, double _z);
```

**C#**

```
public void SetPoint(int _which_point, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetPoint(_which_point As int, _x As double, _y As double, _z As double)
```

## Description

The function sets the coordinates of the indicated point of loading. Only for variable loads at three points. .

### II.1.8.3.8 SetVector

#### C++

```
HRESULT SetVector(double _vx, double _vy, double _vz);
```

#### C#

```
public void SetVector(double _vx, double _vy, double _vz);
```

#### Visual Basic

```
Public Sub SetVector(_vx As Double, _vy As Double, _vz As Double)
```

#### Description

Function sets vector of load projection.

### II.1.1.9 IRobotLoadRecordLinear

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecordLinear : IRobotLoadRecord;
```

#### C#

```
public interface IRobotLoadRecordLinear : IRobotLoadRecord;
```

#### Visual Basic

```
Public Interface IRobotLoadRecordLinear
```

### II.1.1.9.1 IRobotLoadRecordLinear Members

The following tables list the members exposed by IRobotLoadRecordLinear.

#### Public Fields

	Name	Description
◆	Description (see page 56)	Load record description .
◆	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
◆	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
◆	Type (see page 57)	Load record type. .
◆	UniqueId (see page 73)	Unique identifier of a load record Available since version 1.7.

#### Public Methods

	Name	Description
◆	GetPoint (see page 73)	The function gets the coordinate of the indicated point. .
◆	GetValue (see page 58)	The function returns the value of the indicated quantity describing a load The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
◆	SetPoint (see page 73)	The function sets the coordinate of the indicated point. .

	SetValue (see page 58)	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
-----------------------------------------------------------------------------------	------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## II.1.9.2 IRobotLoadRecordLinear Fields

The fields of the IRobotLoadRecordLinear class are listed here.

### Public Fields

	Name	Description
	UniqueId (see page 73)	Unique identifier of a load record Available since version 1.7.

### II.1.9.2.1 UniqueId

#### C++

```
HRESULT get_UniqueId(long*);
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As long
```

#### Description

Unique identifier of a load record Available since version 1.7.

## II.1.9.3 IRobotLoadRecordLinear Methods

The methods of the IRobotLoadRecordLinear class are listed here.

### Public Methods

	Name	Description
	GetPoint (see page 73)	The function gets the coordinate of the indicated point. .
	SetPoint (see page 73)	The function sets the coordinate of the indicated point. .

### II.1.9.3.1 GetPoint

#### C++

```
HRESULT GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### C#

```
public void GetPoint(int _which_point, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Sub GetPoint(_which_point As int, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

#### Description

The function gets the coordinate of the indicated point. .

### II.1.9.3.2 SetPoint

#### C++

```
HRESULT SetPoint(int _which_point, double _x, double _y, double _z);
```

**C#**

```
public void SetPoint(int _which_point, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetPoint(_which_point As int, _x As double, _y As double, _z As double)
```

**Description**

The function sets the coordinate of the indicated point. .

**II.1.10 IRobotLoadRecord2****Class Hierarchy****C++**

```
interface IRobotLoadRecord2 : IRobotLoadRecord;
```

**C#**

```
public interface IRobotLoadRecord2 : IRobotLoadRecord;
```

**Visual Basic**

```
Public Interface IRobotLoadRecord2
```

**Description**

Extended interface of the record that describes a load. .

**II.1.10.1 IRobotLoadRecord2 Members**

The following tables list the members exposed by IRobotLoadRecord2.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Description (see page 56)	Load record description .
❖	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
❖	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
❖	Type (see page 57)	Load record type. .
❖	Uniqueld (see page 75)	Unique identifier of a load record.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	GetGeoLimits (see page 75)	Function returns an object describing a geometrical limitation defined for this load. If the limitation has not been defined, the function returns an empty reference. Available since version 2.0.
❖	GetValue (see page 58)	The function returns the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
❖	SetGeoLimits (see page 76)	Function defines geometrical limitations for a load. It may happen to be the case that a definition method or load type does not allow defining geometrical limitations for it. The function returns the information if application of limitations has been successful. In order to delete geometrical limitations, an empty reference to the object should be provided as a parameter. Available since version 2.0.

	SetValue ( <a href="#">see page 58</a> )	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
-----------------------------------------------------------------------------------	------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## II.1.10.2 IRobotLoadRecord2 Fields

The fields of the IRobotLoadRecord2 class are listed here.

### Public Fields

	Name	Description
	UniqueId ( <a href="#">see page 75</a> )	Unique identifier of a load record.

## II.1.10.2.1 UniqueId

### C++

```
HRESULT get_UniqueId(long*);
```

### C#

```
public long UniqueId { get; }
```

### Visual Basic

```
Public ReadOnly UniqueId As long
```

### Description

Unique identifier of a load record.

## II.1.10.3 IRobotLoadRecord2 Methods

The methods of the IRobotLoadRecord2 class are listed here.

### Public Methods

	Name	Description
	GetGeoLimits ( <a href="#">see page 75</a> )	Function returns an object describing a geometrical limitation defined for this load. If the limitation has not been defined, the function returns an empty reference. Available since version 2.0.
	SetGeoLimits ( <a href="#">see page 76</a> )	Function defines geometrical limitations for a load. It may happen to be the case that a definition method or load type does not allow defining geometrical limitations for it. The function returns the information if application of limitations has been successful. In order to delete geometrical limitations, an empty reference to the object should be provided as a parameter. Available since version 2.0.

## II.1.10.3.1 GetGeoLimits

### C++

```
HRESULT GetGeoLimits(IRobotGeoLayer** ret);
```

### C#

```
public IRobotGeoLayer GetGeoLimits();
```

### Visual Basic

```
Public Function GetGeoLimits() As IRobotGeoLayer
```

### Description

Function returns an object describing a geometrical limitation defined for this load. If the limitation has not been defined, the function returns an empty reference. Available since version 2.0.

### II.1.10.3.2 SetGeoLimits

**C++**

```
HRESULT SetGeoLimits(IRobotGeoLayer* _layer_def, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetGeoLimits(IRobotGeoLayer _layer_def);
```

**Visual Basic**

```
Public Function SetGeoLimits(ByRef _layer_def As IRobotGeoLayer) As Boolean
```

**Description**

Function defines geometrical limitations for a load. It may happen to be the case that a definition method or load type does not allow defining geometrical limitations for it. The function returns the information if application of limitations has been successful. In order to delete geometrical limitations, an empty reference to the object should be provided as a parameter. Available since version 2.0.

### II.1.11 IRobotLoadRecordBarTrapezoidal

**Class Hierarchy**

**C++**

```
interface IRobotLoadRecordBarTrapezoidal : IRobotLoadRecord2;
```

**C#**

```
public interface IRobotLoadRecordBarTrapezoidal : IRobotLoadRecord2;
```

**Visual Basic**

```
Public Interface IRobotLoadRecordBarTrapezoidal
```

**Description**

Trapezoidal load on bars.

**Version**

Available since version 8.5.

#### II.1.11.1 IRobotLoadRecordBarTrapezoidal Members

The following tables list the members exposed by IRobotLoadRecordBarTrapezoidal.

**Public Fields**

	Name	Description
❖	Description (see page 56)	Load record description .
❖	Objects (see page 57)	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
❖	ObjectType (see page 57)	Type (see page 57) of the object where a load is applied .
❖	PointCount (see page 77)	Number of points defining the load.
❖	Type (see page 57)	Load record type..
❖	Uniqueld (see page 75)	Unique identifier of a load record.

## Public Methods

	Name	Description
💡	GetGeoLimits (🔗 see page 75)	Function returns an object describing a geometrical limitation defined for this load. If the limitation has not been defined, the function returns an empty reference. Available since version 2.0.
💡	GetPoint (🔗 see page 78)	Function returns parameters of the load at the determined point.
💡	GetValue (🔗 see page 58)	The function returns the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
💡	SetGeoLimits (🔗 see page 76)	Function defines geometrical limitations for a load. It may happen to be the case that a definition method or load type does not allow defining geometrical limitations for it. The function returns the information if application of limitations has been successful. In order to delete geometrical limitations, an empty reference to the object should be provided as a parameter. Available since version 2.0.
💡	SetPoint (🔗 see page 78)	Function sets parameters of the load at the determined point.
💡	SetValue (🔗 see page 58)	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.11.2 IRobotLoadRecordBarTrapezoidal Fields

The fields of the IRobotLoadRecordBarTrapezoidal class are listed here.

#### Public Fields

	Name	Description
💡	PointCount (🔗 see page 77)	Number of points defining the load.

### II.1.11.2.1 PointCount

#### C++

```
HRESULT get_PointCount(long* );
HRESULT put_PointCount(long);
```

#### C#

```
public long PointCount { get; set; }
```

#### Visual Basic

```
Public PointCount As long
```

#### Description

Number of points defining the load.

#### Version

Available since version 8.5.

### II.1.11.3 IRobotLoadRecordBarTrapezoidal Methods

The methods of the IRobotLoadRecordBarTrapezoidal class are listed here.

#### Public Methods

	Name	Description
💡	GetPoint (🔗 see page 78)	Function returns parameters of the load at the determined point.

	SetPoint (see page 78)	Function sets parameters of the load at the determined point.
-----------------------------------------------------------------------------------	------------------------	---------------------------------------------------------------

### II.1.11.3.1 GetPoint

#### C++

```
HRESULT GetPoint(long _pt_idx, double* _px, double* _py, double* _pz, double* _x);
```

#### C#

```
public void GetPoint(long _pt_idx, double* _px, double* _py, double* _pz, double* _x);
```

#### Visual Basic

```
Public Sub GetPoint(_pt_idx As long, ByRef _px As double*, ByRef _py As double*, ByRef _pz As double*, ByRef _x As double*)
```

#### Description

Function returns parameters of the load at the determined point.

#### Version

Available since version 8.5.

### II.1.11.3.2 SetPoint

#### C++

```
HRESULT SetPoint(long _pt_idx, double _px, double _py, double _pz, double _x);
```

#### C#

```
public void SetPoint(long _pt_idx, double _px, double _py, double _pz, double _x);
```

#### Visual Basic

```
Public Sub SetPoint(_pt_idx As long, _px As double, _py As double, _pz As double, _x As double)
```

#### Description

Function sets parameters of the load at the determined point.

#### Version

Available since version 8.5.

### II.1.12 IRobotLoadRecordDead

#### Class Hierarchy

#### C++

```
interface IRobotLoadRecordDead : IRobotLoadRecord2;
```

#### C#

```
public interface IRobotLoadRecordDead : IRobotLoadRecord2;
```

#### Visual Basic

```
Public Interface IRobotLoadRecordDead
```

#### Description

Load record describing self-weight.

#### Version

Available since version 9.2.

### II.1.1.12.1 IRobotLoadRecordDead Members

The following tables list the members exposed by IRobotLoadRecordDead.

#### Public Fields

	Name	Description
◆	Description ( [ see page 56) )	Load record description .
◆	FiniteElems ( [ see page 79) )	Text describing the selection of finite elements to which the load is applied.
◆	Objects ( [ see page 57) )	Selection of objects - structure components where a load is applied. The object type described by the selection corresponds to the type of objects to which the load may be applied. .
◆	ObjectType ( [ see page 57) )	Type ( [ see page 57) ) of the object where a load is applied .
◆	Type ( [ see page 57) )	Load record type..
◆	UniqueId ( [ see page 75) )	Unique identifier of a load record.

#### Public Methods

	Name	Description
≡◆	GetGeoLimits ( [ see page 75) )	Function returns an object describing a geometrical limitation defined for this load. If the limitation has not been defined, the function returns an empty reference. Available since version 2.0.
≡◆	GetValue ( [ see page 58) )	The function returns the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .
≡◆	SetGeoLimits ( [ see page 76) )	Function defines geometrical limitations for a load. It may happen to be the case that a definition method or load type does not allow defining geometrical limitations for it. The function returns the information if application of limitations has been successful. In order to delete geometrical limitations, an empty reference to the object should be provided as a parameter. Available since version 2.0.
≡◆	SetValue ( [ see page 58) )	The function sets the value of the indicated quantity describing a load. The definition of a load record consists of a set of numbers - values describing the modeled load. The type and significance of particular values depend on the load record type and they are referred to by an appropriate identifier. .

### II.1.1.12.2 IRobotLoadRecordDead Fields

The fields of the IRobotLoadRecordDead class are listed here.

#### Public Fields

	Name	Description
◆	FiniteElems ( [ see page 79) )	Text describing the selection of finite elements to which the load is applied.

### II.1.1.12.2.1 FiniteElems

#### C++

```
HRESULT get_FiniteElems(BSTR* );
HRESULT put_FiniteElems(BSTR);
```

#### C#

```
public String FiniteElems { get; set; }
```

#### Visual Basic

```
Public FiniteElems As String
```

**Description**

Text describing the selection of finite elements to which the load is applied.

**Version**

Available since version 9.2.

**II.1.1.13 IRobotLoadRecordCommon****Class Hierarchy****C++**

```
interface IRobotLoadRecordCommon : IDispatch;
```

**C#**

```
public interface IRobotLoadRecordCommon;
```

**Visual Basic**

```
Public Interface IRobotLoadRecordCommon
```

**Description**

The interface providing access to the extended set of common attributes for load records of different types.

**Version**

Available since version 12.

**II.1.1.13.1 IRobotLoadRecordCommon Members**

The following tables list the members exposed by IRobotLoadRecordCommon.

**Public Fields**

	Name	Description
◆	IsAutoGenerated (see page 80)	The flag indicating that a load record has been generated automatically.

**II.1.1.13.2 IRobotLoadRecordCommon Fields**

The fields of the IRobotLoadRecordCommon class are listed here.

**Public Fields**

	Name	Description
◆	IsAutoGenerated (see page 80)	The flag indicating that a load record has been generated automatically.

**II.1.1.13.2.1 IsAutoGenerated****C++**

```
HRESULT get_IsAutoGenerated(VARIANT_BOOL* );
```

**C#**

```
public bool IsAutoGenerated { get; }
```

**Visual Basic**

```
Public ReadOnly IsAutoGenerated As Boolean
```

**Description**

The flag indicating that a load record has been generated automatically.

**Version**

Available since version 12.

## II.1.2 Snow/wind loads

### Enumerations

	Name	Description
☞	IRobotSWCodePLWindZone (☞ see page 108)	
☞	IRobotSWCodePLWindSite (☞ see page 108)	
☞	IRobotSWCodePLSnowZone (☞ see page 109)	
☞	IRobotSWCodeFRWindSite (☞ see page 128)	
☞	IRobotSWCodeFRWindType (☞ see page 128)	
☞	IRobotSWCodeFRSurfaceType (☞ see page 129)	
☞	IRobotSWCodeFRSnowType (☞ see page 129)	
☞	IRobotSWCodePLWindPressDistribType (☞ see page 129)	
☞	IRobotSWCodeECSiteType (☞ see page 146)	
☞	IRobotSWCodeECGroundType (☞ see page 146)	
☞	IRobotSWCodeECCdType (☞ see page 147)	

### Interfaces

	Name	Description
☞	IRobotSnowWindParams (☞ see page 92)	Set of the non-code-specific parameters describing the manner of generating snow/wind loads. .
☞	IRobotSnowWindEngine (☞ see page 95)	Generator of snow/wind loads.
☞	IRobotSWCodePLParams (☞ see page 98)	Set of parameters for snow/wind loads for Polish code.
☞	IRobotSWCodeFRParams (☞ see page 109)	Parameters of snow/wind load generation for the French code.
☞	IRobotSWCodeECPParams (☞ see page 130)	Parameters of snow/wind load generation for Eurocode. .

### II.1.2.1 Generating 3D loads

#### Interfaces

	Name	Description
☞	IRobotSWStruct3D (☞ see page 82)	Structure model defined for generation of 3D snow/wind loads. The model describes a 3D frame generated by copying n times a 2D frame in the third direction. All structure frames consist of the same number of elements. Corresponding elements in individual frames have the same indexes. If an element occurs only in gable frames, then the bar number (Bar) for the corresponding elements of internal frames is ignored.
☞	IRobotSWStruct3DElement (☞ see page 84)	Element of a 3D structure model defined for generation of snow/wind loads.
☞	IRobotSWStruct3DFrame (☞ see page 85)	Definition of a 2D frame, a component of a 3D frame, that models a structure for the needs of generation of snow/wind loads. The frame elements are indexed with values from the interval [1, ElemCount (☞ see page 86)].

	IRobotSWStruct3DGenParams (see page 87)	Parameters of generation of a 3D frame structure model for the needs of definition of moving loads.
	IRobotSWStruct3DPurlinGenParams (see page 90)	Definition of the method of generation of longitudinal bars (purlins) adjoining to the frame bar.

## II.1.2.1.1 IRobotSWStruct3D

### Class Hierarchy

#### C++

```
interface IRobotSWStruct3D : IDispatch;
```

#### C#

```
public interface IRobotSWStruct3D;
```

### Visual Basic

```
Public Interface IRobotSWStruct3D
```

### Description

Structure model defined for generation of 3D snow/wind loads. The model describes a 3D frame generated by copying n times a 2D frame in the third direction. All structure frames consist of the same number of elements. Corresponding elements in individual frames have the same indexes. If an element occurs only in gable frames, then the bar number (Bar) for the corresponding elements of internal frames is ignored.

## II.1.2.1.1.1 IRobotSWStruct3D Members

The following tables list the members exposed by IRobotSWStruct3D.

### Public Fields

	Name	Description
	FrameCount (see page 82)	Number of 2D frames forming a 3D frame.
	FrameElemCount (see page 83)	Number of elements of a single frame.

### Public Methods

	Name	Description
	GetFrame (see page 83)	The function returns the frame with the specified index. The index of the first (front) frame equals 1, whereas the index of the last (rear) frame equals FrameCount (see page 82).
	SetFrame (see page 83)	The function defines the frame with the specified index. Frame indexes are values from the interval [1, FrameCount (see page 82)].

## II.1.2.1.1.2 IRobotSWStruct3D Fields

The fields of the IRobotSWStruct3D class are listed here.

### Public Fields

	Name	Description
	FrameCount (see page 82)	Number of 2D frames forming a 3D frame.
	FrameElemCount (see page 83)	Number of elements of a single frame.

## II.1.2.1.1.2.1 FrameCount

#### C++

```
HRESULT get_FrameCount(long* );
HRESULT put_FrameCount(long);
```

**C#**

```
public long FrameCount { get; set; }
```

**Visual Basic**

```
Public FrameCount As Long
```

**Description**

Number of 2D frames forming a 3D frame.

**II.1.2.1.1.2.2 FrameElemCount****C++**

```
HRESULT get_FrameElemCount(long* );
```

**C#**

```
public long FrameElemCount { get; }
```

**Visual Basic**

```
Public ReadOnly FrameElemCount As Long
```

**Description**

Number of elements of a single frame.

**II.1.2.1.1.3 IRobotSWStruct3D Methods**

The methods of the IRobotSWStruct3D class are listed here.

**Public Methods**

	Name	Description
✳	GetFrame ( [ see page 83) ]	The function returns the frame with the specified index. The index of the first (front) frame equals 1, whereas the index of the last (rear) frame equals FrameCount ( [ see page 82) ].
✳	SetFrame ( [ see page 83) ]	The function defines the frame with the specified index. Frame indexes are values from the interval [1, FrameCount ( [ see page 82) ]].

**II.1.2.1.1.3.1 GetFrame****C++**

```
HRESULT GetFrame(long _frame_idx, IRobotSWStruct3DFrame** ret);
```

**C#**

```
public IRobotSWStruct3DFrame GetFrame(long _frame_idx);
```

**Visual Basic**

```
Public Function GetFrame(_frame_idx As Long) As IRobotSWStruct3DFrame
```

**Description**

The function returns the frame with the specified index. The index of the first (front) frame equals 1, whereas the index of the last (rear) frame equals FrameCount ( [ see page 82) ].

**II.1.2.1.1.3.2 SetFrame****C++**

```
HRESULT SetFrame(long _frame_idx, IRobotSWStruct3DFrame* _frame);
```

**C#**

```
public void SetFrame(long _frame_idx, IRobotSWStruct3DFrame _frame);
```

**Visual Basic**

```
Public Sub SetFrame(_frame_idx As long, ByRef _frame As IRobotSWStruct3DFrame)
```

**Description**

The function defines the frame with the specified index. Frame indexes are values from the interval [1, FrameCount (see page 82)].

**II.1.2.1.2 IRobotSWStruct3DElement****Class Hierarchy****C++**

```
interface IRobotSWStruct3DElement : IDispatch;
```

**C#**

```
public interface IRobotSWStruct3DElement;
```

**Visual Basic**

```
Public Interface IRobotSWStruct3DElement
```

**Description**

Element of a 3D structure model defined for generation of snow/wind loads.

**II.1.2.1.2.1 IRobotSWStruct3DElement Members**

The following tables list the members exposed by IRobotSWStruct3DElement.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Bar (see page 84)	Bar number.
❖	IsFacadeLoaded (see page 85)	Flag forcing load generation in the gable frame for a bar.
❖	IsFacadeOnly (see page 85)	Flag indicating that a bar occurs only in a gable frame (front or rear one).
❖	Purlins (see page 85)	Table with numbers of bars that define purlins adjoining to this bar.

**II.1.2.1.2.2 IRobotSWStruct3DElement Fields**

The fields of the IRobotSWStruct3DElement class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Bar (see page 84)	Bar number.
❖	IsFacadeLoaded (see page 85)	Flag forcing load generation in the gable frame for a bar.
❖	IsFacadeOnly (see page 85)	Flag indicating that a bar occurs only in a gable frame (front or rear one).
❖	Purlins (see page 85)	Table with numbers of bars that define purlins adjoining to this bar.

**II.1.2.1.2.2.1 Bar****C++**

```
HRESULT get_Bar(long* );
HRESULT put_Bar(long);
```

**C#**

```
public long Bar { get; set; }
```

**Visual Basic**

```
Public Bar As long
```

**Description**

Bar number.

**II.1.2.1.2.2.2 IsFacadeLoaded****C++**

```
HRESULT get_IsFacadeLoaded(VARIANT_BOOL* );
HRESULT put_IsFacadeLoaded(VARIANT_BOOL);
```

**C#**

```
public bool IsFacadeLoaded { get; set; }
```

**Visual Basic**

```
Public IsFacadeLoaded As Boolean
```

**Description**

Flag forcing load generation in the gable frame for a bar.

**II.1.2.1.2.2.3 IsFacadeOnly****C++**

```
HRESULT get_IsFacadeOnly(VARIANT_BOOL* );
HRESULT put_IsFacadeOnly(VARIANT_BOOL);
```

**C#**

```
public bool IsFacadeOnly { get; set; }
```

**Visual Basic**

```
Public IsFacadeOnly As Boolean
```

**Description**

Flag indicating that a bar occurs only in a gable frame (front or rear one).

**II.1.2.1.2.2.4 Purlins****C++**

```
HRESULT get_Purlins(IRobotNumbersArray** );
```

**C#**

```
public IRobotNumbersArray Purlins { get; }
```

**Visual Basic**

```
Public ReadOnly Purlins As IRobotNumbersArray
```

**Description**

Table with numbers of bars that define purlins adjoining to this bar.

**II.1.2.1.3 IRobotSWStruct3DFrame****Class Hierarchy**

**C++**

```
interface IRobotSWStruct3DFrame : IDispatch;
```

**C#**

```
public interface IRobotSWStruct3DFrame;
```

**Visual Basic**

```
Public Interface IRobotSWStruct3DFrame
```

**Description**

Definition of a 2D frame, a component of a 3D frame, that models a structure for the needs of generation of snow/wind loads. The frame elements are indexed with values from the interval [1, ElemCount ([see page 86](#))].

**II.1.2.1.3.1 IRobotSWStruct3DFrame Members**

The following tables list the members exposed by IRobotSWStruct3DFrame.

**Public Fields**

	<b>Name</b>	<b>Description</b>
	ElemCount ( <a href="#">see page 86</a> )	Number of elements defining a frame.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetElem ( <a href="#">see page 87</a> )	The function returns the frame element with the specified index.
	SetElem ( <a href="#">see page 87</a> )	The function positions the specified frame element.

**II.1.2.1.3.2 IRobotSWStruct3DFrame Fields**

The fields of the IRobotSWStruct3DFrame class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
	ElemCount ( <a href="#">see page 86</a> )	Number of elements defining a frame.

**II.1.2.1.3.2.1 ElemCount****C++**

```
HRESULT get_ElemCount(long*);
```

**C#**

```
public long ElemCount { get; }
```

**Visual Basic**

```
Public ReadOnly ElemCount As long
```

**Description**

Number of elements defining a frame.

**II.1.2.1.3.3 IRobotSWStruct3DFrame Methods**

The methods of the IRobotSWStruct3DFrame class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetElem ( <a href="#">see page 87</a> )	The function returns the frame element with the specified index.

	SetElem ( <a href="#">see page 87</a> )	The function positions the specified frame element.
-----------------------------------------------------------------------------------	-----------------------------------------	-----------------------------------------------------

### II.1.2.1.3.3.1 GetElem

#### C++

```
HRESULT GetElem(long _elem_idx, IRobotSWStruct3DElement** ret);
```

#### C#

```
public IRobotSWStruct3DElement GetElem(long _elem_idx);
```

#### Visual Basic

```
Public Function GetElem(_elem_idx As long) As IRobotSWStruct3DElement
```

#### Description

The function returns the frame element with the specified index.

### II.1.2.1.3.3.2 SetElem

#### C++

```
HRESULT SetElem(long _elem_idx, IRobotSWStruct3DElement* _elem);
```

#### C#

```
public void SetElem(long _elem_idx, IRobotSWStruct3DElement _elem);
```

#### Visual Basic

```
Public Sub SetElem(_elem_idx As long, ByRef _elem As IRobotSWStruct3DElement)
```

#### Description

The function positions the specified frame element.

### II.1.2.1.4 IRobotSWStruct3DGenParams

#### Class Hierarchy

#### C++

```
interface IRobotSWStruct3DGenParams : IDispatch;
```

#### C#

```
public interface IRobotSWStruct3DGenParams;
```

#### Visual Basic

```
Public Interface IRobotSWStruct3DGenParams
```

#### Description

Parameters of generation of a 3D frame structure model for the needs of definition of moving loads.

### II.1.2.1.4.1 IRobotSWStruct3DGenParams Members

The following tables list the members exposed by IRobotSWStruct3DGenParams.

#### Public Fields

	Name	Description
	Bars ( <a href="#">see page 88</a> )	Selection of bars forming the initial 2D frame.
	FacadeLoadedBars ( <a href="#">see page 88</a> )	Selection of the gable frame bars for which generation of snow/wind loads will be forced.

❖	FacadeOnlyBars (see page 89)	Selection of bars to be copied only in gable walls.
❖	FrameCount (see page 89)	Ultimate number of 2D frames.
❖	Offsets (see page 89)	
❖	Spacings (see page 89)	Table including distances (in meters) between successive frames; If only one value (with the index 1) is defined in the table, then all frames will be spaced according to that value.

## Public Methods

	Name	Description
❖	GetPurlins (see page 90)	The function returns parameters of generation of longitudinal bars for the indicated frame bar.
❖	SetPurlins (see page 90)	The function defines the method of generation of longitudinal bars for the indicated frame bar.

### II.1.2.1.4.2 IRobotSWStruct3DGenParams Fields

The fields of the IRobotSWStruct3DGenParams class are listed here.

## Public Fields

	Name	Description
❖	Bars (see page 88)	Selection of bars forming the initial 2D frame.
❖	FacadeLoadedBars (see page 88)	Selection of the gable frame bars for which generation of snow/wind loads will be forced.
❖	FacadeOnlyBars (see page 89)	Selection of bars to be copied only in gable walls.
❖	FrameCount (see page 89)	Ultimate number of 2D frames.
❖	Offsets (see page 89)	
❖	Spacings (see page 89)	Table including distances (in meters) between successive frames; If only one value (with the index 1) is defined in the table, then all frames will be spaced according to that value.

### II.1.2.1.4.2.1 Bars

#### C++

```
HRESULT get_Bars(BSTR*) ;
HRESULT put_Bars(BSTR) ;
```

#### C#

```
public String Bars { get; set; }
```

#### Visual Basic

```
Public Bars As String
```

#### Description

Selection of bars forming the initial 2D frame.

### II.1.2.1.4.2.2 FacadeLoadedBars

#### C++

```
HRESULT get_FacadeLoadedBars(BSTR*) ;
HRESULT put_FacadeLoadedBars(BSTR) ;
```

#### C#

```
public String FacadeLoadedBars { get; set; }
```

#### Visual Basic

```
Public FacadeLoadedBars As String
```

**Description**

Selection of the gable frame bars for which generation of snow/wind loads will be forced.

**II.1.2.1.4.2.3 FacadeOnlyBars****C++**

```
HRESULT get_FacadeOnlyBars(BSTR* );
HRESULT put_FacadeOnlyBars(BSTR);
```

**C#**

```
public String FacadeOnlyBars { get; set; }
```

**Visual Basic**

```
Public FacadeOnlyBars As String
```

**Description**

Selection of bars to be copied only in gable walls.

**II.1.2.1.4.2.4 FrameCount****C++**

```
HRESULT get_FrameCount(long* );
HRESULT put_FrameCount(long);
```

**C#**

```
public long FrameCount { get; set; }
```

**Visual Basic**

```
Public FrameCount As long
```

**Description**

Ultimate number of 2D frames.

**II.1.2.1.4.2.5 Offsets****C++**

```
HRESULT get_Offsets(VARIANT_BOOL* );
HRESULT put_Offsets(VARIANT_BOOL);
```

**C#**

```
public bool Offsets { get; set; }
```

**Visual Basic**

```
Public Offsets As Boolean
```

**II.1.2.1.4.2.6 Spacings****C++**

```
HRESULT get_Spacings(IRobotValuesArray** );
```

**C#**

```
public IRobotValuesArray Spacings { get; }
```

**Visual Basic**

```
Public ReadOnly Spacings As IRobotValuesArray
```

## Description

Table including distances (in meters) between successive frames; If only one value (with the index 1) is defined in the table, then all frames will be spaced according to that value.

### II.1.2.1.4.3 IRobotSWStruct3DGenParams Methods

The methods of the IRobotSWStruct3DGenParams class are listed here.

#### Public Methods

	Name	Description
💡	GetPurlins (see page 90)	The function returns parameters of generation of longitudinal bars for the indicated frame bar.
💡	SetPurlins (see page 90)	The function defines the method of generation of longitudinal bars for the indicated frame bar.

#### II.1.2.1.4.3.1 GetPurlins

##### C++

```
HRESULT GetPurlins(long _bar_num, IRobotSWStruct3DPurlinGenParams** ret);
```

##### C#

```
public IRobotSWStruct3DPurlinGenParams GetPurlins(long _bar_num);
```

##### Visual Basic

```
Public Function GetPurlins(_bar_num As long) As IRobotSWStruct3DPurlinGenParams
```

#### Description

The function returns parameters of generation of longitudinal bars for the indicated frame bar.

#### II.1.2.1.4.3.2 SetPurlins

##### C++

```
HRESULT SetPurlins(long _bar_num, IRobotSWStruct3DPurlinGenParams* _purlins_def);
```

##### C#

```
public void SetPurlins(long _bar_num, IRobotSWStruct3DPurlinGenParams _purlins_def);
```

##### Visual Basic

```
Public Sub SetPurlins(_bar_num As long, ByRef _purlins_def As IRobotSWStruct3DPurlinGenParams)
```

#### Description

The function defines the method of generation of longitudinal bars for the indicated frame bar.

### II.1.2.1.5 IRobotSWStruct3DPurlinGenParams

#### Class Hierarchy

##### C++

```
interface IRobotSWStruct3DPurlinGenParams : IDispatch;
```

##### C#

```
public interface IRobotSWStruct3DPurlinGenParams;
```

**Visual Basic**

```
Public Interface IRobotSWStruct3DPurlinGenParams
```

**Description**

Definition of the method of generation of longitudinal bars (purlins) adjoining to the frame bar.

**II.1.2.1.5.1 IRobotSWStruct3DPurlinGenParams Members**

The following tables list the members exposed by IRobotSWStruct3DPurlinGenParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Locations ( <a href="#">see page 91</a> )	Table defining successive points at which purlins should adjoin to the bar.
❖	RelativeLocations ( <a href="#">see page 91</a> )	Flag indicating that location of successive purlins along the bar has been expressed in values relative to the beginning and length of the bar.
❖	SectionName ( <a href="#">see page 92</a> )	Name of the section that will be assigned to bars-purlins.

**II.1.2.1.5.2 IRobotSWStruct3DPurlinGenParams Fields**

The fields of the IRobotSWStruct3DPurlinGenParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Locations ( <a href="#">see page 91</a> )	Table defining successive points at which purlins should adjoin to the bar.
❖	RelativeLocations ( <a href="#">see page 91</a> )	Flag indicating that location of successive purlins along the bar has been expressed in values relative to the beginning and length of the bar.
❖	SectionName ( <a href="#">see page 92</a> )	Name of the section that will be assigned to bars-purlins.

**II.1.2.1.5.2.1 Locations****C++**

```
HRESULT get_Locations(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray Locations { get; }
```

**Visual Basic**

```
Public ReadOnly Locations As IRobotValuesArray
```

**Description**

Table defining successive points at which purlins should adjoin to the bar.

**II.1.2.1.5.2.2 RelativeLocations****C++**

```
HRESULT get_RelativeLocations(VARIANT_BOOL*);  
HRESULT put_RelativeLocations(VARIANT_BOOL);
```

**C#**

```
public bool RelativeLocations { get; set; }
```

**Visual Basic**

```
Public RelativeLocations As Boolean
```

**Description**

Flag indicating that location of successive purlins along the bar has been expressed in values relative to the beginning and length of the bar.

### II.1.2.1.5.2.3 SectionName

#### C++

```
HRESULT get_SectionName(BSTR* );
HRESULT put_SectionName(BSTR);
```

#### C#

```
public String SectionName { get; set; }
```

#### Visual Basic

```
Public SectionName As String
```

#### Description

Name of the section that will be assigned to bars-purlins.

### II.1.2.2 IRobotSnowWindParams

#### Class Hierarchy

#### C++

```
interface IRobotSnowWindParams : IDispatch;
```

#### C#

```
public interface IRobotSnowWindParams;
```

#### Visual Basic

```
Public Interface IRobotSnowWindParams
```

#### Description

Set of the non-code-specific parameters describing the manner of generating snow/wind loads. .

### II.1.2.2.1 IRobotSnowWindParams Members

The following tables list the members exposed by IRobotSnowWindParams.

#### Public Fields

	Name	Description
◆	BaseOnGround (↗ see page 93)	Available since version 1.7.
◆	BaySpacing (↗ see page 93)	Available since version 1.7.
◆	Envelope (↗ see page 93)	Available since version 1.7.
◆	IsolatedRoofs (↗ see page 93)	Available since version 1.7.
◆	IsSnow (↗ see page 94)	Available since version 1.7.
◆	IsWind (↗ see page 94)	Available since version 1.7.
◆	TotalDepth (↗ see page 94)	Available since version 1.7.
◆	WithCavities (↗ see page 95)	Available since version 1.7.
◆	WithParapets (↗ see page 95)	Available since version 1.7.

### II.1.2.2.2 IRobotSnowWindParams Fields

The fields of the IRobotSnowWindParams class are listed here.

#### Public Fields

	Name	Description
◆	BaseOnGround (↗ see page 93)	Available since version 1.7.
◆	BaySpacing (↗ see page 93)	Available since version 1.7.

◆	Envelope (see page 93)	Available since version 1.7.
◆	IsolatedRoofs (see page 93)	Available since version 1.7.
◆	IsSnow (see page 94)	Available since version 1.7.
◆	IsWind (see page 94)	Available since version 1.7.
◆	TotalDepth (see page 94)	Available since version 1.7.
◆	WithCavities (see page 95)	Available since version 1.7.
◆	WithParapets (see page 95)	Available since version 1.7.

### II.1.2.2.2.1 BaseOnGround

**C++**

```
HRESULT get_BaseOnGround(VARIANT_BOOL* );
HRESULT put_BaseOnGround(VARIANT_BOOL);
```

**C#**

```
public bool BaseOnGround { get; set; }
```

**Visual Basic**

```
Public BaseOnGround As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.2.2.2 BaySpacing

**C++**

```
HRESULT get_BaySpacing(double* );
HRESULT put_BaySpacing(double);
```

**C#**

```
public double BaySpacing { get; set; }
```

**Visual Basic**

```
Public BaySpacing As Double
```

**Description**

Available since version 1.7.

### II.1.2.2.2.3 Envelope

**C++**

```
HRESULT get_Envelope(BSTR* );
HRESULT put_Envelope(BSTR);
```

**C#**

```
public String Envelope { get; set; }
```

**Visual Basic**

```
Public Envelope As String
```

**Description**

Available since version 1.7.

### II.1.2.2.2.4 IsolatedRoofs

**C++**

```
HRESULT get_IsolatedRoofs(VARIANT_BOOL* );
```

```
HRESULT put_IsolatedRoofs(VARIANT_BOOL);
```

**C#**

```
public bool IsolatedRoofs { get; set; }
```

**Visual Basic**

```
Public IsolatedRoofs As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.2.2.5 IsSnow****C++**

```
HRESULT get_IsSnow(VARIANT_BOOL*);  
HRESULT put_IsSnow(VARIANT_BOOL);
```

**C#**

```
public bool IsSnow { get; set; }
```

**Visual Basic**

```
Public IsSnow As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.2.2.6 IsWind****C++**

```
HRESULT get_IsWind(VARIANT_BOOL*);  
HRESULT put_IsWind(VARIANT_BOOL);
```

**C#**

```
public bool IsWind { get; set; }
```

**Visual Basic**

```
Public IsWind As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.2.2.7 TotalDepth****C++**

```
HRESULT get_TotalDepth(double*);  
HRESULT put_TotalDepth(double);
```

**C#**

```
public double TotalDepth { get; set; }
```

**Visual Basic**

```
Public TotalDepth As double
```

**Description**

Available since version 1.7.

### II.1.2.2.8 WithCavities

#### C++

```
HRESULT get_WithCavities(VARIANT_BOOL* );
HRESULT put_WithCavities(VARIANT_BOOL);
```

#### C#

```
public bool WithCavities { get; set; }
```

#### Visual Basic

```
Public WithCavities As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.2.9 WithParapets

#### C++

```
HRESULT get_WithParapets(VARIANT_BOOL* );
HRESULT put_WithParapets(VARIANT_BOOL);
```

#### C#

```
public bool WithParapets { get; set; }
```

#### Visual Basic

```
Public WithParapets As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.3 IRobotSnowWindEngine

#### Class Hierarchy

#### C++

```
interface IRobotSnowWindEngine : IDispatch;
```

#### C#

```
public interface IRobotSnowWindEngine;
```

#### Visual Basic

```
Public Interface IRobotSnowWindEngine
```

#### Description

Generator of snow/wind loads.

### II.1.2.3.1 IRobotSnowWindEngine Members

The following tables list the members exposed by IRobotSnowWindEngine.

#### Public Fields

	Name	Description
◆	CodeParams (see page 96)	Available since version 1.7.
◆	Params (see page 96)	Parameters of snow/wind generation (code independent) Available since version 1.7.

## Public Methods

	Name	Description
💡	Generate (🔗 see page 97)	Function generates snow/wind loads on the basis of parameters set earlier. Function returns the collection of numbers of generated load cases. Available since version 1.7.
💡	Generate3D (🔗 see page 97)	The function generates snow/wind loads for the given 3D frame model. The function returns a collection of numbers of generated load cases.
💡	GenerateStruct3D (🔗 see page 97)	The function generates a 3D frame model based on defined parameters.
💡	ShowParamsDlg (🔗 see page 98)	Function opens a dialog for the currently selected snow and wind code. Function returns a value originally transferred through the dialog to the function responsible for dialog closeing (EndDialog).

### II.1.2.3.2 IRobotSnowWindEngine Fields

The fields of the IRobotSnowWindEngine class are listed here.

#### Public Fields

	Name	Description
❖	CodeParams (🔗 see page 96)	Available since version 1.7.
❖	Params (🔗 see page 96)	Parameters of snow/wind generation (code independent) Available since version 1.7.

#### II.1.2.3.2.1 CodeParams

##### C++

```
HRESULT get_CodeParams(IDispatch*);
```

##### C#

```
public IDispatch CodeParams { get; }
```

##### Visual Basic

```
Public ReadOnly CodeParams As IDispatch
```

#### Description

Available since version 1.7.

#### II.1.2.3.2.2 Params

##### C++

```
HRESULT get_Params(IRobotSnowWindParams**);
```

##### C#

```
public IRobotSnowWindParams Params { get; }
```

##### Visual Basic

```
Public ReadOnly Params As IRobotSnowWindParams
```

#### Description

Parameters of snow/wind generation (code independent) Available since version 1.7.

### II.1.2.3.3 IRobotSnowWindEngine Methods

The methods of the IRobotSnowWindEngine class are listed here.

## Public Methods

	Name	Description
💡	Generate (🔗 see page 97)	Function generates snow/wind loads on the basis of parameters set earlier. Function returns the collection of numbers of generated load cases. Available since version 1.7.
💡	Generate3D (🔗 see page 97)	The function generates snow/wind loads for the given 3D frame model. The function returns a collection of numbers of generated load cases.
💡	GenerateStruct3D (🔗 see page 97)	The function generates a 3D frame model based on defined parameters.
💡	ShowParamsDlg (🔗 see page 98)	Function opens a dialog for the currently selected snow and wind code. Function returns a value originally transferred through the dialog to the function responsible for dialog closeing (EndDialog).

### II.1.2.3.3.1 Generate

#### C++

```
HRESULT Generate(IRobotNumbersCollection** ret);
```

#### C#

```
public IRobotNumbersCollection Generate();
```

#### Visual Basic

```
Public Function Generate() As IRobotNumbersCollection
```

#### Description

Function generates snow/wind loads on the basis of parameters set earlier. Function returns the collection of numbers of generated load cases. Available since version 1.7.

### II.1.2.3.3.2 Generate3D

#### C++

```
HRESULT Generate3D(IRobotSWStruct3D* _struct_3d, IRobotNumbersCollection** ret);
```

#### C#

```
public IRobotNumbersCollection Generate3D(IRobotSWStruct3D _struct_3d);
```

#### Visual Basic

```
Public Function Generate3D(ByRef _struct_3d As IRobotSWStruct3D) As IRobotNumbersCollection
```

#### Description

The function generates snow/wind loads for the given 3D frame model. The function returns a collection of numbers of generated load cases.

### II.1.2.3.3.3 GenerateStruct3D

#### C++

```
HRESULT GenerateStruct3D(IRobotSWStruct3DGenParams* _gen_params, IRobotSWStruct3D** ret);
```

#### C#

```
public IRobotSWStruct3D GenerateStruct3D(IRobotSWStruct3DGenParams _gen_params);
```

#### Visual Basic

```
Public Function GenerateStruct3D(ByRef _gen_params As IRobotSWStruct3DGenParams) As IRobotSWStruct3D
```

## Description

The function generates a 3D frame model based on defined parameters.

### II.1.2.3.3.4 ShowParamsDlg

#### C++

```
HRESULT ShowParamsDlg(long* ret);
```

#### C#

```
public long ShowParamsDlg();
```

#### Visual Basic

```
Public Function ShowParamsDlg() As long
```

#### Description

Function opens a dialog for the currently selected snow and wind code. Function returns a value originally transferred through the dialog to the function responsible for dialog closeing (EndDialog).

#### Version

Available since version 11.

### II.1.2.4 IRobotSWCodePLParams

#### Class Hierarchy

#### C++

```
interface IRobotSWCodePLParams : IDispatch;
```

#### C#

```
public interface IRobotSWCodePLParams;
```

#### Visual Basic

```
Public Interface IRobotSWCodePLParams
```

#### Description

Set of parameters for snow/wind loads for Polish code.

### II.1.2.4.1 IRobotSWCodePLParams Members

The following tables list the members exposed by IRobotSWCodePLParams.

#### Public Fields

	Name	Description
❖	Altitude ( <a href="#">see page 100</a> )	Altitude above sea level Available since version 1.7.
❖	IsolatedRoofs ( <a href="#">see page 100</a> )	Table containing numbers of recognized bars Available since version 1.7.
❖	NodalLoadsForAllBars ( <a href="#">see page 101</a> )	Flag indicating the method of modifying loads into nodal loads; If it is set as a value different from zero (True), then the loads for all the bars in the envelope will be modified. Available since version 1.7.
❖	NodalLoadsForBarsList ( <a href="#">see page 101</a> )	Selection list describing bars for which loads will be changed to nodal loads Available since version 1.7.
❖	PermFront ( <a href="#">see page 101</a> )	Available since version 1.7.
❖	PermLeftSide ( <a href="#">see page 101</a> )	Available since version 1.7.
❖	PermRear ( <a href="#">see page 102</a> )	Available since version 1.7.
❖	PermRightSide ( <a href="#">see page 102</a> )	Available since version 1.7.

◆	ReferenceLevel ( <a href="#">see page 102</a> )	Available since version 1.7.
◆	SnowPressure ( <a href="#">see page 102</a> )	Available since version 1.7.
◆	SnowPressureAutomatic ( <a href="#">see page 103</a> )	Available since version 1.7.
◆	SnowRedistribution ( <a href="#">see page 103</a> )	Available since version 1.7.
◆	SnowZone ( <a href="#">see page 103</a> )	Available since version 1.7.
◆	StructureHeight ( <a href="#">see page 103</a> )	Structure height Available since version 1.7.
◆	WindDynamicAction ( <a href="#">see page 104</a> )	Available since version 1.7.
◆	WindDynamicDecrement ( <a href="#">see page 104</a> )	Available since version 1.7.
◆	WindDynamicPeriod ( <a href="#">see page 104</a> )	Available since version 1.7.
◆	WindMultipleRoofs ( <a href="#">see page 104</a> )	Available since version 1.7.
◆	WindPressure ( <a href="#">see page 105</a> )	Available since version 1.7.
◆	WindPressureAutomatic ( <a href="#">see page 105</a> )	Available since version 1.7.
◆	WindPressureDistribOnHeight ( <a href="#">see page 105</a> )	Distribution of wind pressure with respect to height Available since version 1.7.
◆	WindSite ( <a href="#">see page 106</a> )	Available since version 1.7.
◆	WindZone ( <a href="#">see page 106</a> )	Available since version 1.7.

## Public Methods

	Name	Description
◆	IsolatedRoofGetLocation ( <a href="#">see page 106</a> )	Available since version 1.7.
◆	IsolatedRoofSetLocation ( <a href="#">see page 107</a> )	Available since version 1.7.
◆	SnowBarCoeffGet ( <a href="#">see page 107</a> )	Available since version 1.7.
◆	SnowBarCoeffSet ( <a href="#">see page 107</a> )	Available since version 1.7.
◆	WindBarCoeffGet ( <a href="#">see page 107</a> )	Function returns the coefficient value for a bar with the specified number. Coefficients are determined for the bars in the envelope. Available since version 1.7.
◆	WindBarCoeffSet ( <a href="#">see page 108</a> )	Function sets the coefficient value for a bar with the specified number. Coefficients can be determined only for the bars in the envelope. Available since version 1.7.

### II.1.2.4.2 IRobotSWCodePLParams Fields

The fields of the IRobotSWCodePLParams class are listed here.

## Public Fields

	Name	Description
◆	Altitude ( <a href="#">see page 100</a> )	Altitude above sea level Available since version 1.7.
◆	IsolatedRoofs ( <a href="#">see page 100</a> )	Table containing numbers of recognized bars Available since version 1.7.
◆	NodalLoadsForAllBars ( <a href="#">see page 101</a> )	Flag indicating the method of modifying loads into nodal loads; If it is set as a value different from zero (True), then the loads for all the bars in the envelope will be modified. Available since version 1.7.
◆	NodalLoadsForBarsList ( <a href="#">see page 101</a> )	Selection list describing bars for which loads will be changed to nodal loads Available since version 1.7.
◆	PermFront ( <a href="#">see page 101</a> )	Available since version 1.7.
◆	PermLeftSide ( <a href="#">see page 101</a> )	Available since version 1.7.
◆	PermRear ( <a href="#">see page 102</a> )	Available since version 1.7.

❖	PermRightSide (see page 102)	Available since version 1.7.
❖	ReferenceLevel (see page 102)	Available since version 1.7.
❖	SnowPressure (see page 102)	Available since version 1.7.
❖	SnowPressureAutomatic (see page 103)	Available since version 1.7.
❖	SnowRedistribution (see page 103)	Available since version 1.7.
❖	SnowZone (see page 103)	Available since version 1.7.
❖	StructureHeight (see page 103)	Structure height Available since version 1.7.
❖	WindDynamicAction (see page 104)	Available since version 1.7.
❖	WindDynamicDecrement (see page 104)	Available since version 1.7.
❖	WindDynamicPeriod (see page 104)	Available since version 1.7.
❖	WindMultipleRoofs (see page 104)	Available since version 1.7.
❖	WindPressure (see page 105)	Available since version 1.7.
❖	WindPressureAutomatic (see page 105)	Available since version 1.7.
❖	WindPressureDistribOnHeight (see page 105)	Distribution of wind pressure with respect to height Available since version 1.7.
❖	WindSite (see page 106)	Available since version 1.7.
❖	WindZone (see page 106)	Available since version 1.7.

#### II.1.2.4.2.1 Altitude

##### C++

```
HRESULT get_Altitude(double* );
HRESULT put_Altitude(double);
```

##### C#

```
public double Altitude { get; set; }
```

##### Visual Basic

```
Public Altitude As Double
```

##### Description

Altitude above sea level Available since version 1.7.

#### II.1.2.4.2.2 IsolatedRoofs

##### C++

```
HRESULT get_IsolatedRoofs(IRobotNumbersCollection** );
```

##### C#

```
public IRobotNumbersCollection IsolatedRoofs { get; }
```

##### Visual Basic

```
Public ReadOnly IsolatedRoofs As IRobotNumbersCollection
```

##### Description

Table containing numbers of recognized bars Available since version 1.7.

### II.1.2.4.2.3 NodalLoadsForAllBars

**C++**

```
HRESULT get_NodalLoadsForAllBars(VARIANT_BOOL* );
HRESULT put_NodalLoadsForAllBars(VARIANT_BOOL);
```

**C#**

```
public bool NodalLoadsForAllBars { get; set; }
```

**Visual Basic**

```
Public NodalLoadsForAllBars As Boolean
```

**Description**

Flag indicating the method of modifying loads into nodal loads; If it is set as a value different from zero (True), then the loads for all the bars in the envelope will be modified. Available since version 1.7.

### II.1.2.4.2.4 NodalLoadsForBarsList

**C++**

```
HRESULT get_NodalLoadsForBarsList(BSTR* );
HRESULT put_NodalLoadsForBarsList(BSTR);
```

**C#**

```
public String NodalLoadsForBarsList { get; set; }
```

**Visual Basic**

```
Public NodalLoadsForBarsList As String
```

**Description**

Selection list describing bars for which loads will be changed to nodal loads Available since version 1.7.

### II.1.2.4.2.5 PermFront

**C++**

```
HRESULT get_PermFront(double* );
HRESULT put_PermFront(double);
```

**C#**

```
public double PermFront { get; set; }
```

**Visual Basic**

```
Public PermFront As double
```

**Description**

Available since version 1.7.

### II.1.2.4.2.6 PermLeftSide

**C++**

```
HRESULT get_PermLeftSide(double* );
HRESULT put_PermLeftSide(double);
```

**C#**

```
public double PermLeftSide { get; set; }
```

**Visual Basic**

```
Public PermLeftSide As double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.7 PermRear****C++**

```
HRESULT get_PermRear(double*);  
HRESULT put_PermRear(double);
```

**C#**

```
public double PermRear { get; set; }
```

**Visual Basic**

```
Public PermRear As double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.8 PermRightSide****C++**

```
HRESULT get_PermRightSide(double*);  
HRESULT put_PermRightSide(double);
```

**C#**

```
public double PermRightSide { get; set; }
```

**Visual Basic**

```
Public PermRightSide As double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.9 ReferenceLevel****C++**

```
HRESULT get_ReferenceLevel(double*);  
HRESULT put_ReferenceLevel(double);
```

**C#**

```
public double ReferenceLevel { get; set; }
```

**Visual Basic**

```
Public ReferenceLevel As double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.10 SnowPressure****C++**

```
HRESULT get_SnowPressure(double*);  
HRESULT put_SnowPressure(double);
```

**C#**

```
public double SnowPressure { get; set; }
```

**Visual Basic**

```
Public SnowPressure As double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.11 SnowPressureAutomatic****C++**

```
HRESULT get_SnowPressureAutomatic(VARIANT_BOOL* );
HRESULT put_SnowPressureAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool SnowPressureAutomatic { get; set; }
```

**Visual Basic**

```
Public SnowPressureAutomatic As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.4.2.12 SnowRedistribution****C++**

```
HRESULT get_SnowRedistribution(VARIANT_BOOL* );
HRESULT put_SnowRedistribution(VARIANT_BOOL);
```

**C#**

```
public bool SnowRedistribution { get; set; }
```

**Visual Basic**

```
Public SnowRedistribution As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.4.2.13 SnowZone****C++**

```
HRESULT get_SnowZone(IRobotSWCodePLSnowZone* );
HRESULT put_SnowZone(IRobotSWCodePLSnowZone);
```

**C#**

```
public IRobotSWCodePLSnowZone SnowZone { get; set; }
```

**Visual Basic**

```
Public SnowZone As IRobotSWCodePLSnowZone
```

**Description**

Available since version 1.7.

**II.1.2.4.2.14 StructureHeight****C++**

```
HRESULT get_StructureHeight(double* );
HRESULT put_StructureHeight(double);
```

**C#**

```
public double StructureHeight { get; set; }
```

**Visual Basic**

```
Public StructureHeight As Double
```

**Description**

Structure height Available since version 1.7.

**II.1.2.4.2.15 WindDynamicAction****C++**

```
HRESULT get_WindDynamicAction(VARIANT_BOOL* );
HRESULT put_WindDynamicAction(VARIANT_BOOL);
```

**C#**

```
public bool WindDynamicAction { get; set; }
```

**Visual Basic**

```
Public WindDynamicAction As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.4.2.16 WindDynamicDecrement****C++**

```
HRESULT get_WindDynamicDecrement(double* );
HRESULT put_WindDynamicDecrement(double);
```

**C#**

```
public double WindDynamicDecrement { get; set; }
```

**Visual Basic**

```
Public WindDynamicDecrement As Double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.17 WindDynamicPeriod****C++**

```
HRESULT get_WindDynamicPeriod(double* );
HRESULT put_WindDynamicPeriod(double);
```

**C#**

```
public double WindDynamicPeriod { get; set; }
```

**Visual Basic**

```
Public WindDynamicPeriod As Double
```

**Description**

Available since version 1.7.

**II.1.2.4.2.18 WindMultipleRoofs****C++**

```
HRESULT get_WindMultipleRoofs(VARIANT_BOOL* );
```

```
HRESULT put_WindMultipleRoofs(VARIANT_BOOL);
```

**C#**

```
public bool WindMultipleRoofs { get; set; }
```

**Visual Basic**

```
Public WindMultipleRoofs As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.4.2.19 WindPressure

**C++**

```
HRESULT get_WindPressure(double*);  
HRESULT put_WindPressure(double);
```

**C#**

```
public double WindPressure { get; set; }
```

**Visual Basic**

```
Public WindPressure As Double
```

**Description**

Available since version 1.7.

### II.1.2.4.2.20 WindPressureAutomatic

**C++**

```
HRESULT get_WindPressureAutomatic(VARIANT_BOOL*);  
HRESULT put_WindPressureAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool WindPressureAutomatic { get; set; }
```

**Visual Basic**

```
Public WindPressureAutomatic As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.4.2.21 WindPressureDistribOnHeight

**C++**

```
HRESULT get_WindPressureDistribOnHeight(IRobotSWCodePLWindPressDistribType*);  
HRESULT put_WindPressureDistribOnHeight(IRobotSWCodePLWindPressDistribType);
```

**C#**

```
public IRobotSWCodePLWindPressDistribType WindPressureDistribOnHeight { get; set; }
```

**Visual Basic**

```
Public WindPressureDistribOnHeight As IRobotSWCodePLWindPressDistribType
```

**Description**

Distribution of wind pressure with respect to height Available since version 1.7.

### II.1.2.4.2.22 WindSite

#### C++

```
HRESULT get_WindSite(IRobotSWCodePLWindSite* );
HRESULT put_WindSite(IRobotSWCodePLWindSite);
```

#### C#

```
public IRobotSWCodePLWindSite WindSite { get; set; }
```

#### Visual Basic

```
Public WindSite As IRobotSWCodePLWindSite
```

#### Description

Available since version 1.7.

### II.1.2.4.2.23 WindZone

#### C++

```
HRESULT get_WindZone(IRobotSWCodePLWindZone* );
HRESULT put_WindZone(IRobotSWCodePLWindZone);
```

#### C#

```
public IRobotSWCodePLWindZone WindZone { get; set; }
```

#### Visual Basic

```
Public WindZone As IRobotSWCodePLWindZone
```

#### Description

Available since version 1.7.

### II.1.2.4.3 IRobotSWCodePLParams Methods

The methods of the IRobotSWCodePLParams class are listed here.

#### Public Methods

	Name	Description
💡	IsolatedRoofGetLocation (see page 106)	Available since version 1.7.
💡	IsolatedRoofSetLocation (see page 107)	Available since version 1.7.
💡	SnowBarCoeffGet (see page 107)	Available since version 1.7.
💡	SnowBarCoeffSet (see page 107)	Available since version 1.7.
💡	WindBarCoeffGet (see page 107)	Function returns the coefficient value for a bar with the specified number. Coefficients are determined for the bars in the envelope. Available since version 1.7.
💡	WindBarCoeffSet (see page 108)	Function sets the coefficient value for a bar with the specified number. Coefficients can be determined only for the bars in the envelope. Available since version 1.7.

### II.1.2.4.3.1 IsolatedRoofGetLocation

#### C++

```
HRESULT IsolatedRoofGetLocation(long _bar_num, VARIANT_BOOL _relative, double* ret);
```

#### C#

```
public double IsolatedRoofGetLocation(long _bar_num, bool _relative);
```

**Visual Basic**

```
Public Function IsolatedRoofGetLocation(_bar_num As long, _relative As Boolean) As double
```

**Description**

Available since version 1.7.

**II.1.2.4.3.2 IsolatedRoofSetLocation****C++**

```
HRESULT IsolatedRoofSetLocation(long _bar_num, double _loc, VARIANT_BOOL _relative);
```

**C#**

```
public void IsolatedRoofSetLocation(long _bar_num, double _loc, bool _relative);
```

**Visual Basic**

```
Public Sub IsolatedRoofSetLocation(_bar_num As long, _loc As double, _relative As Boolean)
```

**Description**

Available since version 1.7.

**II.1.2.4.3.3 SnowBarCoeffGet****C++**

```
HRESULT SnowBarCoeffGet(long _bar_num, double* ret);
```

**C#**

```
public double SnowBarCoeffGet(long _bar_num);
```

**Visual Basic**

```
Public Function SnowBarCoeffGet(_bar_num As long) As double
```

**Description**

Available since version 1.7.

**II.1.2.4.3.4 SnowBarCoeffSet****C++**

```
HRESULT SnowBarCoeffSet(long _bar_num, double _bar_coef);
```

**C#**

```
public void SnowBarCoeffSet(long _bar_num, double _bar_coef);
```

**Visual Basic**

```
Public Sub SnowBarCoeffSet(_bar_num As long, _bar_coef As double)
```

**Description**

Available since version 1.7.

**II.1.2.4.3.5 WindBarCoeffGet****C++**

```
HRESULT WindBarCoeffGet(long _bar_num, double* ret);
```

**C#**

```
public double WindBarCoeffGet(long _bar_num);
```

**Visual Basic**

```
Public Function WindBarCoeffGet(_bar_num As long) As double
```

**Description**

Function returns the coefficient value for a bar with the specified number. Coefficients are determined for the bars in the envelope. Available since version 1.7.

**II.1.2.4.3.6 WindBarCoeffSet****C++**

```
HRESULT WindBarCoeffSet(long _bar_num, double _bar_coeff);
```

**C#**

```
public void WindBarCoeffSet(long _bar_num, double _bar_coeff);
```

**Visual Basic**

```
Public Sub WindBarCoeffSet(_bar_num As long, _bar_coeff As double)
```

**Description**

Function sets the coefficient value for a bar with the specified number. Coefficients can be determined only for the bars in the envelope. Available since version 1.7.

**II.1.2.5 IRobotSWCodePLWindZone****C++**

```
enum IRobotSWCodePLWindZone;
```

**C#**

```
public enum IRobotSWCodePLWindZone;
```

**Visual Basic**

```
Public Enum IRobotSWCodePLWindZone
```

**Members**

Members	Description
I_SWCPLWZ_I = 0	Available since version 1.7.
I_SWCPLWZ_II = 1	Available since version 1.7.
I_SWCPLWZ_IA = 2	Available since version 1.7.
I_SWCPLWZ_IB = 3	Available since version 1.7.
I_SWCPLWZ_III = 4	Available since version 1.7.

**II.1.2.6 IRobotSWCodePLWindSite****C++**

```
enum IRobotSWCodePLWindSite;
```

**C#**

```
public enum IRobotSWCodePLWindSite;
```

**Visual Basic**

```
Public Enum IRobotSWCodePLWindSite
```

**Members**

Members	Description
I_SWCPLWS_A = 0	Available since version 1.7.
I_SWCPLWS_B = 1	Available since version 1.7.
I_SWCPLWS_C = 2	Available since version 1.7.

**II.1.2.7 IRobotSWCodePLSnowZone****C++**

```
enum IRobotSWCodePLSnowZone;
```

**C#**

```
public enum IRobotSWCodePLSnowZone;
```

**Visual Basic**

```
Public Enum IRobotSWCodePLSnowZone
```

**Members**

Members	Description
I_SWCPLSZ_I = 0	Available since version 1.7.
I_SWCPLSZ_II = 1	Available since version 1.7.
I_SWCPLSZ_III = 2	Available since version 1.7.
I_SWCPLSZ_IV = 3	Available since version 1.7.

**II.1.2.8 IRobotSWCodeFRParams****Class Hierarchy****C++**

```
interface IRobotSWCodeFRParams : IDispatch;
```

**C#**

```
public interface IRobotSWCodeFRParams;
```

**Visual Basic**

```
Public Interface IRobotSWCodeFRParams
```

**Description**

Parameters of snow/wind load generation for the French code.

**II.1.2.8.1 IRobotSWCodeFRParams Members**

The following tables list the members exposed by IRobotSWCodeFRParams.

**Public Fields**

	Name	Description
◆	Altitude (see page 113)	Available since version 1.7.
◆	IsolatedRoofs (see page 113)	Available since version 1.7.
◆	NodalLoadsForAllBars (see page 113)	Available since version 1.7.

◆ NodalLoadsForBarsList (see page 114)	Available since version 1.7.
◆ OpenStructure (see page 114)	Available since version 1.7.
◆ PermDoorFront (see page 114)	Available since version 1.7.
◆ PermDoorFrontPresent (see page 114)	Available since version 1.7.
◆ PermDoorLeftSide (see page 115)	Available since version 1.7.
◆ PermDoorLeftSidePresent (see page 115)	Available since version 1.7.
◆ PermDoorRear (see page 115)	Available since version 1.7.
◆ PermDoorRearPresent (see page 115)	Available since version 1.7.
◆ PermDoorRightSide (see page 116)	Available since version 1.7.
◆ PermDoorRightSidePresent (see page 116)	Available since version 1.7.
◆ PermFront (see page 116)	Available since version 1.7.
◆ PermLeftSide (see page 116)	Available since version 1.7.
◆ PermRear (see page 117)	Available since version 1.7.
◆ PermRightSide (see page 117)	Available since version 1.7.
◆ ReferenceLevel (see page 117)	Available since version 1.7.
◆ RiseOfRoof (see page 117)	Available since version 1.7.
◆ RiseOfRoofAutomatic (see page 118)	Available since version 1.7.
◆ SnowGutterBars (see page 118)	Available since version 1.7.
◆ SnowIlsWaterOutflow (see page 118)	Available since version 1.7.
◆ SnowObstacles (see page 119)	Obstacles - list of bars Available since version 1.7.
◆ SnowPressureExtreme (see page 119)	Available since version 1.7.
◆ SnowPressureExtremeAutomatic (see page 119)	Available since version 1.7.
◆ SnowPressureNormal (see page 119)	Available since version 1.7.
◆ SnowPressureNormalAutomatic (see page 120)	Available since version 1.7.
◆ SnowRedistribution (see page 120)	Available since version 1.7.
◆ SnowRegion (see page 120)	Symbol of snow region Available since version 1.7.
◆ SnowType (see page 120)	Available since version 1.7.
◆ SnowWaterOutflow (see page 121)	Water outflow Available since version 1.7.
◆ StructureHeight (see page 121)	Available since version 1.7.
◆ SurfaceLower (see page 121)	Available since version 1.7.
◆ SurfaceUpper (see page 121)	Available since version 1.7.
◆ WindCoastalArea (see page 122)	Available since version 1.7.
◆ WindDeltaCoeff (see page 122)	Available since version 1.7.
◆ WindDeltaCoeffAutomatic (see page 122)	Available since version 1.7.
◆ WindDynamicAction (see page 122)	Available since version 1.7.

◆	WindDynamicActionCoeff ( <a href="#">see page 123</a> )	Available since version 1.7.
◆	WindDynamicActionCoeffAutomatic ( <a href="#">see page 123</a> )	Available since version 1.7.
◆	WindDynamicActionPeriod ( <a href="#">see page 123</a> )	Available since version 1.7.
◆	WindDynamicActionSteelStructure ( <a href="#">see page 124</a> )	Available since version 1.7.
◆	WindFacadeOffset ( <a href="#">see page 124</a> )	Available since version 1.7.
◆	WindMultipleRoof ( <a href="#">see page 124</a> )	Available since version 1.7.
◆	WindPressure ( <a href="#">see page 124</a> )	Available since version 1.7.
◆	WindPressureAutomatic ( <a href="#">see page 125</a> )	Available since version 1.7.
◆	WindPressureCeCiMinimum ( <a href="#">see page 125</a> )	Available since version 1.7.
◆	WindPressureVariable ( <a href="#">see page 125</a> )	Available since version 1.7.
◆	WindRegion ( <a href="#">see page 125</a> )	Name of wind region Available since version 1.7.
◆	WindSite ( <a href="#">see page 126</a> )	Available since version 1.7.
◆	WindType ( <a href="#">see page 126</a> )	Available since version 1.7.

## Public Methods

	Name	Description
◆	IsolatedRoofGetLocation ( <a href="#">see page 126</a> )	Available since version 1.7.
◆	IsolatedRoofSetLocation ( <a href="#">see page 127</a> )	Available since version 1.7.
◆	SnowBarCoeffGet ( <a href="#">see page 127</a> )	Available since version 1.7.
◆	SnowBarCoeffSet ( <a href="#">see page 127</a> )	Available since version 1.7.
◆	WindBarCoeffGet ( <a href="#">see page 128</a> )	Available since version 1.7.
◆	WindBarCoeffSet ( <a href="#">see page 128</a> )	Available since version 1.7.

### II.1.2.8.2 IRobotSWCodeFRParams Fields

The fields of the IRobotSWCodeFRParams class are listed here.

## Public Fields

	Name	Description
◆	Altitude ( <a href="#">see page 113</a> )	Available since version 1.7.
◆	IsolatedRoofs ( <a href="#">see page 113</a> )	Available since version 1.7.
◆	NodalLoadsForAllBars ( <a href="#">see page 113</a> )	Available since version 1.7.
◆	NodalLoadsForBarsList ( <a href="#">see page 114</a> )	Available since version 1.7.
◆	OpenStructure ( <a href="#">see page 114</a> )	Available since version 1.7.
◆	PermDoorFront ( <a href="#">see page 114</a> )	Available since version 1.7.
◆	PermDoorFrontPresent ( <a href="#">see page 114</a> )	Available since version 1.7.
◆	PermDoorLeftSide ( <a href="#">see page 115</a> )	Available since version 1.7.
◆	PermDoorLeftSidePresent ( <a href="#">see page 115</a> )	Available since version 1.7.
◆	PermDoorRear ( <a href="#">see page 115</a> )	Available since version 1.7.

❖	PermDoorRearPresent (☞ see page 115)	Available since version 1.7.
❖	PermDoorRightSide (☞ see page 116)	Available since version 1.7.
❖	PermDoorRightSidePresent (☞ see page 116)	Available since version 1.7.
❖	PermFront (☞ see page 116)	Available since version 1.7.
❖	PermLeftSide (☞ see page 116)	Available since version 1.7.
❖	PermRear (☞ see page 117)	Available since version 1.7.
❖	PermRightSide (☞ see page 117)	Available since version 1.7.
❖	ReferenceLevel (☞ see page 117)	Available since version 1.7.
❖	RiseOfRoof (☞ see page 117)	Available since version 1.7.
❖	RiseOfRoofAutomatic (☞ see page 118)	Available since version 1.7.
❖	SnowGutterBars (☞ see page 118)	Available since version 1.7.
❖	SnowIsWaterOutflow (☞ see page 118)	Available since version 1.7.
❖	SnowObstacles (☞ see page 119)	Obstacles - list of bars Available since version 1.7.
❖	SnowPressureExtreme (☞ see page 119)	Available since version 1.7.
❖	SnowPressureExtremeAutomatic (☞ see page 119)	Available since version 1.7.
❖	SnowPressureNormal (☞ see page 119)	Available since version 1.7.
❖	SnowPressureNormalAutomatic (☞ see page 120)	Available since version 1.7.
❖	SnowRedistribution (☞ see page 120)	Available since version 1.7.
❖	SnowRegion (☞ see page 120)	Symbol of snow region Available since version 1.7.
❖	SnowType (☞ see page 120)	Available since version 1.7.
❖	SnowWaterOutflow (☞ see page 121)	Water outflow Available since version 1.7.
❖	StructureHeight (☞ see page 121)	Available since version 1.7.
❖	SurfaceLower (☞ see page 121)	Available since version 1.7.
❖	SurfaceUpper (☞ see page 121)	Available since version 1.7.
❖	WindCoastalArea (☞ see page 122)	Available since version 1.7.
❖	WindDeltaCoeff (☞ see page 122)	Available since version 1.7.
❖	WindDeltaCoeffAutomatic (☞ see page 122)	Available since version 1.7.
❖	WindDynamicAction (☞ see page 122)	Available since version 1.7.
❖	WindDynamicActionCoeff (☞ see page 123)	Available since version 1.7.
❖	WindDynamicActionCoeffAutomatic (☞ see page 123)	Available since version 1.7.
❖	WindDynamicActionPeriod (☞ see page 123)	Available since version 1.7.
❖	WindDynamicActionSteelStructure (☞ see page 124)	Available since version 1.7.
❖	WindFacadeOffset (☞ see page 124)	Available since version 1.7.
❖	WindMultipleRoof (☞ see page 124)	Available since version 1.7.
❖	WindPressure (☞ see page 124)	Available since version 1.7.

◆	WindPressureAutomatic (see page 125)	Available since version 1.7.
◆	WindPressureCeCiMinimum (see page 125)	Available since version 1.7.
◆	WindPressureVariable (see page 125)	Available since version 1.7.
◆	WindRegion (see page 125)	Name of wind region Available since version 1.7.
◆	WindSite (see page 126)	Available since version 1.7.
◆	WindType (see page 126)	Available since version 1.7.

### II.1.2.8.2.1 Altitude

#### C++

```
HRESULT get_Altitude(double* );
HRESULT put_Altitude(double);
```

#### C#

```
public double Altitude { get; set; }
```

#### Visual Basic

```
Public Altitude As Double
```

#### Description

Available since version 1.7.

### II.1.2.8.2.2 IsolatedRoofs

#### C++

```
HRESULT get_IsolatedRoofs(IRobotNumbersCollection** );
```

#### C#

```
public IRobotNumbersCollection IsolatedRoofs { get; }
```

#### Visual Basic

```
Public ReadOnly IsolatedRoofs As IRobotNumbersCollection
```

#### Description

Available since version 1.7.

### II.1.2.8.2.3 NodalLoadsForAllBars

#### C++

```
HRESULT get_NodalLoadsForAllBars(VARIANT_BOOL* );
HRESULT put_NodalLoadsForAllBars(VARIANT_BOOL);
```

#### C#

```
public bool NodalLoadsForAllBars { get; set; }
```

#### Visual Basic

```
Public NodalLoadsForAllBars As Boolean
```

#### Description

Available since version 1.7.

#### II.1.2.8.2.4 NodalLoadsForBarsList

**C++**

```
HRESULT get_NodalLoadsForBarsList(BSTR* );
HRESULT put_NodalLoadsForBarsList(BSTR);
```

**C#**

```
public String NodalLoadsForBarsList { get; set; }
```

**Visual Basic**

```
Public NodalLoadsForBarsList As String
```

**Description**

Available since version 1.7.

#### II.1.2.8.2.5 OpenStructure

**C++**

```
HRESULT get_OpenStructure(VARIANT_BOOL* );
HRESULT put_OpenStructure(VARIANT_BOOL);
```

**C#**

```
public bool OpenStructure { get; set; }
```

**Visual Basic**

```
Public OpenStructure As Boolean
```

**Description**

Available since version 1.7.

#### II.1.2.8.2.6 PermDoorFront

**C++**

```
HRESULT get_PermDoorFront(double* );
HRESULT put_PermDoorFront(double);
```

**C#**

```
public double PermDoorFront { get; set; }
```

**Visual Basic**

```
Public PermDoorFront As double
```

**Description**

Available since version 1.7.

#### II.1.2.8.2.7 PermDoorFrontPresent

**C++**

```
HRESULT get_PermDoorFrontPresent(VARIANT_BOOL* );
HRESULT put_PermDoorFrontPresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorFrontPresent { get; set; }
```

**Visual Basic**

```
Public PermDoorFrontPresent As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.8 PermDoorLeftSide****C++**

```
HRESULT get_PermDoorLeftSide(double*);  
HRESULT put_PermDoorLeftSide(double);
```

**C#**

```
public double PermDoorLeftSide { get; set; }
```

**Visual Basic**

```
Public PermDoorLeftSide As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.9 PermDoorLeftSidePresent****C++**

```
HRESULT get_PermDoorLeftSidePresent(VARIANT_BOOL*);  
HRESULT put_PermDoorLeftSidePresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorLeftSidePresent { get; set; }
```

**Visual Basic**

```
Public PermDoorLeftSidePresent As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.10 PermDoorRear****C++**

```
HRESULT get_PermDoorRear(double*);  
HRESULT put_PermDoorRear(double);
```

**C#**

```
public double PermDoorRear { get; set; }
```

**Visual Basic**

```
Public PermDoorRear As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.11 PermDoorRearPresent****C++**

```
HRESULT get_PermDoorRearPresent(VARIANT_BOOL*);  
HRESULT put_PermDoorRearPresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorRearPresent { get; set; }
```

**Visual Basic**

```
Public PermDoorRearPresent As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.12 PermDoorRightSide****C++**

```
HRESULT get_PermDoorRightSide(double*);  
HRESULT put_PermDoorRightSide(double);
```

**C#**

```
public double PermDoorRightSide { get; set; }
```

**Visual Basic**

```
Public PermDoorRightSide As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.13 PermDoorRightSidePresent****C++**

```
HRESULT get_PermDoorRightSidePresent(VARIANT_BOOL*);  
HRESULT put_PermDoorRightSidePresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorRightSidePresent { get; set; }
```

**Visual Basic**

```
Public PermDoorRightSidePresent As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.14 PermFront****C++**

```
HRESULT get_PermFront(double*);  
HRESULT put_PermFront(double);
```

**C#**

```
public double PermFront { get; set; }
```

**Visual Basic**

```
Public PermFront As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.15 PermLeftSide****C++**

```
HRESULT get_PermLeftSide(double*);  
HRESULT put_PermLeftSide(double);
```

**C#**

```
public double PermLeftSide { get; set; }
```

**Visual Basic**

```
Public PermLeftSide As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.16 PermRear****C++**

```
HRESULT get_PermRear(double*);  
HRESULT put_PermRear(double);
```

**C#**

```
public double PermRear { get; set; }
```

**Visual Basic**

```
Public PermRear As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.17 PermRightSide****C++**

```
HRESULT get_PermRightSide(double*);  
HRESULT put_PermRightSide(double);
```

**C#**

```
public double PermRightSide { get; set; }
```

**Visual Basic**

```
Public PermRightSide As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.18 ReferenceLevel****C++**

```
HRESULT get_ReferenceLevel(double*);  
HRESULT put_ReferenceLevel(double);
```

**C#**

```
public double ReferenceLevel { get; set; }
```

**Visual Basic**

```
Public ReferenceLevel As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.19 RiseOfRoof****C++**

```
HRESULT get_RiseOfRoof(double*);
```

```
HRESULT put_RiseOfRoof(double);
```

**C#**

```
public double RiseOfRoof { get; set; }
```

**Visual Basic**

```
Public RiseOfRoof As Double
```

**Description**

Available since version 1.7.

### II.1.2.8.2.20 RiseOfRoofAutomatic

**C++**

```
HRESULT get_RiseOfRoofAutomatic(VARIANT_BOOL*);  
HRESULT put_RiseOfRoofAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool RiseOfRoofAutomatic { get; set; }
```

**Visual Basic**

```
Public RiseOfRoofAutomatic As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.8.2.21 SnowGutterBars

**C++**

```
HRESULT get_SnowGutterBars(BSTR*);  
HRESULT put_SnowGutterBars(BSTR);
```

**C#**

```
public String SnowGutterBars { get; set; }
```

**Visual Basic**

```
Public SnowGutterBars As String
```

**Description**

Available since version 1.7.

### II.1.2.8.2.22 SnowIsWaterOutflow

**C++**

```
HRESULT get_SnowIsWaterOutflow(VARIANT_BOOL*);  
HRESULT put_SnowIsWaterOutflow(VARIANT_BOOL);
```

**C#**

```
public bool SnowIsWaterOutflow { get; set; }
```

**Visual Basic**

```
Public SnowIsWaterOutflow As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.8.2.23 SnowObstacles

#### C++

```
HRESULT get_SnowObstacles(BSTR* );
HRESULT put_SnowObstacles(BSTR);
```

#### C#

```
public String SnowObstacles { get; set; }
```

#### Visual Basic

```
Public SnowObstacles As String
```

#### Description

Obstacles - list of bars Available since version 1.7.

### II.1.2.8.2.24 SnowPressureExtreme

#### C++

```
HRESULT get_SnowPressureExtreme(double* );
HRESULT put_SnowPressureExtreme(double);
```

#### C#

```
public double SnowPressureExtreme { get; set; }
```

#### Visual Basic

```
Public SnowPressureExtreme As Double
```

#### Description

Available since version 1.7.

### II.1.2.8.2.25 SnowPressureExtremeAutomatic

#### C++

```
HRESULT get_SnowPressureExtremeAutomatic(VARIANT_BOOL* );
HRESULT put_SnowPressureExtremeAutomatic(VARIANT_BOOL);
```

#### C#

```
public bool SnowPressureExtremeAutomatic { get; set; }
```

#### Visual Basic

```
Public SnowPressureExtremeAutomatic As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.8.2.26 SnowPressureNormal

#### C++

```
HRESULT get_SnowPressureNormal(double* );
HRESULT put_SnowPressureNormal(double);
```

#### C#

```
public double SnowPressureNormal { get; set; }
```

#### Visual Basic

```
Public SnowPressureNormal As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.27 SnowPressureNormalAutomatic****C++**

```
HRESULT get_SnowPressureNormalAutomatic(VARIANT_BOOL* );
HRESULT put_SnowPressureNormalAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool SnowPressureNormalAutomatic { get; set; }
```

**Visual Basic**

```
Public SnowPressureNormalAutomatic As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.28 SnowRedistribution****C++**

```
HRESULT get_SnowRedistribution(VARIANT_BOOL* );
HRESULT put_SnowRedistribution(VARIANT_BOOL);
```

**C#**

```
public bool SnowRedistribution { get; set; }
```

**Visual Basic**

```
Public SnowRedistribution As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.29 SnowRegion****C++**

```
HRESULT get_SnowRegion(BSTR* );
HRESULT put_SnowRegion(BSTR);
```

**C#**

```
public String SnowRegion { get; set; }
```

**Visual Basic**

```
Public SnowRegion As String
```

**Description**

Symbol of snow region Available since version 1.7.

**II.1.2.8.2.30 SnowType****C++**

```
HRESULT get_SnowType(IRobotSWCodeFRSnowType* );
HRESULT put_SnowType(IRobotSWCodeFRSnowType);
```

**C#**

```
public IRobotSWCodeFRSnowType SnowType { get; set; }
```

**Visual Basic**

```
Public SnowType As IRobotSWCodeFRSnowType
```

**Description**

Available since version 1.7.

**II.1.2.8.2.31 SnowWaterOutflow****C++**

```
HRESULT get_SnowWaterOutflow(double* );
HRESULT put_SnowWaterOutflow(double);
```

**C#**

```
public double SnowWaterOutflow { get; set; }
```

**Visual Basic**

```
Public SnowWaterOutflow As double
```

**Description**

Water outflow Available since version 1.7.

**II.1.2.8.2.32 StructureHeight****C++**

```
HRESULT get_StructureHeight(double* );
HRESULT put_StructureHeight(double);
```

**C#**

```
public double StructureHeight { get; set; }
```

**Visual Basic**

```
Public StructureHeight As double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.33 SurfaceLower****C++**

```
HRESULT get_SurfaceLower(IRobotSWCodeFRSurfaceType* );
HRESULT put_SurfaceLower(IRobotSWCodeFRSurfaceType);
```

**C#**

```
public IRobotSWCodeFRSurfaceType SurfaceLower { get; set; }
```

**Visual Basic**

```
Public SurfaceLower As IRobotSWCodeFRSurfaceType
```

**Description**

Available since version 1.7.

**II.1.2.8.2.34 SurfaceUpper****C++**

```
HRESULT get_SurfaceUpper(IRobotSWCodeFRSurfaceType* );
HRESULT put_SurfaceUpper(IRobotSWCodeFRSurfaceType);
```

**C#**

```
public IRobotSWCodeFRSurfaceType SurfaceUpper { get; set; }
```

**Visual Basic**

```
Public SurfaceUpper As IRobotSWCodeFRSurfaceType
```

**Description**

Available since version 1.7.

**II.1.2.8.2.35 WindCoastalArea****C++**

```
HRESULT get_WindCoastalArea(VARIANT_BOOL* );
HRESULT put_WindCoastalArea(VARIANT_BOOL);
```

**C#**

```
public bool WindCoastalArea { get; set; }
```

**Visual Basic**

```
Public WindCoastalArea As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.36 WindDeltaCoeff****C++**

```
HRESULT get_WindDeltaCoeff(double* );
HRESULT put_WindDeltaCoeff(double);
```

**C#**

```
public double WindDeltaCoeff { get; set; }
```

**Visual Basic**

```
Public WindDeltaCoeff As Double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.37 WindDeltaCoeffAutomatic****C++**

```
HRESULT get_WindDeltaCoeffAutomatic(VARIANT_BOOL* );
HRESULT put_WindDeltaCoeffAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool WindDeltaCoeffAutomatic { get; set; }
```

**Visual Basic**

```
Public WindDeltaCoeffAutomatic As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.38 WindDynamicAction****C++**

```
HRESULT get_WindDynamicAction(VARIANT_BOOL* );
```

```
HRESULT put_WindDynamicAction(VARIANT_BOOL);
```

**C#**

```
public bool WindDynamicAction { get; set; }
```

**Visual Basic**

```
Public WindDynamicAction As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.8.2.39 WindDynamicActionCoeff

**C++**

```
HRESULT get_WindDynamicActionCoeff(double*);  
HRESULT put_WindDynamicActionCoeff(double);
```

**C#**

```
public double WindDynamicActionCoeff { get; set; }
```

**Visual Basic**

```
Public WindDynamicActionCoeff As Double
```

**Description**

Available since version 1.7.

### II.1.2.8.2.40 WindDynamicActionCoeffAutomatic

**C++**

```
HRESULT get_WindDynamicActionCoeffAutomatic(VARIANT_BOOL*);  
HRESULT put_WindDynamicActionCoeffAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool WindDynamicActionCoeffAutomatic { get; set; }
```

**Visual Basic**

```
Public WindDynamicActionCoeffAutomatic As Boolean
```

**Description**

Available since version 1.7.

### II.1.2.8.2.41 WindDynamicActionPeriod

**C++**

```
HRESULT get_WindDynamicActionPeriod(double*);  
HRESULT put_WindDynamicActionPeriod(double);
```

**C#**

```
public double WindDynamicActionPeriod { get; set; }
```

**Visual Basic**

```
Public WindDynamicActionPeriod As Double
```

**Description**

Available since version 1.7.

### II.1.2.8.2.42 WindDynamicActionSteelStructure

#### C++

```
HRESULT get_WindDynamicActionSteelStructure(VARIANT_BOOL* );
HRESULT put_WindDynamicActionSteelStructure(VARIANT_BOOL);
```

#### C#

```
public bool WindDynamicActionSteelStructure { get; set; }
```

#### Visual Basic

```
Public WindDynamicActionSteelStructure As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.8.2.43 WindFacadeOffset

#### C++

```
HRESULT get_WindFacadeOffset(VARIANT_BOOL* );
HRESULT put_WindFacadeOffset(VARIANT_BOOL);
```

#### C#

```
public bool WindFacadeOffset { get; set; }
```

#### Visual Basic

```
Public WindFacadeOffset As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.8.2.44 WindMultipleRoof

#### C++

```
HRESULT get_WindMultipleRoof(VARIANT_BOOL* );
HRESULT put_WindMultipleRoof(VARIANT_BOOL);
```

#### C#

```
public bool WindMultipleRoof { get; set; }
```

#### Visual Basic

```
Public WindMultipleRoof As Boolean
```

#### Description

Available since version 1.7.

### II.1.2.8.2.45 WindPressure

#### C++

```
HRESULT get_WindPressure(double* );
HRESULT put_WindPressure(double);
```

#### C#

```
public double WindPressure { get; set; }
```

#### Visual Basic

```
Public WindPressure As double
```

**Description**

Available since version 1.7.

**II.1.2.8.2.46 WindPressureAutomatic****C++**

```
HRESULT get_WindPressureAutomatic(VARIANT_BOOL* );
HRESULT put_WindPressureAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool WindPressureAutomatic { get; set; }
```

**Visual Basic**

```
Public WindPressureAutomatic As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.47 WindPressureCeCiMinimum****C++**

```
HRESULT get_WindPressureCeCiMinimum(VARIANT_BOOL* );
HRESULT put_WindPressureCeCiMinimum(VARIANT_BOOL);
```

**C#**

```
public bool WindPressureCeCiMinimum { get; set; }
```

**Visual Basic**

```
Public WindPressureCeCiMinimum As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.48 WindPressureVariable****C++**

```
HRESULT get_WindPressureVariable(VARIANT_BOOL* );
HRESULT put_WindPressureVariable(VARIANT_BOOL);
```

**C#**

```
public bool WindPressureVariable { get; set; }
```

**Visual Basic**

```
Public WindPressureVariable As Boolean
```

**Description**

Available since version 1.7.

**II.1.2.8.2.49 WindRegion****C++**

```
HRESULT get_WindRegion(BSTR* );
HRESULT put_WindRegion(BSTR);
```

**C#**

```
public String WindRegion { get; set; }
```

**Visual Basic**

```
Public WindRegion As String
```

**Description**

Name of wind region Available since version 1.7.

**II.1.2.8.2.50 WindSite****C++**

```
HRESULT get_WindSite(IRobotSWCodeFRWindSite* );
HRESULT put_WindSite(IRobotSWCodeFRWindSite);
```

**C#**

```
public IRobotSWCodeFRWindSite WindSite { get; set; }
```

**Visual Basic**

```
Public WindSite As IRobotSWCodeFRWindSite
```

**Description**

Available since version 1.7.

**II.1.2.8.2.51 WindType****C++**

```
HRESULT get_WindType(IRobotSWCodeFRWindType* );
HRESULT put_WindType(IRobotSWCodeFRWindType);
```

**C#**

```
public IRobotSWCodeFRWindType WindType { get; set; }
```

**Visual Basic**

```
Public WindType As IRobotSWCodeFRWindType
```

**Description**

Available since version 1.7.

**II.1.2.8.3 IRobotSWCodeFRParams Methods**

The methods of the IRobotSWCodeFRParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	IsolatedRoofGetLocation (🔗 see page 126)	Available since version 1.7.
💡	IsolatedRoofSetLocation (🔗 see page 127)	Available since version 1.7.
💡	SnowBarCoeffGet (🔗 see page 127)	Available since version 1.7.
💡	SnowBarCoeffSet (🔗 see page 127)	Available since version 1.7.
💡	WindBarCoeffGet (🔗 see page 128)	Available since version 1.7.
💡	WindBarCoeffSet (🔗 see page 128)	Available since version 1.7.

**II.1.2.8.3.1 IsolatedRoofGetLocation****C++**

```
HRESULT IsolatedRoofGetLocation(long _bar_num, VARIANT_BOOL _relative, double* ret);
```

**C#**

```
public double IsolatedRoofGetLocation(long _bar_num, bool _relative);
```

**Visual Basic**

```
Public Function IsolatedRoofGetLocation(_bar_num As long, _relative As Boolean) As double
```

**Description**

Available since version 1.7.

**II.1.2.8.3.2 IsolatedRoofSetLocation****C++**

```
HRESULT IsolatedRoofSetLocation(long _bar_num, double _location, VARIANT_BOOL _relative);
```

**C#**

```
public void IsolatedRoofSetLocation(long _bar_num, double _location, bool _relative);
```

**Visual Basic**

```
Public Sub IsolatedRoofSetLocation(_bar_num As long, _location As double, _relative As Boolean)
```

**Description**

Available since version 1.7.

**II.1.2.8.3.3 SnowBarCoeffGet****C++**

```
HRESULT SnowBarCoeffGet(long _bar_num, double* ret);
```

**C#**

```
public double SnowBarCoeffGet(long _bar_num);
```

**Visual Basic**

```
Public Function SnowBarCoeffGet(_bar_num As long) As double
```

**Description**

Available since version 1.7.

**II.1.2.8.3.4 SnowBarCoeffSet****C++**

```
HRESULT SnowBarCoeffSet(long _bar_num, double _coeff);
```

**C#**

```
public void SnowBarCoeffSet(long _bar_num, double _coeff);
```

**Visual Basic**

```
Public Sub SnowBarCoeffSet(_bar_num As long, _coeff As double)
```

**Description**

Available since version 1.7.

### II.1.2.8.3.5 WindBarCoeffGet

**C++**

```
HRESULT WindBarCoeffGet(long _bar_num, double* ret);
```

**C#**

```
public double WindBarCoeffGet(long _bar_num);
```

**Visual Basic**

```
Public Function WindBarCoeffGet(_bar_num As long) As double
```

**Description**

Available since version 1.7.

### II.1.2.8.3.6 WindBarCoeffSet

**C++**

```
HRESULT WindBarCoeffSet(long _bar_num, double _coeff);
```

**C#**

```
public void WindBarCoeffSet(long _bar_num, double _coeff);
```

**Visual Basic**

```
Public Sub WindBarCoeffSet(_bar_num As long, _coeff As double)
```

**Description**

Available since version 1.7.

### II.1.2.9 IRobotSWCodeFRWindSite

**C++**

```
enum IRobotSWCodeFRWindSite;
```

**C#**

```
public enum IRobotSWCodeFRWindSite;
```

**Visual Basic**

```
Public Enum IRobotSWCodeFRWindSite
```

**Members**

Members	Description
I_SWCFRWS_OBSCURED = 0	Available since version 1.7.
I_SWCFRWS_NORMAL = 1	Available since version 1.7.
I_SWCFRWS_EXPOSED = 2	Available since version 1.7.

### II.1.2.10 IRobotSWCodeFRWindType

**C++**

```
enum IRobotSWCodeFRWindType;
```

**C#**

```
public enum IRobotSWCodeFRWindType;
```

**Visual Basic**

```
Public Enum IRobotSWCodeFRWindType
```

**Members**

Members	Description
I_SWCFRWT_NORMAL = 0	Available since version 1.7.
I_SWCFRWT_EXTREME = 1	Available since version 1.7.

**II.1.2.11 IRobotSWCodeFRSurfaceType****C++**

```
enum IRobotSWCodeFRSurfaceType;
```

**C#**

```
public enum IRobotSWCodeFRSurfaceType;
```

**Visual Basic**

```
Public Enum IRobotSWCodeFRSurfaceType
```

**Members**

Members	Description
I_SWCFRST_SMOOTH_OR_CORRUGATED = 0	Available since version 1.7.
I_SWCFRST_FOLDED_OR_CORRUGATED = 1	Available since version 1.7.
I_SWCFRST_RIBBED = 2	Available since version 1.7.

**II.1.2.12 IRobotSWCodeFRSnowType****C++**

```
enum IRobotSWCodeFRSnowType;
```

**C#**

```
public enum IRobotSWCodeFRSnowType;
```

**Visual Basic**

```
Public Enum IRobotSWCodeFRSnowType
```

**Members**

Members	Description
I_SWCFRST_NORMAL = 1	Available since version 1.7.
I_SWCFRST_ACCIDENTAL = 2	Available since version 1.7.
I_SWCFRST_NORMAL_AND_ACCIDENTAL = 0	Available since version 1.7.

**II.1.2.13 IRobotSWCodePLWindPressDistribType****C++**

```
enum IRobotSWCodePLWindPressDistribType;
```

**C#**

```
public enum IRobotSWCodePLWindPressDistribType;
```

**Visual Basic**

```
Public Enum IRobotSWCodePLWindPressDistribType
```

## Members

Members	Description
I_SWCPLWPDT_CONSTANT = 0	Available since version 1.7.
I_SWCPLWPDT_VARIABLE = 1	Available since version 1.7.

### II.1.2.14 IRobotSWCodeECPParams

#### Class Hierarchy

##### C++

```
interface IRobotSWCodeECPParams : IDispatch;
```

##### C#

```
public interface IRobotSWCodeECPParams;
```

##### Visual Basic

```
Public Interface IRobotSWCodeECPParams
```

#### Description

Parameters of snow/wind load generation for Eurocode. .

#### Version

Available since version 4.

### II.1.2.14.1 IRobotSWCodeECPParams Members

The following tables list the members exposed by IRobotSWCodeECPParams.

#### Public Fields

	Name	Description
◆	Altitude ( <a href="#">see page 133</a> )	
◆	GlobalCDIR ( <a href="#">see page 133</a> )	
◆	GroundType ( <a href="#">see page 133</a> )	
◆	LeftWind2NordAngle ( <a href="#">see page 133</a> )	
◆	NodalLoadsForAllBars ( <a href="#">see page 134</a> )	
◆	NodalLoadsForBarsList ( <a href="#">see page 134</a> )	
◆	PermDoorFront ( <a href="#">see page 134</a> )	
◆	PermDoorFrontPresent ( <a href="#">see page 134</a> )	
◆	PermDoorLeftSide ( <a href="#">see page 135</a> )	
◆	PermDoorLeftSidePresent ( <a href="#">see page 135</a> )	
◆	PermDoorRear ( <a href="#">see page 135</a> )	
◆	PermDoorRearPresent ( <a href="#">see page 136</a> )	
◆	PermDoorRightSide ( <a href="#">see page 136</a> )	
◆	PermDoorRightSidePresent ( <a href="#">see page 136</a> )	
◆	PermFront ( <a href="#">see page 136</a> )	

❖	PermLeftSide (see page 137)	
❖	PermRear (see page 137)	
❖	PermRightSide (see page 137)	
❖	ReferenceLevel (see page 137)	
❖	SnowGutterBars (see page 138)	
❖	SnowPressureExtreme (see page 138)	
❖	SnowPressureNormal (see page 138)	
❖	SnowRedistribution (see page 138)	
❖	StructureAge (see page 139)	
❖	StructureAgeCode (see page 139)	
❖	StructureHeight (see page 139)	
❖	StructureP (see page 139)	
❖	WindAutoCd (see page 140)	
❖	WindCALT (see page 140)	
❖	WindCd (see page 140)	
❖	WindCDIR (see page 141)	
❖	WindCdType (see page 141)	
❖	WindCt (see page 141)	
❖	WindCtAuto (see page 141)	
❖	WindCTEM (see page 142)	
❖	WindE (see page 142)	
❖	WindKT (see page 142)	
❖	WindPressureAutomatic (see page 142)	
❖	WindQref (see page 143)	
❖	WindQref0 (see page 143)	
❖	WindQref0p (see page 143)	
❖	WindSiteType (see page 143)	
❖	WindVref0 (see page 144)	
❖	WindZ0 (see page 144)	
❖	WindZMin (see page 144)	

## Public Methods

	Name	Description
❖	SnowBarCoeffGet (see page 145)	
❖	SnowBarCoeffSet (see page 145)	
❖	WindBarCoeffGet (see page 145)	
❖	WindBarCoeffSet (see page 145)	

## II.1.2.14.2 IRobotSWCodeECParams Fields

The fields of the IRobotSWCodeECParams class are listed here.

### Public Fields

	Name	Description
❖	Altitude (see page 133)	
❖	GlobalCDIR (see page 133)	
❖	GroundType (see page 133)	

◆	LeftWind2NordAngle (☞ see page 133)	
◆	NodalLoadsForAllBars (☞ see page 134)	
◆	NodalLoadsForBarsList (☞ see page 134)	
◆	PermDoorFront (☞ see page 134)	
◆	PermDoorFrontPresent (☞ see page 134)	
◆	PermDoorLeftSide (☞ see page 135)	
◆	PermDoorLeftSidePresent (☞ see page 135)	
◆	PermDoorRear (☞ see page 135)	
◆	PermDoorRearPresent (☞ see page 136)	
◆	PermDoorRightSide (☞ see page 136)	
◆	PermDoorRightSidePresent (☞ see page 136)	
◆	PermFront (☞ see page 136)	
◆	PermLeftSide (☞ see page 137)	
◆	PermRear (☞ see page 137)	
◆	PermRightSide (☞ see page 137)	
◆	ReferenceLevel (☞ see page 137)	
◆	SnowGutterBars (☞ see page 138)	
◆	SnowPressureExtreme (☞ see page 138)	
◆	SnowPressureNormal (☞ see page 138)	
◆	SnowRedistribution (☞ see page 138)	
◆	StructureAge (☞ see page 139)	
◆	StructureAgeCode (☞ see page 139)	
◆	StructureHeight (☞ see page 139)	
◆	StructureP (☞ see page 139)	
◆	WindAutoCd (☞ see page 140)	
◆	WindCALT (☞ see page 140)	
◆	WindCd (☞ see page 140)	
◆	WindCDIR (☞ see page 141)	
◆	WindCdType (☞ see page 141)	
◆	WindCt (☞ see page 141)	
◆	WindCtAuto (☞ see page 141)	
◆	WindCTEM (☞ see page 142)	
◆	WindE (☞ see page 142)	
◆	WindKT (☞ see page 142)	
◆	WindPressureAutomatic (☞ see page 142)	
◆	WindQref (☞ see page 143)	
◆	WindQref0 (☞ see page 143)	
◆	WindQref0p (☞ see page 143)	

◆	WindSiteType (see page 143)	
◆	WindVref0 (see page 144)	
◆	WindZ0 (see page 144)	
◆	WindZMin (see page 144)	

#### II.1.2.14.2.1 Altitude

**C++**

```
HRESULT get_Altitude(double* );
HRESULT put_Altitude(double);
```

**C#**

```
public double Altitude { get; set; }
```

**Visual Basic**

```
Public Altitude As Double
```

**Version**

Available since version 4.

#### II.1.2.14.2.2 GlobalCDIR

**C++**

```
HRESULT get_GlobalCDIR(VARIANT_BOOL* );
HRESULT put_GlobalCDIR(VARIANT_BOOL);
```

**C#**

```
public bool GlobalCDIR { get; set; }
```

**Visual Basic**

```
Public GlobalCDIR As Boolean
```

**Version**

Available since version 4.

#### II.1.2.14.2.3 GroundType

**C++**

```
HRESULT get_GroundType(IRobotSWCodeECGroundType* );
HRESULT put_GroundType(IRobotSWCodeECGroundType);
```

**C#**

```
public IRobotSWCodeECGroundType GroundType { get; set; }
```

**Visual Basic**

```
Public GroundType As IRobotSWCodeECGroundType
```

**Version**

Available since version 4.

#### II.1.2.14.2.4 LeftWind2NordAngle

**C++**

```
HRESULT get_LeftWind2NordAngle(double* );
HRESULT put_LeftWind2NordAngle(double);
```

**C#**

```
public double LeftWind2NordAngle { get; set; }
```

**Visual Basic**

```
Public LeftWind2NordAngle As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.5 NodalLoadsForAllBars****C++**

```
HRESULT get_NodalLoadsForAllBars(VARIANT_BOOL* );
HRESULT put_NodalLoadsForAllBars(VARIANT_BOOL);
```

**C#**

```
public bool NodalLoadsForAllBars { get; set; }
```

**Visual Basic**

```
Public NodalLoadsForAllBars As Boolean
```

**Version**

Available since version 4.

**II.1.2.14.2.6 NodalLoadsForBarsList****C++**

```
HRESULT get_NodalLoadsForBarsList(BSTR* );
HRESULT put_NodalLoadsForBarsList(BSTR);
```

**C#**

```
public String NodalLoadsForBarsList { get; set; }
```

**Visual Basic**

```
Public NodalLoadsForBarsList As String
```

**Version**

Available since version 4.

**II.1.2.14.2.7 PermDoorFront****C++**

```
HRESULT get_PermDoorFront(double* );
HRESULT put_PermDoorFront(double);
```

**C#**

```
public double PermDoorFront { get; set; }
```

**Visual Basic**

```
Public PermDoorFront As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.8 PermDoorFrontPresent****C++**

```
HRESULT get_PermDoorFrontPresent(VARIANT_BOOL* );
```

```
HRESULT put_PermDoorFrontPresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorFrontPresent { get; set; }
```

**Visual Basic**

```
Public PermDoorFrontPresent As Boolean
```

**Version**

Available since version 4.

### II.1.2.14.2.9 PermDoorLeftSide

**C++**

```
HRESULT get_PermDoorLeftSide(double*);  
HRESULT put_PermDoorLeftSide(double);
```

**C#**

```
public double PermDoorLeftSide { get; set; }
```

**Visual Basic**

```
Public PermDoorLeftSide As double
```

**Version**

Available since version 4.

### II.1.2.14.2.10 PermDoorLeftSidePresent

**C++**

```
HRESULT get_PermDoorLeftSidePresent(VARIANT_BOOL*);  
HRESULT put_PermDoorLeftSidePresent(VARIANT_BOOL);
```

**C#**

```
public bool PermDoorLeftSidePresent { get; set; }
```

**Visual Basic**

```
Public PermDoorLeftSidePresent As Boolean
```

**Version**

Available since version 4.

### II.1.2.14.2.11 PermDoorRear

**C++**

```
HRESULT get_PermDoorRear(double*);  
HRESULT put_PermDoorRear(double);
```

**C#**

```
public double PermDoorRear { get; set; }
```

**Visual Basic**

```
Public PermDoorRear As double
```

**Version**

Available since version 4.

### II.1.2.14.2.12 PermDoorRearPresent

#### C++

```
HRESULT get_PermDoorRearPresent(VARIANT_BOOL* );
HRESULT put_PermDoorRearPresent(VARIANT_BOOL);
```

#### C#

```
public bool PermDoorRearPresent { get; set; }
```

#### Visual Basic

```
Public PermDoorRearPresent As Boolean
```

#### Version

Available since version 4.

### II.1.2.14.2.13 PermDoorRightSide

#### C++

```
HRESULT get_PermDoorRightSide(double* );
HRESULT put_PermDoorRightSide(double);
```

#### C#

```
public double PermDoorRightSide { get; set; }
```

#### Visual Basic

```
Public PermDoorRightSide As Double
```

#### Version

Available since version 4.

### II.1.2.14.2.14 PermDoorRightSidePresent

#### C++

```
HRESULT get_PermDoorRightSidePresent(VARIANT_BOOL* );
HRESULT put_PermDoorRightSidePresent(VARIANT_BOOL);
```

#### C#

```
public bool PermDoorRightSidePresent { get; set; }
```

#### Visual Basic

```
Public PermDoorRightSidePresent As Boolean
```

#### Version

Available since version 4.

### II.1.2.14.2.15 PermFront

#### C++

```
HRESULT get_PermFront(double* );
HRESULT put_PermFront(double);
```

#### C#

```
public double PermFront { get; set; }
```

#### Visual Basic

```
Public PermFront As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.16 PermLeftSide****C++**

```
HRESULT get_PermLeftSide(double*);  
HRESULT put_PermLeftSide(double);
```

**C#**

```
public double PermLeftSide { get; set; }
```

**Visual Basic**

```
Public PermLeftSide As double
```

**Version**

Available since version 4.

**II.1.2.14.2.17 PermRear****C++**

```
HRESULT get_PermRear(double*);  
HRESULT put_PermRear(double);
```

**C#**

```
public double PermRear { get; set; }
```

**Visual Basic**

```
Public PermRear As double
```

**Version**

Available since version 4.

**II.1.2.14.2.18 PermRightSide****C++**

```
HRESULT get_PermRightSide(double*);  
HRESULT put_PermRightSide(double);
```

**C#**

```
public double PermRightSide { get; set; }
```

**Visual Basic**

```
Public PermRightSide As double
```

**Version**

Available since version 4.

**II.1.2.14.2.19 ReferenceLevel****C++**

```
HRESULT get_ReferenceLevel(double*);  
HRESULT put_ReferenceLevel(double);
```

**C#**

```
public double ReferenceLevel { get; set; }
```

**Visual Basic**

```
Public ReferenceLevel As double
```

**Version**

Available since version 4.

**II.1.2.14.2.20 SnowGutterBars****C++**

```
HRESULT get_SnowGutterBars(BSTR* );
HRESULT put_SnowGutterBars(BSTR);
```

**C#**

```
public String SnowGutterBars { get; set; }
```

**Visual Basic**

```
Public SnowGutterBars As String
```

**Version**

Available since version 4.

**II.1.2.14.2.21 SnowPressureExtreme****C++**

```
HRESULT get_SnowPressureExtreme(double* );
HRESULT put_SnowPressureExtreme(double);
```

**C#**

```
public double SnowPressureExtreme { get; set; }
```

**Visual Basic**

```
Public SnowPressureExtreme As double
```

**Version**

Available since version 4.

**II.1.2.14.2.22 SnowPressureNormal****C++**

```
HRESULT get_SnowPressureNormal(double* );
HRESULT put_SnowPressureNormal(double);
```

**C#**

```
public double SnowPressureNormal { get; set; }
```

**Visual Basic**

```
Public SnowPressureNormal As double
```

**Version**

Available since version 4.

**II.1.2.14.2.23 SnowRedistribution****C++**

```
HRESULT get_SnowRedistribution(VARIANT_BOOL* );
HRESULT put_SnowRedistribution(VARIANT_BOOL);
```

**C#**

```
public bool SnowRedistribution { get; set; }
```

**Visual Basic**

```
Public SnowRedistribution As Boolean
```

**Version**

Available since version 4.

**II.1.2.14.2.24 StructureAge****C++**

```
HRESULT get_StructureAge(double*);  
HRESULT put_StructureAge(double);
```

**C#**

```
public double StructureAge { get; set; }
```

**Visual Basic**

```
Public StructureAge As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.25 StructureAgeCode****C++**

```
HRESULT get_StructureAgeCode(BSTR*);  
HRESULT put_StructureAgeCode(BSTR);
```

**C#**

```
public String StructureAgeCode { get; set; }
```

**Visual Basic**

```
Public StructureAgeCode As String
```

**Version**

Available since version 4.

**II.1.2.14.2.26 StructureHeight****C++**

```
HRESULT get_StructureHeight(double*);  
HRESULT put_StructureHeight(double);
```

**C#**

```
public double StructureHeight { get; set; }
```

**Visual Basic**

```
Public StructureHeight As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.27 StructureP****C++**

```
HRESULT get_StructureP(double*);
```

```
HRESULT put_StructureP(double);
```

**C#**

```
public double StructureP { get; set; }
```

**Visual Basic**

```
Public StructureP As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.28 WindAutoCd****C++**

```
HRESULT get_WindAutoCd(VARIANT_BOOL*);  
HRESULT put_WindAutoCd(VARIANT_BOOL);
```

**C#**

```
public bool WindAutoCd { get; set; }
```

**Visual Basic**

```
Public WindAutoCd As Boolean
```

**Version**

Available since version 4.

**II.1.2.14.2.29 WindCALT****C++**

```
HRESULT get_WindCALT(double*);  
HRESULT put_WindCALT(double);
```

**C#**

```
public double WindCALT { get; set; }
```

**Visual Basic**

```
Public WindCALT As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.30 WindCd****C++**

```
HRESULT get_WindCd(double*);  
HRESULT put_WindCd(double);
```

**C#**

```
public double WindCd { get; set; }
```

**Visual Basic**

```
Public WindCd As Double
```

**Version**

Available since version 4.

### II.1.2.14.2.31 WindCDIR

#### C++

```
HRESULT get_WindCDIR(double*);  
HRESULT put_WindCDIR(double);
```

#### C#

```
public double WindCDIR { get; set; }
```

#### Visual Basic

```
Public WindCDIR As double
```

#### Version

Available since version 4.

### II.1.2.14.2.32 WindCdType

#### C++

```
HRESULT get_WindCdType(IRobotSWCodeECCdType*);  
HRESULT put_WindCdType(IRobotSWCodeECCdType);
```

#### C#

```
public IRobotSWCodeECCdType WindCdType { get; set; }
```

#### Visual Basic

```
Public WindCdType As IRobotSWCodeECCdType
```

#### Version

Available since version 4.

### II.1.2.14.2.33 WindCt

#### C++

```
HRESULT get_WindCt(double*);  
HRESULT put_WindCt(double);
```

#### C#

```
public double WindCt { get; set; }
```

#### Visual Basic

```
Public WindCt As double
```

#### Version

Available since version 4.

### II.1.2.14.2.34 WindCtAuto

#### C++

```
HRESULT get_WindCtAuto(VARIANT_BOOL*);  
HRESULT put_WindCtAuto(VARIANT_BOOL);
```

#### C#

```
public bool WindCtAuto { get; set; }
```

#### Visual Basic

```
Public WindCtAuto As Boolean
```

**Version**

Available since version 4.

**II.1.2.14.2.35 WindCTEM****C++**

```
HRESULT get_WindCTEM(double*);  
HRESULT put_WindCTEM(double);
```

**C#**

```
public double WindCTEM { get; set; }
```

**Visual Basic**

```
Public WindCTEM As double
```

**Version**

Available since version 4.

**II.1.2.14.2.36 WindE****C++**

```
HRESULT get_WindE(double*);  
HRESULT put_WindE(double);
```

**C#**

```
public double WindE { get; set; }
```

**Visual Basic**

```
Public WindE As double
```

**Version**

Available since version 4.

**II.1.2.14.2.37 WindKT****C++**

```
HRESULT get_WindKT(double*);  
HRESULT put_WindKT(double);
```

**C#**

```
public double WindKT { get; set; }
```

**Visual Basic**

```
Public WindKT As double
```

**Version**

Available since version 4.

**II.1.2.14.2.38 WindPressureAutomatic****C++**

```
HRESULT get_WindPressureAutomatic(VARIANT_BOOL*);  
HRESULT put_WindPressureAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool WindPressureAutomatic { get; set; }
```

**Visual Basic**

```
Public WindPressureAutomatic As Boolean
```

**Version**

Available since version 4.

**II.1.2.14.2.39 WindQref****C++**

```
HRESULT get_WindQref(double*);  
HRESULT put_WindQref(double);
```

**C#**

```
public double WindQref { get; set; }
```

**Visual Basic**

```
Public WindQref As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.40 WindQref0****C++**

```
HRESULT get_WindQref0(double*);  
HRESULT put_WindQref0(double);
```

**C#**

```
public double WindQref0 { get; set; }
```

**Visual Basic**

```
Public WindQref0 As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.41 WindQref0p****C++**

```
HRESULT get_WindQref0p(double*);  
HRESULT put_WindQref0p(double);
```

**C#**

```
public double WindQref0p { get; set; }
```

**Visual Basic**

```
Public WindQref0p As Double
```

**Version**

Available since version 4.

**II.1.2.14.2.42 WindSiteType****C++**

```
HRESULT get_WindSiteType(IRobotSWCodeECSiteType*);  
HRESULT put_WindSiteType(IRobotSWCodeECSiteType);
```

**C#**

```
public IRobotSWCodeECSiteType WindSiteType { get; set; }
```

**Visual Basic**

```
Public WindSiteType As IRobotSWCodeECSiteType
```

**Version**

Available since version 4.

**II.1.2.14.2.43 WindVref0****C++**

```
HRESULT get_WindVref0(double*);  
HRESULT put_WindVref0(double);
```

**C#**

```
public double WindVref0 { get; set; }
```

**Visual Basic**

```
Public WindVref0 As double
```

**Version**

Available since version 4.

**II.1.2.14.2.44 WindZ0****C++**

```
HRESULT get_WindZ0(double*);  
HRESULT put_WindZ0(double);
```

**C#**

```
public double WindZ0 { get; set; }
```

**Visual Basic**

```
Public WindZ0 As double
```

**Version**

Available since version 4.

**II.1.2.14.2.45 WindZMin****C++**

```
HRESULT get_WindZMin(double*);  
HRESULT put_WindZMin(double);
```

**C#**

```
public double WindZMin { get; set; }
```

**Visual Basic**

```
Public WindZMin As double
```

**Version**

Available since version 4.

**II.1.2.14.3 IRobotSWCodeECParams Methods**

The methods of the IRobotSWCodeECParams class are listed here.

## Public Methods

	Name	Description
⊕	SnowBarCoeffGet (see page 145)	
⊕	SnowBarCoeffSet (see page 145)	
⊕	WindBarCoeffGet (see page 145)	
⊕	WindBarCoeffSet (see page 145)	

### II.1.2.14.3.1 SnowBarCoeffGet

#### C++

```
HRESULT SnowBarCoeffGet(long _bar_num, double* ret);
```

#### C#

```
public double SnowBarCoeffGet(long _bar_num);
```

#### Visual Basic

```
Public Function SnowBarCoeffGet(_bar_num As long) As double
```

#### Version

Available since version 4.

### II.1.2.14.3.2 SnowBarCoeffSet

#### C++

```
HRESULT SnowBarCoeffSet(long _bar_num, double _coeff);
```

#### C#

```
public void SnowBarCoeffSet(long _bar_num, double _coeff);
```

#### Visual Basic

```
Public Sub SnowBarCoeffSet(_bar_num As long, _coeff As double)
```

#### Version

Available since version 4.

### II.1.2.14.3.3 WindBarCoeffGet

#### C++

```
HRESULT WindBarCoeffGet(long _bar_num, double* ret);
```

#### C#

```
public double WindBarCoeffGet(long _bar_num);
```

#### Visual Basic

```
Public Function WindBarCoeffGet(_bar_num As long) As double
```

#### Version

Available since version 4.

### II.1.2.14.3.4 WindBarCoeffSet

#### C++

```
HRESULT WindBarCoeffSet(long _bar_num, double _coeff);
```

**C#**

```
public void WindBarCoeffSet(long _bar_num, double _coeff);
```

**Visual Basic**

```
Public Sub WindBarCoeffSet(_bar_num As long, _coeff As double)
```

**Version**

Available since version 4.

**II.1.2.15 IRobotSWCodeECSiteType****C++**

```
enum IRobotSWCodeECSiteType;
```

**C#**

```
public enum IRobotSWCodeECSiteType;
```

**Visual Basic**

```
Public Enum IRobotSWCodeECSiteType
```

**Members**

<b>Members</b>	<b>Description</b>
I_SWCECST_TYPE_I = 1	Available since version 4.
I_SWCECST_TYPE_II = 2	Available since version 4.
I_SWCECST_TYPE_III = 3	Available since version 4.
I_SWCECST_TYPE_IV = 4	Available since version 4.
I_SWCECST_TYPE_V = 5	Available since version 4.
I_SWCECST_TYPE_VI = 6	Available since version 4.

**Version**

Available since version 4.

**II.1.2.16 IRobotSWCodeECGroundType****C++**

```
enum IRobotSWCodeECGroundType;
```

**C#**

```
public enum IRobotSWCodeECGroundType;
```

**Visual Basic**

```
Public Enum IRobotSWCodeECGroundType
```

**Members**

<b>Members</b>	<b>Description</b>
I_SWCECGT_TYPE_I = 1	Available since version 4.
I_SWCECGT_TYPE_II = 2	Available since version 4.
I_SWCECGT_TYPE_III = 3	Available since version 4.
I_SWCECGT_TYPE_IV = 4	Available since version 4.

**Version**

Available since version 4.

## II.1.2.17 IRobotSWCodeECCdType

### C++

```
enum IRobotSWCodeECCdType;
```

### C#

```
public enum IRobotSWCodeECCdType;
```

### Visual Basic

```
Public Enum IRobotSWCodeECCdType
```

### Members

Members	Description
I_SWCECCT_TYPE_I = 1	Available since version 4.
I_SWCECCT_TYPE_II = 2	Available since version 4.
I_SWCECCT_TYPE_III = 3	Available since version 4.

### Version

Available since version 4.

## II.1.3 Code combinations

### Enumerations

	Name	Description
	IRobotCodeCmbFlag ( <a href="#">see page 155</a> )	
	IRobotCodeCmbActionCoeffType ( <a href="#">see page 162</a> )	Coefficients used for describing actions. .
	IRobotCodeCmbCombPartType ( <a href="#">see page 165</a> )	
	IRobotCodeCmbCombType ( <a href="#">see page 166</a> )	
	IRobotCodeCmbDecidingValueType ( <a href="#">see page 173</a> )	
	IRobotCodeCmbOperator ( <a href="#">see page 176</a> )	Available operators used for defining relations between cases and case groups. .
	IRobotCodeCmbGenerationType ( <a href="#">see page 186</a> )	

### Interfaces

	Name	Description
	IRobotCodeCombinationEngine ( <a href="#">see page 148</a> )	
	IRobotCodeCombination ( <a href="#">see page 149</a> )	Combination of load cases defined according to selected code.
	IRobotCodeCmbComponentMngr ( <a href="#">see page 152</a> )	Access interface to definitions of code combination components. .
	IRobotCodeCmbFactor ( <a href="#">see page 153</a> )	
	IRobotCodeCmbComponent ( <a href="#">see page 154</a> )	
	IRobotCodeCmbRegulations ( <a href="#">see page 156</a> )	Definition of the regulations describing the manner of generating code-defined loads. .

	IRobotCodeCmbActionServer (see page 158)	Definition of actions. .
	IRobotCodeCmbCombs (see page 163)	Combination definition.
	IRobotCodeCmbGenerationParams (see page 166)	Parameters of code combination generation.
	IRobotCodeCmbGroup (see page 173)	Definition of load case groups.
	IRobotCodeCmbGroupServer (see page 176)	Server managing the groups of cases.
	IRobotCodeCmbGroupRelation (see page 179)	Definition of relations between groups. Definition of relations between groups can be illustrated by means of the matrix whose elements are group numbers. All groups whose numbers are located in the same matrix row are in the same relation with each other (either AND or AND_OR). Whereas among the matrix rows the relation EXCLUSIVE_OR occurs. .
	IRobotCodeCmbGroupRelationServer (see page 183)	Server managing relations among groups. .
	IRobotCodeCmbActiveCaseInfo (see page 184)	

### II.1.3.1 IRobotCodeCombinationEngine

#### Class Hierarchy

##### C++

```
interface IRobotCodeCombinationEngine : IDispatch;
```

##### C#

```
public interface IRobotCodeCombinationEngine;
```

#### Visual Basic

```
Public Interface IRobotCodeCombinationEngine
```

### II.1.3.1.1 IRobotCodeCombinationEngine Members

The following tables list the members exposed by IRobotCodeCombinationEngine.

#### Public Fields

	Name	Description
	Params (see page 149)	Parameters of code combination generation Available since version 1.7.

#### Public Methods

	Name	Description
	Generate (see page 149)	Function generates code-defined loads according to the selected code and current settings of generation parameters. Available since version 1.7.

### II.1.3.1.2 IRobotCodeCombinationEngine Fields

The fields of the IRobotCodeCombinationEngine class are listed here.

#### Public Fields

	Name	Description
	Params (see page 149)	Parameters of code combination generation Available since version 1.7.

### II.1.3.1.2.1 Params

#### C++

```
HRESULT get_Params( IRobotCodeCmbGenerationParams** );
```

#### C#

```
public IRobotCodeCmbGenerationParams Params { get; }
```

#### Visual Basic

```
Public ReadOnly Params As IRobotCodeCmbGenerationParams
```

#### Description

Parameters of code combination generation Available since version 1.7.

### II.1.3.1.3 IRobotCodeCombinationEngine Methods

The methods of the IRobotCodeCombinationEngine class are listed here.

#### Public Methods

	Name	Description
💡	Generate (see page 149)	Function generates code-defined loads according to the selected code and current settings of generation parameters. Available since version 1.7.

### II.1.3.1.3.1 Generate

#### C++

```
HRESULT Generate();
```

#### C#

```
public void Generate();
```

#### Visual Basic

```
Public Sub Generate()
```

#### Description

Function generates code-defined loads according to the selected code and current settings of generation parameters. Available since version 1.7.

### II.1.3.2 IRobotCodeCombination

#### Class Hierarchy

#### C++

```
interface IRobotCodeCombination : IRobotCase;
```

#### C#

```
public interface IRobotCodeCombination : IRobotCase;
```

#### Visual Basic

```
Public Interface IRobotCodeCombination
```

#### Description

Combination of load cases defined according to selected code.

### II.1.3.2.1 IRobotCodeCombination Members

The following tables list the members exposed by IRobotCodeCombination.

#### Public Fields

	Name	Description
◆	AnalyzeType ( [ see page 405 )	Analysis type.
◆	CombinationType ( [ see page 150 )	Combination type Available since version 1.7.
◆	Components ( [ see page 150 )	Set of component combinations Available since version 1.7.
◆	Flag ( [ see page 151 )	Available since version 1.7.
◆	Name ( [ see page 405 )	User-defined case name .
◆	Nature ( [ see page 405 )	Case nature .
◆	Number ( [ see page 406 )	User-defined number linked with the load case .
◆	Type ( [ see page 406 )	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .
◆	Uniqueld ( [ see page 151 )	Unique combination identifier Available since version 1.7.

#### Public Methods

	Name	Description
◆	FindByFlag ( [ see page 151 )	Function returns a user's number for the code combination linked with the combination that has a determined flag. If such a combination is not found, then zero value is returned. Available since version 1.7.

### II.1.3.2.2 IRobotCodeCombination Fields

The fields of the IRobotCodeCombination class are listed here.

#### Public Fields

	Name	Description
◆	CombinationType ( [ see page 150 )	Combination type Available since version 1.7.
◆	Components ( [ see page 150 )	Set of component combinations Available since version 1.7.
◆	Flag ( [ see page 151 )	Available since version 1.7.
◆	Uniqueld ( [ see page 151 )	Unique combination identifier Available since version 1.7.

### II.1.3.2.2.1 CombinationType

#### C++

```
HRESULT get_CombinationType( IRobotCombinationType* );
HRESULT put_CombinationType( IRobotCombinationType );
```

#### C#

```
public IRobotCombinationType CombinationType { get; set; }
```

#### Visual Basic

```
Public CombinationType As IRobotCombinationType
```

#### Description

Combination type Available since version 1.7.

### II.1.3.2.2 Components

#### C++

```
HRESULT get_Components( IRobotCodeCmbComponentMngr** );
```

**C#**

```
public IRobotCodeCmbComponentMngr Components { get; }
```

**Visual Basic**

```
Public ReadOnly Components As IRobotCodeCmbComponentMngr
```

**Description**

Set of component combinations Available since version 1.7.

**II.1.3.2.2.3 Flag****C++**

```
HRESULT get_Flag(IRobotCodeCmbFlag*);
```

**C#**

```
public IRobotCodeCmbFlag Flag { get; }
```

**Visual Basic**

```
Public ReadOnly Flag As IRobotCodeCmbFlag
```

**Description**

Available since version 1.7.

**II.1.3.2.2.4 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique combination identifier Available since version 1.7.

**II.1.3.2.3 IRobotCodeCombination Methods**

The methods of the IRobotCodeCombination class are listed here.

**Public Methods**

	Name	Description
💡	FindByFlag (see page 151)	Function returns a user's number for the code combination linked with the combination that has a determined flag. If such a combination is not found, then zero value is returned. Available since version 1.7.

**II.1.3.2.3.1 FindByFlag****C++**

```
HRESULT FindByFlag(IRobotCodeCmbFlag _cmb_flag, long* ret);
```

**C#**

```
public long FindByFlag(IRobotCodeCmbFlag _cmb_flag);
```

**Visual Basic**

```
Public Function FindByFlag(_cmb_flag As IRobotCodeCmbFlag) As long
```

**Description**

Function returns a user's number for the code combination linked with the combination that has a determined flag. If such a combination is not found, then zero value is returned. Available since version 1.7.

**II.1.3.3 IRobotCodeCmbComponentMngr****Class Hierarchy****C++**

```
interface IRobotCodeCmbComponentMngr : IDispatch;
```

**C#**

```
public interface IRobotCodeCmbComponentMngr;
```

**Visual Basic**

```
Public Interface IRobotCodeCmbComponentMngr
```

**Description**

Access interface to definitions of code combination components. .

**II.1.3.3.1 IRobotCodeCmbComponentMngr Members**

The following tables list the members exposed by IRobotCodeCmbComponentMngr.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Count ( <a href="#">see page 152</a> )	Number of combinations Available since version 1.7.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Get ( <a href="#">see page 153</a> )	Function returns a set of pairs (case number, coefficient) defining the selected combination. Available since version 1.7.

**II.1.3.3.2 IRobotCodeCmbComponentMngr Fields**

The fields of the IRobotCodeCmbComponentMngr class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Count ( <a href="#">see page 152</a> )	Number of combinations Available since version 1.7.

**II.1.3.3.2.1 Count****C++**

```
HRESULT get_Count(long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of combinations Available since version 1.7.

### II.1.3.3.3 IRobotCodeCmbComponentMngr Methods

The methods of the IRobotCodeCmbComponentMngr class are listed here.

#### Public Methods

	Name	Description
💡	Get (see page 153)	Function returns a set of pairs (case number, coefficient) defining the selected combination. Available since version 1.7.

#### II.1.3.3.3.1 Get

##### C++

```
HRESULT Get(long _cmb_idx, IRobotCodeCmbComponent** ret);
```

##### C#

```
public IRobotCodeCmbComponent Get(long _cmb_idx);
```

##### Visual Basic

```
Public Function Get(_cmb_idx As long) As IRobotCodeCmbComponent
```

#### Description

Function returns a set of pairs (case number, coefficient) defining the selected combination. Available since version 1.7.

### II.1.3.4 IRobotCodeCmbFactor

#### Class Hierarchy

##### C++

```
interface IRobotCodeCmbFactor : IDispatch;
```

##### C#

```
public interface IRobotCodeCmbFactor;
```

##### Visual Basic

```
Public Interface IRobotCodeCmbFactor
```

#### II.1.3.4.1 IRobotCodeCmbFactor Members

The following tables list the members exposed by IRobotCodeCmbFactor.

#### Public Fields

	Name	Description
💡	CaseNumber (see page 154)	Available since version 1.7.
💡	Factor (see page 154)	Available since version 1.7.

#### II.1.3.4.2 IRobotCodeCmbFactor Fields

The fields of the IRobotCodeCmbFactor class are listed here.

#### Public Fields

	Name	Description
💡	CaseNumber (see page 154)	Available since version 1.7.
💡	Factor (see page 154)	Available since version 1.7.

### II.1.3.4.2.1 CaseNumber

#### C++

```
HRESULT get_CaseNumber(long*);
```

#### C#

```
public long CaseNumber { get; }
```

#### Visual Basic

```
Public ReadOnly CaseNumber As long
```

#### Description

Available since version 1.7.

### II.1.3.4.2.2 Factor

#### C++

```
HRESULT get_Factor(double*);
```

#### C#

```
public double Factor { get; }
```

#### Visual Basic

```
Public ReadOnly Factor As double
```

#### Description

Available since version 1.7.

### II.1.3.5 IRobotCodeCmbComponent

#### Class Hierarchy

#### C++

```
interface IRobotCodeCmbComponent : IDispatch;
```

#### C#

```
public interface IRobotCodeCmbComponent;
```

#### Visual Basic

```
Public Interface IRobotCodeCmbComponent
```

### II.1.3.5.1 IRobotCodeCmbComponent Members

The following tables list the members exposed by IRobotCodeCmbComponent.

#### Public Fields

	Name	Description
◆	Count (  see page 155)	Available since version 1.7.

#### Public Methods

	Name	Description
◆	Get (  see page 155)	Available since version 1.7.

### II.1.3.5.2 IRobotCodeCmbComponent Fields

The fields of the IRobotCodeCmbComponent class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 155)	Available since version 1.7.

### II.1.3.5.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Available since version 1.7.

## II.1.3.5.3 IRobotCodeCmbComponent Methods

The methods of the IRobotCodeCmbComponent class are listed here.

### Public Methods

	Name	Description
◆	Get (see page 155)	Available since version 1.7.

### II.1.3.5.3.1 Get

#### C++

```
HRESULT Get(long _idx, IRobotCodeCmbFactor** ret);
```

#### C#

```
public IRobotCodeCmbFactor Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As IRobotCodeCmbFactor
```

#### Description

Available since version 1.7.

## II.1.3.6 IRobotCodeCmbFlag

#### C++

```
enum IRobotCodeCmbFlag;
```

#### C#

```
public enum IRobotCodeCmbFlag;
```

#### Visual Basic

```
Public Enum IRobotCodeCmbFlag
```

### Members

Members	Description
I_CCF_MAIN = 3	Available since version 1.7.

I_CCF_MAX = 0	Available since version 1.7.
I_CCF_MIN = 1	Available since version 1.7.

### II.1.3.7 IRobotCodeCmbRegulations

#### Class Hierarchy

##### C++

```
interface IRobotCodeCmbRegulations : IDispatch;
```

##### C#

```
public interface IRobotCodeCmbRegulations;
```

#### Visual Basic

```
Public Interface IRobotCodeCmbRegulations
```

#### Description

Definition of the regulations describing the manner of generating code-defined loads. .

### II.1.3.7.1 IRobotCodeCmbRegulations Members

The following tables list the members exposed by IRobotCodeCmbRegulations.

#### Public Fields

	Name	Description
❖	Actions (☞ see page 157)	Actions described in the regulations Available since version 1.7.
❖	CodeName (☞ see page 157)	Name of the regulations Available since version 1.7.
❖	Combinations (☞ see page 157)	Combinations described in the regulations Available since version 1.7.
❖	IsEuroCode (☞ see page 157)	Flag indicating if the regulations are compatible with the EUROCODE standard Available since version 1.7.
❖	MaterialType (☞ see page 158)	Material type; material type is recognized by its name - in case when a non-standard material name is used, its type will not be recognized correctly and the value designating "OTHER" unknown material type will be returned Available since version 1.7.
❖	Version (☞ see page 158)	Text describing the version of the regulations Available since version 1.7.

### II.1.3.7.2 IRobotCodeCmbRegulations Fields

The fields of the IRobotCodeCmbRegulations class are listed here.

#### Public Fields

	Name	Description
❖	Actions (☞ see page 157)	Actions described in the regulations Available since version 1.7.
❖	CodeName (☞ see page 157)	Name of the regulations Available since version 1.7.
❖	Combinations (☞ see page 157)	Combinations described in the regulations Available since version 1.7.
❖	IsEuroCode (☞ see page 157)	Flag indicating if the regulations are compatible with the EUROCODE standard Available since version 1.7.
❖	MaterialType (☞ see page 158)	Material type; material type is recognized by its name - in case when a non-standard material name is used, its type will not be recognized correctly and the value designating "OTHER" unknown material type will be returned Available since version 1.7.
❖	Version (☞ see page 158)	Text describing the version of the regulations Available since version 1.7.

### II.1.3.7.2.1 Actions

#### C++

```
HRESULT get_Actions(IRobotCodeCmbActionServer**);
```

#### C#

```
public IRobotCodeCmbActionServer Actions { get; }
```

#### Visual Basic

```
Public ReadOnly Actions As IRobotCodeCmbActionServer
```

#### Description

Actions described in the regulations Available since version 1.7.

### II.1.3.7.2.2 CodeName

#### C++

```
HRESULT get_CodeName(BSTR*);  
HRESULT put_CodeName(BSTR);
```

#### C#

```
public String CodeName { get; set; }
```

#### Visual Basic

```
Public CodeName As String
```

#### Description

Name of the regulations Available since version 1.7.

### II.1.3.7.2.3 Combinations

#### C++

```
HRESULT get_Combinations(IRobotCodeCmbCombs**);
```

#### C#

```
public IRobotCodeCmbCombs Combinations { get; }
```

#### Visual Basic

```
Public ReadOnly Combinations As IRobotCodeCmbCombs
```

#### Description

Combinations described in the regulations Available since version 1.7.

### II.1.3.7.2.4 IsEuroCode

#### C++

```
HRESULT get_IsEuroCode(VARIANT_BOOL*);  
HRESULT put_IsEuroCode(VARIANT_BOOL);
```

#### C#

```
public bool IsEuroCode { get; set; }
```

#### Visual Basic

```
Public IsEuroCode As Boolean
```

#### Description

Flag indicating if the regulations are compatible with the EUROCODE standard Available since version 1.7.

### II.1.3.7.2.5 MaterialType

#### C++

```
HRESULT get_MaterialType(IRobotMaterialType* );
HRESULT put_MaterialType(IRobotMaterialType);
```

#### C#

```
public IRobotMaterialType MaterialType { get; set; }
```

#### Visual Basic

```
Public MaterialType As IRobotMaterialType
```

#### Description

Material type; material type is recognized by its name - in case when a non-standard material name is used, its type will not be recognized correctly and the value designating "OTHER" unknown material type will be returned Available since version 1.7.

### II.1.3.7.2.6 Version

#### C++

```
HRESULT get_Version(BSTR* );
HRESULT put_Version(BSTR);
```

#### C#

```
public String Version { get; set; }
```

#### Visual Basic

```
Public Version As String
```

#### Description

Text describing the version of the regulations Available since version 1.7.

### II.1.3.8 IRobotCodeCmbActionServer

#### Class Hierarchy

#### C++

```
interface IRobotCodeCmbActionServer : IDispatch;
```

#### C#

```
public interface IRobotCodeCmbActionServer;
```

#### Visual Basic

```
Public Interface IRobotCodeCmbActionServer
```

#### Description

Definition of actions..

### II.1.3.8.1 IRobotCodeCmbActionServer Members

The following tables list the members exposed by IRobotCodeCmbActionServer.

#### Public Fields

	Name	Description
Count (see page 159)	Count (see page 159)	Number of defined actions Available since version 1.7.

## Public Methods

	Name	Description
💡	GetCoeff (🔗 see page 160)	Function returns the value of a selected coefficient for a selected action. Available since version 1.7.
💡	GetName (🔗 see page 160)	Function returns the name of the indicated action. Available since version 1.7.
💡	GetNature (🔗 see page 160)	Function returns the nature of the selected action Available since version 1.7.
💡	New (🔗 see page 161)	Function defines a new action and returns its index which should be used while determining the values of individual coefficients. Available since version 1.7.
💡	Remove (🔗 see page 161)	Function deletes the indicated action definition. Available since version 1.7.
💡	SetCoeff (🔗 see page 161)	Function sets the value of a specified coefficient for a specified action. Available since version 1.7.
💡	SetName (🔗 see page 161)	Function sets (changes) the name of the indicated action. Available since version 1.7.
💡	SetNature (🔗 see page 162)	Function changes the nature of a selected action. Available since version 1.7.

### II.1.3.8.2 IRobotCodeCmbActionServer Fields

The fields of the IRobotCodeCmbActionServer class are listed here.

#### Public Fields

	Name	Description
💡	Count (🔗 see page 159)	Number of defined actions Available since version 1.7.

### II.1.3.8.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of defined actions Available since version 1.7.

### II.1.3.8.3 IRobotCodeCmbActionServer Methods

The methods of the IRobotCodeCmbActionServer class are listed here.

#### Public Methods

	Name	Description
💡	GetCoeff (🔗 see page 160)	Function returns the value of a selected coefficient for a selected action. Available since version 1.7.
💡	GetName (🔗 see page 160)	Function returns the name of the indicated action. Available since version 1.7.
💡	GetNature (🔗 see page 160)	Function returns the nature of the selected action Available since version 1.7.

	New (see page 161)	Function defines a new action and returns its index which should be used while determining the values of individual coefficients. Available since version 1.7.
	Remove (see page 161)	Function deletes the indicated action definition. Available since version 1.7.
	SetCoeff (see page 161)	Function sets the value of a specified coefficient for a specified action. Available since version 1.7.
	SetName (see page 161)	Function sets (changes) the name of the indicated action. Available since version 1.7.
	SetNature (see page 162)	Function changes the nature of a selected action. Available since version 1.7.

### II.1.3.8.3.1 GetCoeff

#### C++

```
HRESULT GetCoeff(long _idx, IRobotCodeCmbActionCoeffType _coeff_type, double* ret);
```

#### C#

```
public double GetCoeff(long _idx, IRobotCodeCmbActionCoeffType _coeff_type);
```

#### Visual Basic

```
Public Function GetCoeff(_idx As long, _coeff_type As IRobotCodeCmbActionCoeffType) As double
```

#### Description

Function returns the value of a selected coefficient for a selected action. Available since version 1.7.

### II.1.3.8.3.2 GetName

#### C++

```
HRESULT GetName(long _idx, BSTR* ret);
```

#### C#

```
public String GetName(long _idx);
```

#### Visual Basic

```
Public Function GetName(_idx As long) As String
```

#### Description

Function returns the name of the indicated action. Available since version 1.7.

### II.1.3.8.3.3 GetNature

#### C++

```
HRESULT GetNature(long _idx, IRobotCaseNature* ret);
```

#### C#

```
public IRobotCaseNature GetNature(long _idx);
```

#### Visual Basic

```
Public Function GetNature(_idx As long) As IRobotCaseNature
```

#### Description

Function returns the nature of the selected action Available since version 1.7.

### II.1.3.8.3.4 New

**C++**

```
HRESULT New(IRobotCaseNature _case_nature, BSTR _name, long* ret);
```

**C#**

```
public long New(IRobotCaseNature _case_nature, String _name);
```

**Visual Basic**

```
Public Function New(_case_nature As IRobotCaseNature, _name As String) As long
```

**Description**

Function defines a new action and returns its index which should be used while determining the values of individual coefficients. Available since version 1.7.

### II.1.3.8.3.5 Remove

**C++**

```
HRESULT Remove(long _idx);
```

**C#**

```
public void Remove(long _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As long)
```

**Description**

Function deletes the indicated action definition. Available since version 1.7.

### II.1.3.8.3.6 SetCoeff

**C++**

```
HRESULT SetCoeff(long _idx, IRobotCodeCmbActionCoeffType _coeff_type, double _coeff_val);
```

**C#**

```
public void SetCoeff(long _idx, IRobotCodeCmbActionCoeffType _coeff_type, double _coeff_val);
```

**Visual Basic**

```
Public Sub SetCoeff(_idx As long, _coeff_type As IRobotCodeCmbActionCoeffType, _coeff_val As double)
```

**Description**

Function sets the value of a specified coefficient for a specified action. Available since version 1.7.

### II.1.3.8.3.7 SetName

**C++**

```
HRESULT SetName(long _idx, BSTR _name);
```

**C#**

```
public void SetName(long _idx, String _name);
```

**Visual Basic**

```
Public Sub SetName(_idx As long, _name As String)
```

**Description**

Function sets (changes) the name of the indicated action. Available since version 1.7.

**II.1.3.8.3.8 SetNature****C++**

```
HRESULT SetNature(long _idx, IRobotCaseNature _case_nat);
```

**C#**

```
public void SetNature(long _idx, IRobotCaseNature _case_nat);
```

**Visual Basic**

```
Public Sub SetNature(_idx As long, _case_nat As IRobotCaseNature)
```

**Description**

Function changes the nature of a selected action. Available since version 1.7.

**II.1.3.9 IRobotCodeCmbActionCoeffType****C++**

```
enum IRobotCodeCmbActionCoeffType;
```

**C#**

```
public enum IRobotCodeCmbActionCoeffType;
```

**Visual Basic**

```
Public Enum IRobotCodeCmbActionCoeffType
```

**Members**

Members	Description
I_CCACT_GU_MAX = 0	Partial safety factor for ULS (maximum) Available since version 1.7.
I_CCACT_GU_MIN = 1	Partial safety factor for ULS (minimum) Available since version 1.7.
I_CCACT_GS = 2	Partial safety factor for SLS Available since version 1.7.
I_CCACT_GA = 3	Partial safety factor for accidental state Available since version 1.7.
I_CCACT_PSI0_1 = 4	Simultaneity coefficient for the first load Available since version 1.7.
I_CCACT_PSI0_2 = 5	Simultaneity coefficient for the second load Available since version 1.7.
I_CCACT_PSI0_3 = 6	Simultaneity coefficient for the third load Available since version 1.7.
I_CCACT_PSI0_N = 7	Simultaneity coefficient for the fourth load Available since version 1.7.
I_CCACT_PSI1 = 8	Factor for long-term loads Available since version 1.7.
I_CCACT_PSI2_1 = 9	Reduction factor for accidental combination Available since version 1.7.
I_CCACT_PSI2_N = 10	Reduction factor Available since version 1.7.
I_CCACT_PSIK = 11	Coefficient for second load and next loads Available since version 1.7.

I_CCACT_KSI_MAX = 12	Available since version 1.7.
I_CCACT_KSI_MIN = 13	Available since version 1.7.

**Description**

Coefficients used for describing actions. .

**II.1.3.10 IRobotCodeCmbCombs****Class Hierarchy****C++**

```
interface IRobotCodeCmbCombs : IDispatch;
```

**C#**

```
public interface IRobotCodeCmbCombs;
```

**Visual Basic**

```
Public Interface IRobotCodeCmbCombs
```

**Description**

Combination definition.

**II.1.3.10.1 IRobotCodeCmbCombs Members**

The following tables list the members exposed by IRobotCodeCmbCombs.

**Public Fields**

	Name	Description
◆	Count ( <a href="#">see page 163</a> )	Number of combinations Available since version 1.7.

**Public Methods**

	Name	Description
◆	Get ( <a href="#">see page 164</a> )	Function returns the definition of selected combination. Available since version 1.7.
◆	New ( <a href="#">see page 164</a> )	Function creates a new combination and adds its definition at the end of the list. Index of the newly-created combination is returned. Available since version 1.7.
◆	Remove ( <a href="#">see page 165</a> )	Function deletes the definition of the selected combination. Available since version 1.7.
◆	Set ( <a href="#">see page 165</a> )	Function changes the definition of the selected combination. Available since version 1.7.

**II.1.3.10.2 IRobotCodeCmbCombs Fields**

The fields of the IRobotCodeCmbCombs class are listed here.

**Public Fields**

	Name	Description
◆	Count ( <a href="#">see page 163</a> )	Number of combinations Available since version 1.7.

**II.1.3.10.2.1 Count****C++**

```
HRESULT get_Count(long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of combinations Available since version 1.7.

**II.1.3.10.3 IRobotCodeCmbCombs Methods**

The methods of the IRobotCodeCmbCombs class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Get (🔗 see page 164)	Function returns the definition of selected combination. Available since version 1.7.
💡	New (🔗 see page 164)	Function creates a new combination and adds its definition at the end of the list. Index of the newly-created combination is returned. Available since version 1.7.
💡	Remove (🔗 see page 165)	Function deletes the definition of the selected combination. Available since version 1.7.
💡	Set (🔗 see page 165)	Function changes the definition of the selected combination. Available since version 1.7.

**II.1.3.10.3.1 Get****C++**

```
HRESULT Get(long _idx, RobotLimitState* _limit_state, RobotCodeCmbCombType* _cmb_type, BSTR _name, short* _dead_coef, short* _live_coef, short* _acc_coef, short* _seis_coef);
```

**C#**

```
public void Get(long _idx, RobotLimitState* _limit_state, RobotCodeCmbCombType* _cmb_type, String _name, short* _dead_coef, short* _live_coef, short* _acc_coef, short* _seis_coef);
```

**Visual Basic**

```
Public Sub Get(_idx As Long, ByRef _limit_state As RobotLimitState*, ByRef _cmb_type As RobotCodeCmbCombType*, ByRef _name As String, ByRef _dead_coef As Short*, ByRef _live_coef As Short*, ByRef _acc_coef As Short*, ByRef _seis_coef As Short*)
```

**Description**

Function returns the definition of selected combination. Available since version 1.7.

**II.1.3.10.3.2 New****C++**

```
HRESULT New(IRobotLimitState _lim_state, IRobotCodeCmbCombType _cmb_type, BSTR _name, short _dead_coeff, short _live_coeff, short _acc_coeff, short _seis_coeff, long* ret);
```

**C#**

```
public long New(IRobotLimitState _lim_state, IRobotCodeCmbCombType _cmb_type, String _name, short _dead_coeff, short _live_coeff, short _acc_coeff, short _seis_coeff);
```

**Visual Basic**

```
Public Function New(_lim_state As IRobotLimitState, _cmb_type As IRobotCodeCmbCombType, _name As String, _dead_coeff As Short, _live_coeff As Short, _acc_coeff As Short, _seis_coeff As Short) As Long
```

## Description

Function creates a new combination and adds its definition at the end of the list. Index of the newly-created combination is returned. Available since version 1.7.

### II.1.3.10.3.3 Remove

#### C++

```
HRESULT Remove(long _idx);
```

#### C#

```
public void Remove(long _idx);
```

#### Visual Basic

```
Public Sub Remove(_idx As long)
```

## Description

Function deletes the definition of the selected combination. Available since version 1.7.

### II.1.3.10.3.4 Set

#### C++

```
HRESULT Set(long _idx, IRobotLimitState _limit_state, IRobotCodeCmbCombType _cmb_type, BSTR _name, short _dead_coef, short _live_coef, short _acc_coef, short _seis_coef);
```

#### C#

```
public void Set(long _idx, IRobotLimitState _limit_state, IRobotCodeCmbCombType _cmb_type, String _name, short _dead_coef, short _live_coef, short _acc_coef, short _seis_coef);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _limit_state As IRobotLimitState, _cmb_type As IRobotCodeCmbCombType, _name As String, _dead_coef As short, _live_coef As short, _acc_coef As short, _seis_coef As short)
```

## Description

Function changes the definition of the selected combination. Available since version 1.7.

### II.1.3.11 IRobotCodeCmbCombPartType

#### C++

```
enum IRobotCodeCmbCombPartType;
```

#### C#

```
public enum IRobotCodeCmbCombPartType;
```

#### Visual Basic

```
Public Enum IRobotCodeCmbCombPartType
```

## Members

Members	Description
I_CCCPT_DEAD = 0	Available since version 1.7.
I_CCCPT_LIVE = 1	Available since version 1.7.
I_CCCPT_ACCIDENTAL = 2	Available since version 1.7.
I_CCCPT_SEISMIC = 3	Available since version 1.7.

### II.1.3.12 IRobotCodeCmbCombType

**C++**

```
enum IRobotCodeCmbCombType;
```

**C#**

```
public enum IRobotCodeCmbCombType;
```

**Visual Basic**

```
Public Enum IRobotCodeCmbCombType
```

**Members**

Members	Description
I_CCCT_STANDARD = 1	Standard combination Available since version 1.7.
I_CCCT_SEISMIC = 2	Seismic combination Available since version 1.7.
I_CCCT_ACCIDENTAL = 3	Accidental combination Available since version 1.7.
I_CCCT_RARE = 4	Rare combination Available since version 1.7.
I_CCCT_FREQUENT = 5	Frequent combination Available since version 1.7.
I_CCCT_QUASI_PERM = 6	Quasi-permanent combination Available since version 1.7.
I_CCCT_USER = 7	User-defined combination Available since version 1.7.
I_CCCT_FUNDAMENTAL = 8	Basic combination Available since version 1.7.
I_CCCT_SIMPLIFIED = 9	Simplified combination Available since version 1.7.
I_CCCT_EXTREMAL = 10	Extreme combination Available since version 1.7.

### II.1.3.13 IRobotCodeCmbGenerationParams

**Class Hierarchy**

**C++**

```
interface IRobotCodeCmbGenerationParams : IDispatch;
```

**C#**

```
public interface IRobotCodeCmbGenerationParams;
```

**Visual Basic**

```
Public Interface IRobotCodeCmbGenerationParams
```

**Description**

Parameters of code combination generation.

#### II.1.3.13.1 IRobotCodeCmbGenerationParams Members

The following tables list the members exposed by IRobotCodeCmbGenerationParams.

**Public Fields**

	Name	Description
❖	ActiveCases (☞ see page 168)	Information about cases participating in a definition of code combinations (object collection of the RobotCodeCmbActiveCaseInfo type) Available since version 1.7.
❖	AllBars (☞ see page 168)	Available since version 1.7.
❖	AllNodes (☞ see page 168)	Available since version 1.7.
❖	BarSel (☞ see page 168)	Bar selection (taken into account only for simplified generation when AllBars (☞ see page 168) are set as zero - False) Available since version 1.7.

◆	ExtremalSnowFactor ( <a href="#">see page 169</a> )	Factor for extreme snow Available since version 1.7.
◆	GenType ( <a href="#">see page 169</a> )	Method of generation of code combinations.
◆	Groups ( <a href="#">see page 169</a> )	Groups Available since version 1.7.
◆	NodeSel ( <a href="#">see page 169</a> )	Node selection (taken into account only for simplified generation when AllNodes ( <a href="#">see page 168</a> ) are set as zero - False) Available since version 1.7.
◆	PointsOnBar ( <a href="#">see page 170</a> )	Number of points along the bar length (it is taken into account only for simplified generation) Available since version 1.7.
◆	Regulations ( <a href="#">see page 170</a> )	Definition of the regulations Available since version 1.7.
◆	Relations ( <a href="#">see page 170</a> )	Relations between groups Available since version 1.7.

**Public Methods**

	Name	Description
◆	IsCombinationSelected ( <a href="#">see page 171</a> )	Function returns a value different from zero (True) if a specified template for generating combination is to be taken into account during generation. Available since version 1.7.
◆	IsCombinationTypeSelected ( <a href="#">see page 171</a> )	Function returns True if templates describing combination of the given type are used during combination generation.
◆	IsDecidingValueSelected ( <a href="#">see page 171</a> )	Function returns a value different from zero (True) if the specified decisive value has been selected. Available since version 1.7.
◆	SelectCombination ( <a href="#">see page 172</a> )	Function enables selection of templates for calculating a combination among all the templates defined in the regulations. Available since version 1.7.
◆	SelectCombinationType ( <a href="#">see page 172</a> )	Function allows for a selection of templates describing combination of the given type from among all templates defined in the regulation.
◆	SelectDecidingValue ( <a href="#">see page 172</a> )	Function enables selection of decisive values. Available since version 1.7.

**II.1.3.13.2 IRobotCodeCmbGenerationParams Fields**

The fields of the IRobotCodeCmbGenerationParams class are listed here.

**Public Fields**

	Name	Description
◆	ActiveCases ( <a href="#">see page 168</a> )	Information about cases participating in a definition of code combinations (object collection of the RobotCodeCmbActiveCaseInfo type) Available since version 1.7.
◆	AllBars ( <a href="#">see page 168</a> )	Available since version 1.7.
◆	AllNodes ( <a href="#">see page 168</a> )	Available since version 1.7.
◆	BarSel ( <a href="#">see page 168</a> )	Bar selection (taken into account only for simplified generation when AllBars ( <a href="#">see page 168</a> ) are set as zero - False) Available since version 1.7.
◆	ExtremalSnowFactor ( <a href="#">see page 169</a> )	Factor for extreme snow Available since version 1.7.
◆	GenType ( <a href="#">see page 169</a> )	Method of generation of code combinations.
◆	Groups ( <a href="#">see page 169</a> )	Groups Available since version 1.7.
◆	NodeSel ( <a href="#">see page 169</a> )	Node selection (taken into account only for simplified generation when AllNodes ( <a href="#">see page 168</a> ) are set as zero - False) Available since version 1.7.
◆	PointsOnBar ( <a href="#">see page 170</a> )	Number of points along the bar length (it is taken into account only for simplified generation) Available since version 1.7.
◆	Regulations ( <a href="#">see page 170</a> )	Definition of the regulations Available since version 1.7.
◆	Relations ( <a href="#">see page 170</a> )	Relations between groups Available since version 1.7.

### II.1.3.13.2.1 ActiveCases

#### C++

```
HRESULT get_ActiveCases(IRobotCollection**);
```

#### C#

```
public IRobotCollection ActiveCases { get; }
```

#### Visual Basic

```
Public ReadOnly ActiveCases As IRobotCollection
```

#### Description

Information about cases participating in a definition of code combinations (object collection of the RobotCodeCmbActiveCaseInfo type) Available since version 1.7.

### II.1.3.13.2.2 AllBars

#### C++

```
HRESULT get_AllBars(VARIANT_BOOL*);  
HRESULT put_AllBars(VARIANT_BOOL);
```

#### C#

```
public bool AllBars { get; set; }
```

#### Visual Basic

```
Public AllBars As Boolean
```

#### Description

Available since version 1.7.

### II.1.3.13.2.3 AllNodes

#### C++

```
HRESULT get_AllNodes(VARIANT_BOOL*);  
HRESULT put_AllNodes(VARIANT_BOOL);
```

#### C#

```
public bool AllNodes { get; set; }
```

#### Visual Basic

```
Public AllNodes As Boolean
```

#### Description

Available since version 1.7.

### II.1.3.13.2.4 BarSel

#### C++

```
HRESULT get_BarSel(BSTR*);  
HRESULT put_BarSel(BSTR);
```

#### C#

```
public String BarSel { get; set; }
```

#### Visual Basic

```
Public BarSel As String
```

**Description**

Bar selection (taken into account only for simplified generation when AllBars ( see page 168) are set as zero - False)  
Available since version 1.7.

**II.1.3.13.2.5 ExtremalSnowFactor****C++**

```
HRESULT get_ExtremalSnowFactor(double* );
HRESULT put_ExtremalSnowFactor(double);
```

**C#**

```
public double ExtremalSnowFactor { get; set; }
```

**Visual Basic**

```
Public ExtremalSnowFactor As Double
```

**Description**

Factor for extreme snow Available since version 1.7.

**II.1.3.13.2.6 GenType****C++**

```
HRESULT get_GenType(IRobotCodeCmbGenerationType* );
HRESULT put_GenType(IRobotCodeCmbGenerationType);
```

**C#**

```
public IRobotCodeCmbGenerationType GenType { get; set; }
```

**Visual Basic**

```
Public GenType As IRobotCodeCmbGenerationType
```

**Description**

Method of generation of code combinations.

**Version**

Available since version 8.2.

**II.1.3.13.2.7 Groups****C++**

```
HRESULT get_Groups(IRobotCodeCmbGroupServer** );
```

**C#**

```
public IRobotCodeCmbGroupServer Groups { get; }
```

**Visual Basic**

```
Public ReadOnly Groups As IRobotCodeCmbGroupServer
```

**Description**

Groups Available since version 1.7.

**II.1.3.13.2.8 NodeSel****C++**

```
HRESULT get_NodeSel(BSTR* );
HRESULT put_NodeSel(BSTR);
```

**C#**

```
public String NodeSel { get; set; }
```

**Visual Basic**

```
Public NodeSel As String
```

**Description**

Node selection (taken into account only for simplified generation when AllNodes (see page 168) are set as zero - False)  
Available since version 1.7.

**II.1.3.13.2.9 PointsOnBar****C++**

```
HRESULT get_PointsOnBar(short* );
HRESULT put_PointsOnBar(short);
```

**C#**

```
public short PointsOnBar { get; set; }
```

**Visual Basic**

```
Public PointsOnBar As short
```

**Description**

Number of points along the bar length (it is taken into account only for simplified generation) Available since version 1.7.

**II.1.3.13.2.10 Regulations****C++**

```
HRESULT get_Regulations(IRobotCodeCmbRegulations** );
```

**C#**

```
public IRobotCodeCmbRegulations Regulations { get; }
```

**Visual Basic**

```
Public ReadOnly Regulations As IRobotCodeCmbRegulations
```

**Description**

Definition of the regulations Available since version 1.7.

**II.1.3.13.2.11 Relations****C++**

```
HRESULT get_Relations(IRobotCodeCmbGroupRelationServer** );
```

**C#**

```
public IRobotCodeCmbGroupRelationServer Relations { get; }
```

**Visual Basic**

```
Public ReadOnly Relations As IRobotCodeCmbGroupRelationServer
```

**Description**

Relations between groups Available since version 1.7.

**II.1.3.13.3 IRobotCodeCmbGenerationParams Methods**

The methods of the IRobotCodeCmbGenerationParams class are listed here.

## Public Methods

	Name	Description
⌚	IsCombinationSelected (see page 171)	Function returns a value different from zero (True) if a specified template for generating combination is to be taken into account during generation. Available since version 1.7.
⌚	IsCombinationTypeSelected (see page 171)	Function returns True if templates describing combination of the given type are used during combination generation.
⌚	IsDecidingValueSelected (see page 171)	Function returns a value different from zero (True) if the specified decisive value has been selected. Available since version 1.7.
⌚	SelectCombination (see page 172)	Function enables selection of templates for calculating a combination among all the templates defined in the regulations. Available since version 1.7.
⌚	SelectCombinationType (see page 172)	Function allows for a selection of templates describing combination of the given type from among all templates defined in the regulation.
⌚	SelectDecidingValue (see page 172)	Function enables selection of decisive values. Available since version 1.7.

### II.1.3.13.3.1 IsCombinationSelected

#### C++

```
HRESULT IsCombinationSelected(long _cmb_idx, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsCombinationSelected(long _cmb_idx);
```

#### Visual Basic

```
Public Function IsCombinationSelected(_cmb_idx As long) As Boolean
```

#### Description

Function returns a value different from zero (True) if a specified template for generating combination is to be taken into account during generation. Available since version 1.7.

### II.1.3.13.3.2 IsCombinationTypeSelected

#### C++

```
HRESULT IsCombinationTypeSelected(IRobotCombinationType _cmb_type, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsCombinationTypeSelected(IRobotCombinationType _cmb_type);
```

#### Visual Basic

```
Public Function IsCombinationTypeSelected(_cmb_type As IRobotCombinationType) As Boolean
```

#### Description

Function returns True if templates describing combination of the given type are used during combination generation.

#### Version

Available since version 11.

### II.1.3.13.3.3 IsDecidingValueSelected

#### C++

```
HRESULT IsDecidingValueSelected(IRobotCodeCmbDecidingValueType _dvalue, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsDecidingValueSelected(IRobotCodeCmbDecidingValueType _dvalue);
```

**Visual Basic**

```
Public Function IsDecidingValueSelected(_dvalue As IRobotCodeCmbDecidingValueType) As Boolean
```

**Description**

Function returns a value different from zero (True) if the specified decisive value has been selected. Available since version 1.7.

**II.1.3.13.3.4 SelectCombination****C++**

```
HRESULT SelectCombination(long _cmb_idx, VARIANT_BOOL _select);
```

**C#**

```
public void SelectCombination(long _cmb_idx, bool _select);
```

**Visual Basic**

```
Public Sub SelectCombination(_cmb_idx As long, _select As Boolean)
```

**Description**

Function enables selection of templates for calculating a combination among all the templates defined in the regulations. Available since version 1.7.

**II.1.3.13.3.5 SelectCombinationType****C++**

```
HRESULT SelectCombinationType(IRobotCombinationType _cmb_type, VARIANT_BOOL _select);
```

**C#**

```
public void SelectCombinationType(IRobotCombinationType _cmb_type, bool _select);
```

**Visual Basic**

```
Public Sub SelectCombinationType(_cmb_type As IRobotCombinationType, _select As Boolean)
```

**Description**

Function allows for a selection of templates describing combination of the given type from among all templates defined in the regulation.

**Version**

Available since version 11.

**II.1.3.13.3.6 SelectDecidingValue****C++**

```
HRESULT SelectDecidingValue(IRobotCodeCmbDecidingValueType _dvalue, VARIANT_BOOL _select);
```

**C#**

```
public void SelectDecidingValue(IRobotCodeCmbDecidingValueType _dvalue, bool _select);
```

**Visual Basic**

```
Public Sub SelectDecidingValue(_dvalue As IRobotCodeCmbDecidingValueType, _select As
```

```
Boolean)
```

#### Description

Function enables selection of decisive values. Available since version 1.7.

### II.1.3.14 IRobotCodeCmbDecidingValueType

#### C++

```
enum IRobotCodeCmbDecidingValueType;
```

#### C#

```
public enum IRobotCodeCmbDecidingValueType;
```

#### Visual Basic

```
Public Enum IRobotCodeCmbDecidingValueType
```

#### Members

Members	Description
I_CCDVT_FX = 1	Available since version 1.7.
I_CCDVT_FY = 2	Available since version 1.7.
I_CCDVT_FZ = 3	Available since version 1.7.
I_CCDVT_MX = 4	Available since version 1.7.
I_CCDVT_MY = 5	Available since version 1.7.
I_CCDVT_MZ = 6	Available since version 1.7.
I_CCDVT_UX = 7	Available since version 1.7.
I_CCDVT_UY = 8	Available since version 1.7.
I_CCDVT_UZ = 9	Available since version 1.7.
I_CCDVT_FXX_MAX = 10	Available since version 1.7.
I_CCDVT_MXX_MAX = 11	Available since version 1.7.
I_CCDVT_SXX_MAX = 12	Available since version 1.7.
I_CCDVTREACTIONS = 35	Available since version 1.7.
I_CCDVT_SIGMA_X = 13	Available since version 1.7.
I_CCDVT_DEFLECTION = 14	Available since version 1.7.

### II.1.3.15 IRobotCodeCmbGroup

#### Class Hierarchy

#### C++

```
interface IRobotCodeCmbGroup : IDispatch;
```

#### C#

```
public interface IRobotCodeCmbGroup;
```

#### Visual Basic

```
Public Interface IRobotCodeCmbGroup
```

#### Description

Definition of load case groups.

### II.1.3.15.1 IRobotCodeCmbGroup Members

The following tables list the members exposed by IRobotCodeCmbGroup.

## Public Fields

	Name	Description
◆	CaseCount (see page 174)	Number of load cases included in the group Available since version 1.7.
◆	Nature (see page 174)	Nature of cases included in the group Available since version 1.7.
◆	Operator (see page 175)	Operator determining relation between the group components Available since version 1.7.

## Public Methods

	Name	Description
◆	Add (see page 175)	Function adds a case with the specified number to the group. If the case nature is incompatible with the group nature the function returns zero (False). Adding a case to this group means that it is simultaneously deleted from the group to which it has hitherto belonged. Available since version 1.7.
◆	AddAll (see page 176)	Functions adds to the group all the load cases whose nature is compatible with the group nature. Simultaneously, it means that these cases will be deleted from the groups to which they have hitherto belonged. Available since version 1.7.
◆	IsFull (see page 176)	Function returns a value different from zero (True) if the natures of all the load cases in a group are compatible with the group nature. Available since version 1.7.

### II.1.3.15.2 IRobotCodeCmbGroup Fields

The fields of the IRobotCodeCmbGroup class are listed here.

## Public Fields

	Name	Description
◆	CaseCount (see page 174)	Number of load cases included in the group Available since version 1.7.
◆	Nature (see page 174)	Nature of cases included in the group Available since version 1.7.
◆	Operator (see page 175)	Operator determining relation between the group components Available since version 1.7.

### II.1.3.15.2.1 CaseCount

#### C++

```
HRESULT get_CaseCount(long*);
```

#### C#

```
public long CaseCount { get; }
```

#### Visual Basic

```
Public ReadOnly CaseCount As long
```

#### Description

Number of load cases included in the group Available since version 1.7.

### II.1.3.15.2.2 Nature

#### C++

```
HRESULT get_Nature(IRobotCaseNature*);
```

#### C#

```
public IRobotCaseNature Nature { get; }
```

**Visual Basic**

```
Public ReadOnly Nature As IRobotCaseNature
```

**Description**

Nature of cases included in the group Available since version 1.7.

**II.1.3.15.2.3 Operator****C++**

```
HRESULT get_Operator(IRobotCodeCmbOperator* );
HRESULT put_Operator(IRobotCodeCmbOperator);
```

**C#**

```
public IRobotCodeCmbOperator Operator { get; set; }
```

**Visual Basic**

```
Public Operator As IRobotCodeCmbOperator
```

**Description**

Operator determining relation between the group components Available since version 1.7.

**II.1.3.15.3 IRobotCodeCmbGroup Methods**

The methods of the IRobotCodeCmbGroup class are listed here.

**Public Methods**

	Name	Description
+=	Add (see page 175)	Function adds a case with the specified number to the group. If the case nature is incompatible with the group nature the function returns zero (False). Adding a case to this group means that it is simultaneously deleted from the group to which it has hitherto belonged. Available since version 1.7.
+=	AddAll (see page 176)	Functions adds to the group all the load cases whose nature is compatible with the group nature. Simultaneously, it means that these cases will be deleted from the groups to which they have hitherto belonged. Available since version 1.7.
+=	IsFull (see page 176)	Function returns a value different from zero (True) if the natures of all the load cases in a group are compatible with the group nature. Available since version 1.7.

**II.1.3.15.3.1 Add****C++**

```
HRESULT Add(long _case_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool Add(long _case_num);
```

**Visual Basic**

```
Public Function Add(_case_num As long) As Boolean
```

**Description**

Function adds a case with the specified number to the group. If the case nature is incompatible with the group nature the function returns zero (False). Adding a case to this group means that it is simultaneously deleted from the group to which it has hitherto belonged. Available since version 1.7.

### II.1.3.15.3.2 AddAll

**C++**

```
HRESULT AddAll();
```

**C#**

```
public void AddAll();
```

**Visual Basic**

```
Public Sub AddAll()
```

**Description**

Functions adds to the group all the load cases whose nature is compatible with the group nature. Simultaneously, it means that these cases will be deleted from the groups to which they have hitherto belonged. Available since version 1.7.

### II.1.3.15.3.3 IsFull

**C++**

```
HRESULT IsFull(VARIANT_BOOL* ret);
```

**C#**

```
public bool IsFull();
```

**Visual Basic**

```
Public Function IsFull() As Boolean
```

**Description**

Function returns a value different from zero (True) if the natures of all the load cases in a group are compatible with the group nature. Available since version 1.7.

### II.1.3.16 IRobotCodeCmbOperator

**C++**

```
enum IRobotCodeCmbOperator;
```

**C#**

```
public enum IRobotCodeCmbOperator;
```

**Visual Basic**

```
Public Enum IRobotCodeCmbOperator
```

**Members**

Members	Description
I_CCO_AND = 0	Operator: and Available since version 1.7.
I_CCO_EXCLUSIVE_OR = 1	Operator: exclusive or Available since version 1.7.
I_CCO_AND_OR = 2	Operator: and/or Available since version 1.7.

**Description**

Available operators used for defining relations between cases and case groups. .

### II.1.3.17 IRobotCodeCmbGroupServer

**Class Hierarchy**

**C++**

```
interface IRobotCodeCmbGroupServer : IDispatch;
```

**C#**

```
public interface IRobotCodeCmbGroupServer;
```

**Visual Basic**

```
Public Interface IRobotCodeCmbGroupServer
```

**Description**

Server managing the groups of cases.

**II.1.3.17.1 IRobotCodeCmbGroupServer Members**

The following tables list the members exposed by IRobotCodeCmbGroupServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳️	FindByCase (🔗 see page 177)	Function finds the group which includes the specified load case and returns its number among the groups whose nature is compatible with the nature of that case. Available since version 1.7.
✳️	Get (🔗 see page 178)	Function returns a group with the specified number among the groups of the defined nature. Groups are numbered within the nature from 1 to their number. Available since version 1.7.
✳️	GetCount (🔗 see page 178)	Function returns the number of groups of the specified nature. Available since version 2.0.
✳️	New (🔗 see page 178)	Function creates a new group with the defined parameters and returns its number. Groups are numbered within a given nature from 1 to Count. Available since version 1.7.

**II.1.3.17.2 IRobotCodeCmbGroupServer Methods**

The methods of the IRobotCodeCmbGroupServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳️	FindByCase (🔗 see page 177)	Function finds the group which includes the specified load case and returns its number among the groups whose nature is compatible with the nature of that case. Available since version 1.7.
✳️	Get (🔗 see page 178)	Function returns a group with the specified number among the groups of the defined nature. Groups are numbered within the nature from 1 to their number. Available since version 1.7.
✳️	GetCount (🔗 see page 178)	Function returns the number of groups of the specified nature. Available since version 2.0.
✳️	New (🔗 see page 178)	Function creates a new group with the defined parameters and returns its number. Groups are numbered within a given nature from 1 to Count. Available since version 1.7.

**II.1.3.17.2.1 FindByCase****C++**

```
HRESULT FindByCase(long _case_num, long* ret);
```

**C#**

```
public long FindByCase(long _case_num);
```

**Visual Basic**

```
Public Function FindByCase(_case_num As long) As long
```

**Description**

Function finds the group which includes the specified load case and returns its number among the groups whose nature is compatible with the nature of that case. Available since version 1.7.

**II.1.3.17.2.2 Get****C++**

```
HRESULT Get(IRobotCaseNature _grp_nature, long _grp_number, IRobotCodeCmbGroup** ret);
```

**C#**

```
public IRobotCodeCmbGroup Get(IRobotCaseNature _grp_nature, long _grp_number);
```

**Visual Basic**

```
Public Function Get(_grp_nature As IRobotCaseNature, _grp_number As long) As IRobotCodeCmbGroup
```

**Description**

Function returns a group with the specified number among the groups of the defined nature. Groups are numbered within the nature from 1 to their number. Available since version 1.7.

**II.1.3.17.2.3 GetCount****C++**

```
HRESULT GetCount(IRobotCaseNature _grp_nature, long* ret);
```

**C#**

```
public long GetCount(IRobotCaseNature _grp_nature);
```

**Visual Basic**

```
Public Function GetCount(_grp_nature As IRobotCaseNature) As long
```

**Description**

Function returns the number of groups of the specified nature. Available since version 2.0.

**II.1.3.17.2.4 New****C++**

```
HRESULT New(IRobotCaseNature _grp_nature, IRobotCodeCmbOperator _oper, BSTR _case_list, long* ret);
```

**C#**

```
public long New(IRobotCaseNature _grp_nature, IRobotCodeCmbOperator _oper, String _case_list);
```

**Visual Basic**

```
Public Function New(_grp_nature As IRobotCaseNature, _oper As IRobotCodeCmbOperator, _case_list As String) As long
```

**Description**

Function creates a new group with the defined parameters and returns its number. Groups are numbered within a given nature from 1 to Count. Available since version 1.7.

## II.1.3.18 IRobotCodeCmbGroupRelation

### Class Hierarchy

#### C++

```
interface IRobotCodeCmbGroupRelation : IDispatch;
```

#### C#

```
public interface IRobotCodeCmbGroupRelation;
```

### Visual Basic

```
Public Interface IRobotCodeCmbGroupRelation
```

### Description

Definition of relations between groups. Definition of relations between groups can be illustrated by means of the matrix whose elements are group numbers. All groups whose numbers are located in the same matrix row are in the same relation with each other (either AND or AND\_OR). Whereas among the matrix rows the relation EXCLUSIVE\_OR occurs. .

## II.1.3.18.1 IRobotCodeCmbGroupRelation Members

The following tables list the members exposed by IRobotCodeCmbGroupRelation.

### Public Fields

	Name	Description
◆	Nature (see page 180)	Nature of groups defining relation Available since version 1.7.
◆	RowCount (see page 180)	Number of rows defining relation Available since version 1.7.

### Public Methods

	Name	Description
◆	AddGroup (see page 181)	Function adds a specified number group to the indicated row (at the end of the row). Only the groups whose nature is compatible with the relation nature can be added to the relation. Available since version 1.7.
◆	Clear (see page 181)	Function deletes all the rows of the relation definition matrix. Available since version 1.7.
◆	GetGroup (see page 181)	Function returns the group number entered into the matrix into a specified position of the selected row. Available since version 1.7.
◆	GetGroupCount (see page 182)	Function returns the number of groups defining a selected matrix row. Matrix rows are numbered from 1 to RowCount (see page 180). Available since version 1.7.
◆	GetOperator (see page 182)	Function returns the operator for the indicated row of relation definition matrix. Available since version 1.7.
◆	NewRow (see page 182)	Function creates a new row of the relation definition matrix and returns its index. Available since version 1.7.
◆	RemoveRow (see page 182)	Function deletes a row with the indicated index from the relation definition matrix. The numbers of the remaining rows are updated automatically so that they constitute a sequence of numbers from 1 to RowCount (see page 180). Available since version 1.7.
◆	SetGroup (see page 183)	Function enters a specified number into the indicated position of the selected row. Only the groups whose nature is compatible with the relation nature can be saved in the relation. Available since version 1.7.
◆	SetOperator (see page 183)	Function sets the operator for the selected matrix row. Available since version 1.7.

## II.1.3.18.2 IRobotCodeCmbGroupRelation Fields

The fields of the IRobotCodeCmbGroupRelation class are listed here.

## Public Fields

	Name	Description
◆	Nature (see page 180)	Nature of groups defining relation Available since version 1.7.
◆	RowCount (see page 180)	Number of rows defining relation Available since version 1.7.

### II.1.3.18.2.1 Nature

#### C++

```
HRESULT get_Nature(IRobotCaseNature*);
```

#### C#

```
public IRobotCaseNature Nature { get; }
```

#### Visual Basic

```
Public ReadOnly Nature As IRobotCaseNature
```

#### Description

Nature of groups defining relation Available since version 1.7.

### II.1.3.18.2.2 RowCount

#### C++

```
HRESULT get_RowCount(long*);
```

#### C#

```
public long RowCount { get; }
```

#### Visual Basic

```
Public ReadOnly RowCount As long
```

#### Description

Number of rows defining relation Available since version 1.7.

### II.1.3.18.3 IRobotCodeCmbGroupRelation Methods

The methods of the IRobotCodeCmbGroupRelation class are listed here.

#### Public Methods

	Name	Description
◆	AddGroup (see page 181)	Function adds a specified number group to the indicated row (at the end of the row). Only the groups whose nature is compatible with the relation nature can be added to the relation. Available since version 1.7.
◆	Clear (see page 181)	Function deletes all the rows of the relation definition matrix. Available since version 1.7.
◆	GetGroup (see page 181)	Function returns the group number entered into the matrix into a specified position of the selected row. Available since version 1.7.
◆	GetGroupCount (see page 182)	Function returns the number of groups defining a selected matrix row. Matrix rows are numbered from 1 to RowCount (see page 180). Available since version 1.7.
◆	GetOperator (see page 182)	Function returns the operator for the indicated row of relation definition matrix. Available since version 1.7.
◆	NewRow (see page 182)	Function creates a new row of the relation definition matrix and returns its index. Available since version 1.7.

	RemoveRow ( <a href="#">see page 182</a> )	Function deletes a row with the indicated index from the relation definition matrix. The numbers of the remaining rows are updated automatically so that they constitute a sequence of numbers from 1 to RowCount ( <a href="#">see page 180</a> ). Available since version 1.7.
	SetGroup ( <a href="#">see page 183</a> )	Function enters a specified number into the indicated position of the selected row. Only the groups whose nature is compatible with the relation nature can be saved in the relation. Available since version 1.7.
	SetOperator ( <a href="#">see page 183</a> )	Function sets the operator for the selected matrix row. Available since version 1.7.

### II.1.3.18.3.1 AddGroup

#### C++

```
HRESULT AddGroup(long _row_idx, long _grp_number);
```

#### C#

```
public void AddGroup(long _row_idx, long _grp_number);
```

#### Visual Basic

```
Public Sub AddGroup(_row_idx As long, _grp_number As long)
```

#### Description

Function adds a specified number group to the indicated row (at the end of the row). Only the groups whose nature is compatible with the relation nature can be added to the relation. Available since version 1.7.

### II.1.3.18.3.2 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

Function deletes all the rows of the relation definition matrix. Available since version 1.7.

### II.1.3.18.3.3 GetGroup

#### C++

```
HRESULT GetGroup(long _row_idx, long _grp_idx, long* ret);
```

#### C#

```
public long GetGroup(long _row_idx, long _grp_idx);
```

#### Visual Basic

```
Public Function GetGroup(_row_idx As long, _grp_idx As long) As long
```

#### Description

Function returns the group number entered into the matrix into a specified position of the selected row. Available since version 1.7.

#### II.1.3.18.3.4 GetGroupCount

**C++**

```
HRESULT GetGroupCount(long _row_idx, long* ret);
```

**C#**

```
public long GetGroupCount(long _row_idx);
```

**Visual Basic**

```
Public Function GetGroupCount(_row_idx As long) As long
```

**Description**

Function returns the number of groups defining a selected matrix row. Matrix rows are numbered from 1 to RowCount (see page 180). Available since version 1.7.

#### II.1.3.18.3.5 GetOperator

**C++**

```
HRESULT GetOperator(long _row_idx, IRobotCodeCmbOperator* ret);
```

**C#**

```
public IRobotCodeCmbOperator GetOperator(long _row_idx);
```

**Visual Basic**

```
Public Function GetOperator(_row_idx As long) As IRobotCodeCmbOperator
```

**Description**

Function returns the operator for the indicated row of relation definition matrix. Available since version 1.7.

#### II.1.3.18.3.6 NewRow

**C++**

```
HRESULT NewRow(IRobotCodeCmbOperator _oper_id, long* ret);
```

**C#**

```
public long NewRow(IRobotCodeCmbOperator _oper_id);
```

**Visual Basic**

```
Public Function NewRow(_oper_id As IRobotCodeCmbOperator) As long
```

**Description**

Function creates a new row of the relation definition matrix and returns its index. Available since version 1.7.

#### II.1.3.18.3.7 RemoveRow

**C++**

```
HRESULT RemoveRow(long _row_idx);
```

**C#**

```
public void RemoveRow(long _row_idx);
```

**Visual Basic**

```
Public Sub RemoveRow(_row_idx As long)
```

## Description

Function deletes a row with the indicated index from the relation definition matrix. The numbers of the remaining rows are updated automatically so that they constitute a sequence of numbers from 1 to RowCount (see page 180). Available since version 1.7.

### II.1.3.18.3.8 SetGroup

#### C++

```
HRESULT SetGroup(long _row_idx, long _grp_idx, long _grp_number);
```

#### C#

```
public void SetGroup(long _row_idx, long _grp_idx, long _grp_number);
```

#### Visual Basic

```
Public Sub SetGroup(_row_idx As long, _grp_idx As long, _grp_number As long)
```

## Description

Function enters a specified number into the indicated position of the selected row. Only the groups whose nature is compatible with the relation nature can be saved in the relation. Available since version 1.7.

### II.1.3.18.3.9 SetOperator

#### C++

```
HRESULT SetOperator(long _row_idx, IRobotCodeCmbOperator _oper_id);
```

#### C#

```
public void SetOperator(long _row_idx, IRobotCodeCmbOperator _oper_id);
```

#### Visual Basic

```
Public Sub SetOperator(_row_idx As long, _oper_id As IRobotCodeCmbOperator)
```

## Description

Function sets the operator for the selected matrix row. Available since version 1.7.

### II.1.3.19 IRobotCodeCmbGroupRelationServer

#### Class Hierarchy

#### C++

```
interface IRobotCodeCmbGroupRelationServer : IDispatch;
```

#### C#

```
public interface IRobotCodeCmbGroupRelationServer;
```

#### Visual Basic

```
Public Interface IRobotCodeCmbGroupRelationServer
```

## Description

Server managing relations among groups..

### II.1.3.19.1 IRobotCodeCmbGroupRelationServer Members

The following tables list the members exposed by IRobotCodeCmbGroupRelationServer.

## Public Methods

	Name	Description
💡	Get (see page 184)	Function returns the definition of the relation with the specified nature. Available since version 1.7.
💡	Set (see page 184)	Function saves a specified group relation definition. The previous group relation definition for the same nature will be replaced. Available since version 1.7.

### II.1.3.19.2 IRobotCodeCmbGroupRelationServer Methods

The methods of the IRobotCodeCmbGroupRelationServer class are listed here.

## Public Methods

	Name	Description
💡	Get (see page 184)	Function returns the definition of the relation with the specified nature. Available since version 1.7.
💡	Set (see page 184)	Function saves a specified group relation definition. The previous group relation definition for the same nature will be replaced. Available since version 1.7.

#### II.1.3.19.2.1 Get

##### C++

```
HRESULT Get(IRobotCaseNature _rel_nature, IRobotCodeCmbGroupRelation** ret);
```

##### C#

```
public IRobotCodeCmbGroupRelation Get(IRobotCaseNature _rel_nature);
```

##### Visual Basic

```
Public Function Get(_rel_nature As IRobotCaseNature) As IRobotCodeCmbGroupRelation
```

##### Description

Function returns the definition of the relation with the specified nature. Available since version 1.7.

#### II.1.3.19.2.2 Set

##### C++

```
HRESULT Set(IRobotCodeCmbGroupRelation* _rel_def);
```

##### C#

```
public void Set(IRobotCodeCmbGroupRelation _rel_def);
```

##### Visual Basic

```
Public Sub Set(ByRef _rel_def As IRobotCodeCmbGroupRelation)
```

##### Description

Function saves a specified group relation definition. The previous group relation definition for the same nature will be replaced. Available since version 1.7.

### II.1.3.20 IRobotCodeCmbActiveCaseInfo

#### Class Hierarchy

##### C++

```
interface IRobotCodeCmbActiveCaseInfo : IDispatch;
```

**C#**

```
public interface IRobotCodeCmbActiveCaseInfo;
```

**Visual Basic**

```
Public Interface IRobotCodeCmbActiveCaseInfo
```

**II.1.3.20.1 IRobotCodeCmbActiveCaseInfo Members**

The following tables list the members exposed by IRobotCodeCmbActiveCaseInfo.

**Public Fields**

	Name	Description
❖	CaseNature (see page 185)	Nature of the load case Available since version 1.7.
❖	CaseNumber (see page 185)	Number of the load case Available since version 1.7.
❖	Coefficient (see page 186)	Factor Available since version 1.7.
❖	GroupNumber (see page 186)	Number of the group to which the case belongs Available since version 1.7.
❖	IsSelected (see page 186)	Flag indicating if the case will be taken into account while generating code combinations Available since version 1.7.

**II.1.3.20.2 IRobotCodeCmbActiveCaseInfo Fields**

The fields of the IRobotCodeCmbActiveCaseInfo class are listed here.

**Public Fields**

	Name	Description
❖	CaseNature (see page 185)	Nature of the load case Available since version 1.7.
❖	CaseNumber (see page 185)	Number of the load case Available since version 1.7.
❖	Coefficient (see page 186)	Factor Available since version 1.7.
❖	GroupNumber (see page 186)	Number of the group to which the case belongs Available since version 1.7.
❖	IsSelected (see page 186)	Flag indicating if the case will be taken into account while generating code combinations Available since version 1.7.

**II.1.3.20.2.1 CaseNature****C++**

```
HRESULT get_CaseNature(IRobotCaseNature*);
```

**C#**

```
public IRobotCaseNature CaseNature { get; }
```

**Visual Basic**

```
Public ReadOnly CaseNature As IRobotCaseNature
```

**Description**

Nature of the load case Available since version 1.7.

**II.1.3.20.2.2 CaseNumber****C++**

```
HRESULT get_CaseNumber(long*);
```

**C#**

```
public long CaseNumber { get; }
```

**Visual Basic**

```
Public ReadOnly CaseNumber As long
```

**Description**

Number of the load case Available since version 1.7.

**II.1.3.20.2.3 Coefficient****C++**

```
HRESULT get_Coefficient(double* );
HRESULT put_Coefficient(double);
```

**C#**

```
public double Coefficient { get; set; }
```

**Visual Basic**

```
Public Coefficient As double
```

**Description**

Factor Available since version 1.7.

**II.1.3.20.2.4 GroupNumber****C++**

```
HRESULT get_GroupNumber(long* );
```

**C#**

```
public long GroupNumber { get; }
```

**Visual Basic**

```
Public ReadOnly GroupNumber As long
```

**Description**

Number of the group to which the case belongs Available since version 1.7.

**II.1.3.20.2.5 IsSelected****C++**

```
HRESULT get_IsSelected(VARIANT_BOOL* );
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Description**

Flag indicating if the case will be taken into account while generating code combinations Available since version 1.7.

**II.1.3.21 IRobotCodeCmbGenerationType****C++**

```
enum IRobotCodeCmbGenerationType;
```

**C#**

```
public enum IRobotCodeCmbGenerationType;
```

## Visual Basic

```
Public Enum IRobotCodeCmbGenerationType
```

### Members

Members	Description
I_CCGT_NONE = 0	No generation of code combinations. Available since version 8.2.
I_CCGT_SIMPLIFIED = 1	Generation of simplified code combinations. Available since version 8.2.
I_CCGT_FULL = 2	Generation of full code combinations. Available since version 8.2.
I_CCGT_MANUAL = 3	Generation of manual combinations. Available since version 11.

### Version

Available since version 8.2.

## II.1.4 Modal analysis parameters

### Enumerations

	Name	Description
	IRobotModalAnalysisMode (see page 190)	Available types of modal analysis.
	IRobotModalAnalysisAlgorithm (see page 191)	Available methods of eigenproblem solution.
	IRobotModalAnalysisMassMatrixType (see page 191)	Available mass matrix types.
	IRobotModalAnalysisLimitType (see page 192)	Available quantities by means of which a limit value may be defined.

### Interfaces

	Name	Description
	IRobotModalAnalysisParams (see page 192)	Parameters of modal analysis.
	IRobotModalAnalysisBase (see page 199)	Reduced base definition.
	IRobotModalAnalysisLimits (see page 201)	Definition of a limit range.
	IRobotModalAnalysisShifts (see page 203)	Shift definition.
	IRobotMassEccentricities (see page 204)	Definition of eccentricities.

### II.1.4.1 Modal analysis parameters recognizing static forces

Available since version 2.5.

### Interfaces

	Name	Description
	IRobotModalWithStaticForcesAnalysisParams (see page 187)	Modal analysis parameters with static forces taken into account.

### II.1.4.1.1 IRobotModalWithStaticForcesAnalysisParams

#### Class Hierarchy

**C++**

```
interface IRobotModalWithStaticForcesAnalysisParams : IRobotModalAnalysisParams;
```

**C#**

```
public interface IRobotModalWithStaticForcesAnalysisParams : IRobotModalAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotModalWithStaticForcesAnalysisParams
```

**Description**

Modal analysis parameters with static forces taken into account.

**Version**

Available since version 2.5.

**II.1.4.1.1.1 IRobotModalWithStaticForcesAnalysisParams Members**

The following tables list the members exposed by IRobotModalWithStaticForcesAnalysisParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Acceleration ( <a href="#">see page 194</a> )	Gravity value.
◆	Base ( <a href="#">see page 194</a> )	Reduced base definition.
◆	Damping ( <a href="#">see page 194</a> )	Damping value.
◆	DisregardDensity ( <a href="#">see page 194</a> )	Flag indicating if density of the structure element is to be taken into account during analysis .
◆	IncludeDampingInCalculations ( <a href="#">see page 195</a> )	Flag indicating if damping is to be taken into account in calculations.
◆	IterationsCount ( <a href="#">see page 195</a> )	Number of iterations.
◆	Limits ( <a href="#">see page 195</a> )	Limitation of the calculated mode number.
◆	MassEccentricities ( <a href="#">see page 196</a> )	Definition of eccentricities.
◆	MassMatrix ( <a href="#">see page 196</a> )	Mass matrix.
◆	MassParticipation ( <a href="#">see page 196</a> )	Definition of the value of mass participation percentage.
◆	Method ( <a href="#">see page 197</a> )	Method of eigenproblem solution
◆	Mode ( <a href="#">see page 197</a> )	Analysis mode.
◆	ModesCount ( <a href="#">see page 197</a> )	Number of modes.
◆	Nonlinear ( <a href="#">see page 189</a> )	Non-linear analysis.
◆	Shifts ( <a href="#">see page 198</a> )	Shift definition.
◆	SturmVerification ( <a href="#">see page 198</a> )	Switching on or off Sturm Check .
◆	Tolerance ( <a href="#">see page 198</a> )	Calculation tolerance value.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	CreateFromStatic ( <a href="#">see page 189</a> )	Function initializes parameters of modal analysis with static forces taken into account based on the specified static load case. .
◆	GetStaticStateParams ( <a href="#">see page 190</a> )	Function returns parameters of non-linear analysis.
◆	SetStaticStateParams ( <a href="#">see page 190</a> )	Function sets parameters of non-linear analysis.

**II.1.4.1.1.2 IRobotModalWithStaticForcesAnalysisParams Fields**

The fields of the IRobotModalWithStaticForcesAnalysisParams class are listed here.

## Public Fields

	Name	Description
◆	Nonlinear (see page 189)	Non-linear analysis.

### II.1.4.1.1.2.1 Nonlinear

#### C++

```
HRESULT get_Nonlinear(VARIANT_BOOL* );
HRESULT put_Nonlinear(VARIANT_BOOL);
```

#### C#

```
public bool Nonlinear { get; set; }
```

#### Visual Basic

```
Public Nonlinear As Boolean
```

#### Description

Non-linear analysis.

#### Version

Available since version 2.5.

### II.1.4.1.1.3 IRobotModalWithStaticForcesAnalysisParams Methods

The methods of the IRobotModalWithStaticForcesAnalysisParams class are listed here.

#### Public Methods

	Name	Description
◆	CreateFromStatic (see page 189)	Function initializes parameters of modal analysis with static forces taken into account based on the specified static load case. .
◆	GetStaticStateParams (see page 190)	Function returns parameters of non-linear analysis.
◆	SetStaticStateParams (see page 190)	Function sets parameters of non-linear analysis.

### II.1.4.1.1.3.1 CreateFromStatic

#### C++

```
HRESULT CreateFromStatic(long _static_case_num, VARIANT_BOOL* ret);
```

#### C#

```
public bool CreateFromStatic(long _static_case_num);
```

#### Visual Basic

```
Public Function CreateFromStatic(_static_case_num As long) As Boolean
```

#### Description

Function initializes parameters of modal analysis with static forces taken into account based on the specified static load case. .

#### Version

Available since version 2.5.

### II.1.4.1.1.3.2 GetStaticStateParams

#### C++

```
HRESULT GetStaticStateParams(IRobotNonlinearAnalysisParams** ret);
```

#### C#

```
public IRobotNonlinearAnalysisParams GetStaticStateParams();
```

#### Visual Basic

```
Public Function GetStaticStateParams() As IRobotNonlinearAnalysisParams
```

#### Description

Function returns parameters of non-linear analysis.

#### Version

Available since version 2.5.

### II.1.4.1.1.3.3 SetStaticStateParams

#### C++

```
HRESULT SetStaticStateParams(IRobotNonlinearAnalysisParams* _params);
```

#### C#

```
public void SetStaticStateParams(IRobotNonlinearAnalysisParams _params);
```

#### Visual Basic

```
Public Sub SetStaticStateParams(ByRef _params As IRobotNonlinearAnalysisParams)
```

#### Description

Function sets parameters of non-linear analysis.

#### Version

Available since version 2.5.

### II.1.4.2 IRobotModalAnalysisMode

#### C++

```
enum IRobotModalAnalysisMode;
```

#### C#

```
public enum IRobotModalAnalysisMode;
```

#### Visual Basic

```
Public Enum IRobotModalAnalysisMode
```

#### Members

Members	Description
I_MAM_MODAL = 0	Modal analysis. Available since version 2.5.
I_MAM_SEISMIC = 1	Seismic analysis. Available since version 2.5.
I_MAM_SEISMIC_PSEUDO = 2	Pseudomodal seismic analysis. Available since version 2.5.

**Description**

Available types of modal analysis.

**Version**

Available since version 2.5.

**II.1.4.3 IRobotModalAnalysisAlgorithm****C++**

```
enum IRobotModalAnalysisAlgorithm;
```

**C#**

```
public enum IRobotModalAnalysisAlgorithm;
```

**Visual Basic**

```
Public Enum IRobotModalAnalysisAlgorithm
```

**Members**

Members	Description
I_MAA_BLOCK_SUBSPACE_ITERATION = 0	Block subspace iteration. Available since version 2.5.
I_MAA_SUBSPACE_ITERATION = 1	Subspace iteration. Available since version 2.5.
I_MAA_LANZOS = 2	Lanczos method. Available since version 2.5.
I_MAA_BASE_REDUCTION = 3	Base reduction method. Available since version 2.5.
I_MAA_PCG_RITZ = 4	Ritz method. Available since version 2.5.
I_MAA_PCG = 5	Preconditioned conjugate gradient method . Available since version 2.5.

**Description**

Available methods of eigenproblem solution.

**Version**

Available since version 2.5.

**II.1.4.4 IRobotModalAnalysisMassMatrixType****C++**

```
enum IRobotModalAnalysisMassMatrixType;
```

**C#**

```
public enum IRobotModalAnalysisMassMatrixType;
```

**Visual Basic**

```
Public Enum IRobotModalAnalysisMassMatrixType
```

**Members**

Members	Description
I_MAMMT_CONSISTENT = 0	Consistent mass matrix. Available since version 2.5.

I_MAMMT_LUMPED_WITH_ROTATIONS = 1	Lumped mass matrix with rotations. Available since version 2.5.
I_MAMMT_LUMPED = 2	Lumped mass matrix without rotations. Available since version 2.5.

**Description**

Available mass matrix types.

**Version**

Available since version 2.5.

**II.1.4.5 IRobotModalAnalysisLimitType****C++**

```
enum IRobotModalAnalysisLimitType;
```

**C#**

```
public enum IRobotModalAnalysisLimitType;
```

**Visual Basic**

```
Public Enum IRobotModalAnalysisLimitType
```

**Members**

Members	Description
I_MALT_PULSATION = 0	Pulsation. Available since version 2.5.
I_MALT_FREQUENCY = 1	Frequency. Available since version 2.5.
I_MALT_PERIOD = 2	Period. Available since version 2.5.

**Description**

Available quantities by means of which a limit value may be defined.

**Version**

Available since version 2.5.

**II.1.4.6 IRobotModalAnalysisParams****Class Hierarchy****C++**

```
interface IRobotModalAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotModalAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotModalAnalysisParams
```

**Description**

Parameters of modal analysis.

**Version**

Available since version 2.5.

### II.1.4.6.1 IRobotModalAnalysisParams Members

The following tables list the members exposed by IRobotModalAnalysisParams.

#### Public Fields

	<b>Name</b>	<b>Description</b>
◆	Acceleration ( <a href="#">see page 194</a> )	Gravity value.
◆	Base ( <a href="#">see page 194</a> )	Reduced base definition.
◆	Damping ( <a href="#">see page 194</a> )	Damping value.
◆	DisregardDensity ( <a href="#">see page 194</a> )	Flag indicating if density of the structure element is to be taken into account during analysis .
◆	IncludeDampingInCalculations ( <a href="#">see page 195</a> )	Flag indicating if damping is to be taken into account in calculations.
◆	IterationsCount ( <a href="#">see page 195</a> )	Number of iterations.
◆	Limits ( <a href="#">see page 195</a> )	Limitation of the calculated mode number.
◆	MassEccentricities ( <a href="#">see page 196</a> )	Definition of eccentricities.
◆	MassMatrix ( <a href="#">see page 196</a> )	Mass matrix.
◆	MassParticipation ( <a href="#">see page 196</a> )	Definition of the value of mass participation percentage.
◆	Method ( <a href="#">see page 197</a> )	Method of eigenproblem solution
◆	Mode ( <a href="#">see page 197</a> )	Analysis mode.
◆	ModesCount ( <a href="#">see page 197</a> )	Number of modes.
◆	Shifts ( <a href="#">see page 198</a> )	Shift definition.
◆	SturmVerification ( <a href="#">see page 198</a> )	Switching on or off Sturm Check .
◆	Tolerance ( <a href="#">see page 198</a> )	Calculation tolerance value.

### II.1.4.6.2 IRobotModalAnalysisParams Fields

The fields of the IRobotModalAnalysisParams class are listed here.

#### Public Fields

	<b>Name</b>	<b>Description</b>
◆	Acceleration ( <a href="#">see page 194</a> )	Gravity value.
◆	Base ( <a href="#">see page 194</a> )	Reduced base definition.
◆	Damping ( <a href="#">see page 194</a> )	Damping value.
◆	DisregardDensity ( <a href="#">see page 194</a> )	Flag indicating if density of the structure element is to be taken into account during analysis .
◆	IncludeDampingInCalculations ( <a href="#">see page 195</a> )	Flag indicating if damping is to be taken into account in calculations.
◆	IterationsCount ( <a href="#">see page 195</a> )	Number of iterations.
◆	Limits ( <a href="#">see page 195</a> )	Limitation of the calculated mode number.
◆	MassEccentricities ( <a href="#">see page 196</a> )	Definition of eccentricities.
◆	MassMatrix ( <a href="#">see page 196</a> )	Mass matrix.
◆	MassParticipation ( <a href="#">see page 196</a> )	Definition of the value of mass participation percentage.
◆	Method ( <a href="#">see page 197</a> )	Method of eigenproblem solution
◆	Mode ( <a href="#">see page 197</a> )	Analysis mode.
◆	ModesCount ( <a href="#">see page 197</a> )	Number of modes.
◆	Shifts ( <a href="#">see page 198</a> )	Shift definition.
◆	SturmVerification ( <a href="#">see page 198</a> )	Switching on or off Sturm Check .
◆	Tolerance ( <a href="#">see page 198</a> )	Calculation tolerance value.

#### II.1.4.6.2.1 Acceleration

##### C++

```
HRESULT get_Acceleration(double*);  
HRESULT put_Acceleration(double);
```

##### C#

```
public double Acceleration { get; set; }
```

##### Visual Basic

```
Public Acceleration As Double
```

##### Description

Gravity value.

##### Version

Available since version 2.5.

#### II.1.4.6.2.2 Base

##### C++

```
HRESULT get_Base(IRobotModalAnalysisBase**);
```

##### C#

```
public IRobotModalAnalysisBase Base { get; }
```

##### Visual Basic

```
Public ReadOnly Base As IRobotModalAnalysisBase
```

##### Description

Reduced base definition.

##### Version

Available since version 2.5.

#### II.1.4.6.2.3 Damping

##### C++

```
HRESULT get_Damping(double*);  
HRESULT put_Damping(double);
```

##### C#

```
public double Damping { get; set; }
```

##### Visual Basic

```
Public Damping As Double
```

##### Description

Damping value.

##### Version

Available since version 2.5.

#### II.1.4.6.2.4 DisregardDensity

##### C++

```
HRESULT get_DisregardDensity(VARIANT_BOOL*);
```

```
HRESULT put_DisregardDensity(VARIANT_BOOL);
```

**C#**

```
public bool DisregardDensity { get; set; }
```

**Visual Basic**

```
Public DisregardDensity As Boolean
```

**Description**

Flag indicating if density of the structure element is to be taken into account during analysis .

**Version**

Available since version 2.5.

**II.1.4.6.2.5 IncludeDampingInCalculations****C++**

```
HRESULT get_IncludeDampingInCalculations(VARIANT_BOOL*);  
HRESULT put_IncludeDampingInCalculations(VARIANT_BOOL);
```

**C#**

```
public bool IncludeDampingInCalculations { get; set; }
```

**Visual Basic**

```
Public IncludeDampingInCalculations As Boolean
```

**Description**

Flag indicating if damping is to be taken into account in calculations.

**Version**

Available since version 2.5.

**II.1.4.6.2.6 IterationsCount****C++**

```
HRESULT get_IterationsCount(long*);  
HRESULT put_IterationsCount(long);
```

**C#**

```
public long IterationsCount { get; set; }
```

**Visual Basic**

```
Public IterationsCount As long
```

**Description**

Number of iterations.

**Version**

Available since version 2.5.

**II.1.4.6.2.7 Limits****C++**

```
HRESULT get_Limits(IRobotModalAnalysisLimits**);
```

**C#**

```
public IRobotModalAnalysisLimits Limits { get; }
```

**Visual Basic**

```
Public ReadOnly Limits As IRobotModalAnalysisLimits
```

**Description**

Limitation of the calculated mode number.

**Version**

Available since version 2.5.

## II.1.4.6.2.8 MassEccentricities

**C++**

```
HRESULT get_MassEccentricities(IRobotMassEccentricities**);
```

**C#**

```
public IRobotMassEccentricities MassEccentricities { get; }
```

**Visual Basic**

```
Public ReadOnly MassEccentricities As IRobotMassEccentricities
```

**Description**

Definition of eccentricities.

**Version**

Available since version 6.21.

## II.1.4.6.2.9 MassMatrix

**C++**

```
HRESULT get_MassMatrix(IRobotModalAnalysisMassMatrixType*);  
HRESULT put_MassMatrix(IRobotModalAnalysisMassMatrixType);
```

**C#**

```
public IRobotModalAnalysisMassMatrixType MassMatrix { get; set; }
```

**Visual Basic**

```
Public MassMatrix As IRobotModalAnalysisMassMatrixType
```

**Description**

Mass matrix.

**Version**

Available since version 2.5.

## II.1.4.6.2.10 MassParticipation

**C++**

```
HRESULT get_MassParticipation(double*);  
HRESULT put_MassParticipation(double);
```

**C#**

```
public double MassParticipation { get; set; }
```

**Visual Basic**

```
Public MassParticipation As Double
```

**Description**

Definition of the value of mass participation percentage.

**Version**

Available since version 2.5.

**II.1.4.6.2.11 Method****C++**

```
HRESULT get_Method(IRobotModalAnalysisAlgorithm* );
HRESULT put_Method(IRobotModalAnalysisAlgorithm);
```

**C#**

```
public IRobotModalAnalysisAlgorithm Method { get; set; }
```

**Visual Basic**

```
Public Method As IRobotModalAnalysisAlgorithm
```

**Description**

Method of eigenproblem solution

**Notes**

The selected algorithm of eigenproblem solution must be compatible with the active solver..

**Version**

Available since version 2.5.

**II.1.4.6.2.12 Mode****C++**

```
HRESULT get_Mode(IRobotModalAnalysisMode* );
HRESULT put_Mode(IRobotModalAnalysisMode);
```

**C#**

```
public IRobotModalAnalysisMode Mode { get; set; }
```

**Visual Basic**

```
Public Mode As IRobotModalAnalysisMode
```

**Description**

Analysis mode.

**Version**

Available since version 2.5.

**II.1.4.6.2.13 ModesCount****C++**

```
HRESULT get_ModesCount(long* );
HRESULT put_ModesCount(long);
```

**C#**

```
public long ModesCount { get; set; }
```

**Visual Basic**

```
Public ModesCount As long
```

**Description**

Number of modes.

**Version**

Available since version 2.5.

**II.1.4.6.2.14 Shifts****C++**

```
HRESULT get_Shifts(IRobotModalAnalysisShifts**);
```

**C#**

```
public IRobotModalAnalysisShifts Shifts { get; }
```

**Visual Basic**

```
Public ReadOnly Shifts As IRobotModalAnalysisShifts
```

**Description**

Shift definition.

**Version**

Available since version 2.5.

**II.1.4.6.2.15 SturmVerification****C++**

```
HRESULT get_SturmVerification(VARIANT_BOOL*);  
HRESULT put_SturmVerification(VARIANT_BOOL);
```

**C#**

```
public bool SturmVerification { get; set; }
```

**Visual Basic**

```
Public SturmVerification As Boolean
```

**Description**

Switching on or off Sturm Check .

**Version**

Available since version 2.5.

**II.1.4.6.2.16 Tolerance****C++**

```
HRESULT get_Tolerance(double*);  
HRESULT put_Tolerance(double);
```

**C#**

```
public double Tolerance { get; set; }
```

**Visual Basic**

```
Public Tolerance As Double
```

**Description**

Calculation tolerance value.

**Version**

Available since version 2.5.

**II.1.4.7 IRobotModalAnalysisBase****Class Hierarchy****C++**

```
interface IRobotModalAnalysisBase : IDispatch;
```

**C#**

```
public interface IRobotModalAnalysisBase;
```

**Visual Basic**

```
Public Interface IRobotModalAnalysisBase
```

**Description**

Reduced base definition.

**Version**

Available since version 2.5.

**II.1.4.7.1 IRobotModalAnalysisBase Members**

The following tables list the members exposed by IRobotModalAnalysisBase.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Add ( <a href="#">see page 199</a> )	Function generates a new element defining the base and returns its index. .
≡	Count ( <a href="#">see page 200</a> )	Function returns the number of elements defining the base. Elements defining the base are indexed from 1 to Count.
≡	Delete ( <a href="#">see page 200</a> )	Function deletes the indicated element defining the base.
≡	Get ( <a href="#">see page 200</a> )	Function returns the selected element defining the base. .

**II.1.4.7.2 IRobotModalAnalysisBase Methods**

The methods of the IRobotModalAnalysisBase class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Add ( <a href="#">see page 199</a> )	Function generates a new element defining the base and returns its index. .
≡	Count ( <a href="#">see page 200</a> )	Function returns the number of elements defining the base. Elements defining the base are indexed from 1 to Count.
≡	Delete ( <a href="#">see page 200</a> )	Function deletes the indicated element defining the base.
≡	Get ( <a href="#">see page 200</a> )	Function returns the selected element defining the base. .

**II.1.4.7.2.1 Add****C++**

```
HRESULT Add(BSTR _nodes, VARIANT_BOOL _ux, VARIANT_BOOL _uy, VARIANT_BOOL _uz, long* ret);
```

**C#**

```
public long Add(String _nodes, bool _ux, bool _uy, bool _uz);
```

**Visual Basic**

```
Public Function Add(_nodes As String, _ux As Boolean, _uy As Boolean, _uz As Boolean) As long
```

**Description**

Function generates a new element defining the base and returns its index. .

**Version**

Available since version 2.5.

**II.1.4.7.2.2 Count****C++**

```
HRESULT Count(long* ret);
```

**C#**

```
public long Count();
```

**Visual Basic**

```
Public Function Count() As long
```

**Description**

Function returns the number of elements defining the base. Elements defining the base are indexed from 1 to Count.

**Version**

Available since version 2.5.

**II.1.4.7.2.3 Delete****C++**

```
HRESULT Delete(long _idx, VARIANT_BOOL* ret);
```

**C#**

```
public bool Delete(long _idx);
```

**Visual Basic**

```
Public Function Delete(_idx As long) As Boolean
```

**Description**

Function deletes the indicated element defining the base.

**Version**

Available since version 2.5.

**II.1.4.7.2.4 Get****C++**

```
HRESULT Get(long _idx, BSTR* _nodes, long* _ux, long* _uy, long* _uz);
```

**C#**

```
public void Get(long _idx, BSTR* _nodes, long* _ux, long* _uy, long* _uz);
```

**Visual Basic**

```
Public Sub Get(_idx As long, ByRef _nodes As BSTR*, ByRef _ux As long*, ByRef _uy As long*, ByRef _uz As long*)
```

## Description

Function returns the selected element defining the base. .

## Version

Available since version 2.5.

### II.1.4.8 IRobotModalAnalysisLimits

#### Class Hierarchy

#### C++

```
interface IRobotModalAnalysisLimits : IDispatch;
```

#### C#

```
public interface IRobotModalAnalysisLimits;
```

#### Visual Basic

```
Public Interface IRobotModalAnalysisLimits
```

#### Description

Definition of a limit range.

## Version

Available since version 2.5.

### II.1.4.8.1 IRobotModalAnalysisLimits Members

The following tables list the members exposed by IRobotModalAnalysisLimits.

#### Public Fields

	Name	Description
❖	FrequencyLimitValue ( <a href="#">see page 202</a> )	Limit frequency value.
❖	PeriodLimitValue ( <a href="#">see page 202</a> )	Limit period value.
❖	PulsationLimitValue ( <a href="#">see page 202</a> )	Limit pulsation value.

#### Public Methods

	Name	Description
❖	DefineLimits ( <a href="#">see page 203</a> )	Function defines the limit range based on the specified parameters. .

### II.1.4.8.2 IRobotModalAnalysisLimits Fields

The fields of the IRobotModalAnalysisLimits class are listed here.

#### Public Fields

	Name	Description
❖	FrequencyLimitValue ( <a href="#">see page 202</a> )	Limit frequency value.
❖	PeriodLimitValue ( <a href="#">see page 202</a> )	Limit period value.
❖	PulsationLimitValue ( <a href="#">see page 202</a> )	Limit pulsation value.

### II.1.4.8.2.1 FrequencyLimitValue

#### C++

```
HRESULT get_FrequencyLimitValue(double*);
```

#### C#

```
public double FrequencyLimitValue { get; }
```

#### Visual Basic

```
Public ReadOnly FrequencyLimitValue As double
```

#### Description

Limit frequency value.

#### Version

Available since version 2.5.

### II.1.4.8.2.2 PeriodLimitValue

#### C++

```
HRESULT get_PeriodLimitValue(double*);
```

#### C#

```
public double PeriodLimitValue { get; }
```

#### Visual Basic

```
Public ReadOnly PeriodLimitValue As double
```

#### Description

Limit period value.

#### Version

Available since version 2.5.

### II.1.4.8.2.3 PulsationLimitValue

#### C++

```
HRESULT get_PulsationLimitValue(double*);
```

#### C#

```
public double PulsationLimitValue { get; }
```

#### Visual Basic

```
Public ReadOnly PulsationLimitValue As double
```

#### Description

Limit pulsation value.

#### Version

Available since version 2.5.

### II.1.4.8.3 IRobotModalAnalysisLimits Methods

The methods of the IRobotModalAnalysisLimits class are listed here.

**Public Methods**

	Name	Description
💡	DefineLimits (see page 203)	Function defines the limit range based on the specified parameters..

**II.1.4.8.3.1 DefineLimits****C++**

```
HRESULT DefineLimits(IRobotModalAnalysisLimitType _limit_type, double _value);
```

**C#**

```
public void DefineLimits(IRobotModalAnalysisLimitType _limit_type, double _value);
```

**Visual Basic**

```
Public Sub DefineLimits(_limit_type As IRobotModalAnalysisLimitType, _value As double)
```

**Description**

Function defines the limit range based on the specified parameters..

**Version**

Available since version 2.5.

**II.1.4.9 IRobotModalAnalysisShifts****Class Hierarchy****C++**

```
interface IRobotModalAnalysisShifts : IDispatch;
```

**C#**

```
public interface IRobotModalAnalysisShifts;
```

**Visual Basic**

```
Public Interface IRobotModalAnalysisShifts
```

**Description**

Shift definition.

**Version**

Available since version 2.5.

**II.1.4.9.1 IRobotModalAnalysisShifts Members**

The following tables list the members exposed by IRobotModalAnalysisShifts.

**Public Fields**

	Name	Description
💡	IterationsCount (see page 204)	Number of iterations between shifts.

**Public Methods**

	Name	Description
💡	SetDefault (see page 204)	Function sets the default value for the number of iterations between shifts.

**II.1.4.9.2 IRobotModalAnalysisShifts Fields**

The fields of the IRobotModalAnalysisShifts class are listed here.

## Public Fields

	Name	Description
◆	IterationsCount (see page 204)	Number of iterations between shifts.

### II.1.4.9.2.1 IterationsCount

#### C++

```
HRESULT get_IterationsCount(long* );
HRESULT put_IterationsCount(long);
```

#### C#

```
public long IterationsCount { get; set; }
```

#### Visual Basic

```
Public IterationsCount As long
```

#### Description

Number of iterations between shifts.

#### Version

Available since version 2.5.

### II.1.4.9.3 IRobotModalAnalysisShifts Methods

The methods of the IRobotModalAnalysisShifts class are listed here.

#### Public Methods

	Name	Description
◆	SetDefault (see page 204)	Function sets the default value for the number of iterations between shifts.

### II.1.4.9.3.1 SetDefault

#### C++

```
HRESULT SetDefault();
```

#### C#

```
public void SetDefault();
```

#### Visual Basic

```
Public Sub SetDefault()
```

#### Description

Function sets the default value for the number of iterations between shifts.

#### Version

Available since version 2.5.

### II.1.4.10 IRobotMassEccentricities

#### Class Hierarchy

#### C++

```
interface IRobotMassEccentricities : IDispatch;
```

#### C#

```
public interface IRobotMassEccentricities;
```

## Visual Basic

```
Public Interface IRobotMassEccentricities
```

### Description

Definition of eccentricities.

### Version

Available since version 6.21.

## II.1.4.10.1 IRobotMassEccentricities Members

The following tables list the members exposed by IRobotMassEccentricities.

### Public Fields

	Name	Description
❖	IsDirX (see page 205)	Flag indicating if the value for the X direction is defined.
❖	IsDirY (see page 205)	Flag indicating if the value for the Y direction is defined.
❖	RelativeValues (see page 206)	Flag indicating if the specified values are relative or absolute.
❖	ValueDirX (see page 206)	Value for the X direction.
❖	ValueDirY (see page 206)	Value for the Y direction.

## II.1.4.10.2 IRobotMassEccentricities Fields

The fields of the IRobotMassEccentricities class are listed here.

### Public Fields

	Name	Description
❖	IsDirX (see page 205)	Flag indicating if the value for the X direction is defined.
❖	IsDirY (see page 205)	Flag indicating if the value for the Y direction is defined.
❖	RelativeValues (see page 206)	Flag indicating if the specified values are relative or absolute.
❖	ValueDirX (see page 206)	Value for the X direction.
❖	ValueDirY (see page 206)	Value for the Y direction.

## II.1.4.10.2.1 IsDirX

### C++

```
HRESULT get_IsDirX(VARIANT_BOOL* );
HRESULT put_IsDirX(VARIANT_BOOL);
```

### C#

```
public bool IsDirX { get; set; }
```

## Visual Basic

```
Public IsDirX As Boolean
```

### Description

Flag indicating if the value for the X direction is defined.

### Version

Available since version 6.21.

## II.1.4.10.2.2 IsDirY

### C++

```
HRESULT get_IsDirY(VARIANT_BOOL* );
```

```
HRESULT put_IsDirY(VARIANT_BOOL);
```

**C#**

```
public bool IsDirY { get; set; }
```

**Visual Basic**

```
Public IsDirY As Boolean
```

**Description**

Flag indicating if the value for the Y direction is defined.

**Version**

Available since version 6.21.

### II.1.4.10.2.3 RelativeValues

**C++**

```
HRESULT get_RelativeValues(VARIANT_BOOL*);  
HRESULT put_RelativeValues(VARIANT_BOOL);
```

**C#**

```
public bool RelativeValues { get; set; }
```

**Visual Basic**

```
Public RelativeValues As Boolean
```

**Description**

Flag indicating if the specified values are relative or absolute.

**Version**

Available since version 6.21.

### II.1.4.10.2.4 ValueDirX

**C++**

```
HRESULT get_ValueDirX(double*);  
HRESULT put_ValueDirX(double);
```

**C#**

```
public double ValueDirX { get; set; }
```

**Visual Basic**

```
Public ValueDirX As Double
```

**Description**

Value for the X direction.

**Version**

Available since version 6.21.

### II.1.4.10.2.5 ValueDirY

**C++**

```
HRESULT get_ValueDirY(double*);  
HRESULT put_ValueDirY(double);
```

**C#**

```
public double ValueDirY { get; set; }
```

## Visual Basic

```
Public ValueDirY As double
```

### Description

Value for the Y direction.

### Version

Available since version 6.21.

## II.1.5 Seismic analysis parameters

Available since version 2.5.

### Enumerations

	Name	Description
	IRobotSeismicAnalysis_AFPS_90_ZoneType (see page 250)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_CIRSOC_103_ZoneType (see page 251)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_P_100_92_ZoneType (see page 251)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_PS_92_ZoneType (see page 252)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_RPA_88_ZoneType (see page 252)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_UBC_97_ZoneType (see page 252)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_EAK_2000_ZoneType (see page 253)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_TURKISH_23098_ZoneType (see page 253)	Available numbers of seismic zones.
	IRobotSeismicAnalysis_AFPS_90_StructureType (see page 254)	Available structure types.
	IRobotSeismicAnalysis_AFPS_90_SiteType (see page 254)	Available types of structure site.
	IRobotSeismicAnalysisSpectrumType (see page 255)	Available spectrum types. .
	IRobotSeismicAnalysis_CHINESE_StructureType (see page 255)	Available structure types.
	IRobotSeismicAnalysis_CHINESE_SiteType (see page 256)	Available types of structure site.
	IRobotSeismicAnalysis_CHINESE_IntensityType (see page 256)	Available intensity types.
	IRobotSeismicAnalysis_CHINESE_DesignType (see page 256)	Available design standards.
	IRobotSeismicAnalysis_CHINESE_EarthquakeType (see page 257)	Available types of earthquake distance.
	IRobotSeismicAnalysis_CIRSOC_103_SoilType (see page 257)	Available ground types. .
	IRobotSeismicAnalysis_CIRSOC_103_StructureType (see page 258)	Available structure types.
	IRobotSeismicAnalysisDirectionType (see page 258)	Available direction types.
	IRobotSeismicAnalysis_DM_16_1_96_ProtectionCoeffType (see page 259)	Available values of seismic protection coefficients. .

	IRobotSeismicAnalysis_P_100_92_ImportanceClassType ( <a href="#">see page 259</a> )	Available importance classes. .
	IRobotSeismicAnalysis_PS_69_DampingType ( <a href="#">see page 259</a> )	Available damping types. .
	IRobotSeismicAnalysis_PS_69_SoilType ( <a href="#">see page 260</a> )	Available ground types. .
	IRobotSeismicAnalysis_PS_92_StructureType ( <a href="#">see page 260</a> )	Available structure types.
	IRobotSeismicAnalysis_PS_92_SiteType ( <a href="#">see page 261</a> )	Available site types.
	IRobotSeismicAnalysis_RPA_88_UsageType ( <a href="#">see page 261</a> )	Available usage types.
	IRobotSeismicAnalysis_RPA_88_CategoryType ( <a href="#">see page 262</a> )	Available categories.
	IRobotSeismicAnalysis_RPA_88_SoilType ( <a href="#">see page 262</a> )	Available soil types.
	IRobotSeismicAnalysis_TURKISH_23098_SoilType ( <a href="#">see page 263</a> )	Available soil types.
	IRobotSeismicAnalysis_UBC_97_SoilType ( <a href="#">see page 263</a> )	Available soil types.
	IRobotSeismicAnalysis_UBC_97_SourceType ( <a href="#">see page 264</a> )	Available seismic source types.
	IRobotSeismicAnalysis_IBC_2000_SiteClassType ( <a href="#">see page 264</a> )	Available soil types.
	IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType ( <a href="#">see page 264</a> )	Available importance factors. .
	IRobotSeismicAnalysis_EAK_2000_GroundCategoryType ( <a href="#">see page 265</a> )	Available ground categories.
	IRobotSeismicAnalysis_RPS_2000_ZoneType ( <a href="#">see page 267</a> )	
	IRobotSeismicAnalysis_RPS_2000_SiteType ( <a href="#">see page 267</a> )	
	IRobotSeismicAnalysis_RPS_2000_StructureClass ( <a href="#">see page 267</a> )	
	IRobotSeismicAnalysis_RPA_2003_ZoneType ( <a href="#">see page 270</a> )	
	IRobotSeismicAnalysis_RPA_2003_UsageType ( <a href="#">see page 270</a> )	
	IRobotSeismicAnalysis_RPA_2003_SiteType ( <a href="#">see page 270</a> )	
	IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType ( <a href="#">see page 275</a> )	Soil type according to the Italian code Ordinanza 3274.
	IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType ( <a href="#">see page 275</a> )	
	IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum ( <a href="#">see page 276</a> )	
	IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction ( <a href="#">see page 276</a> )	
	IRobotSeismicAnalysis_IBC_2006_SiteClassType ( <a href="#">see page 283</a> )	Available soil types.
	IRobotSeismicAnalysis_PS_92_2008_ZoneType ( <a href="#">see page 289</a> )	Available numbers of seismic zones.
	IRobotSeismicAnalysis_PS_92_2008_StructureType ( <a href="#">see page 289</a> )	Available structure types.
	IRobotSeismicAnalysis_PS_92_2008_SiteType ( <a href="#">see page 289</a> )	Available site types.

	IRobotSeismicResidualModeDefinitionType (see page 291)	
	IRobotSeismicAnalysis_EC8_SpectrumType (see page 298)	
	IRobotSeismicAnalysis_EC8_GroundType (see page 298)	

## Interfaces

	<b>Name</b>	<b>Description</b>
	IRobotSeismicAnalysis_AFPS_90_Params (see page 210)	Seismic analysis parameters for the French AFPS code. .
	IRobotSeismicAnalysis_CHINESE_Params (see page 214)	Seismic analysis parameters for the Chinese code. .
	IRobotSeismicAnalysis_CIRSOC_103_Params (see page 217)	Seismic analysis parameters for the Argentinian code. .
	IRobotSeismicAnalysis_DM_16_1_96_Params (see page 221)	Seismic analysis parameters for the Italian DM 16.1.96 code. .
	IRobotSeismicAnalysis_P_100_92_Params (see page 223)	Seismic analysis parameters for the Romanian P100-92 code. .
	IRobotSeismicAnalysis_PS_69_Params (see page 225)	Seismic analysis parameters for the French PS69 code. .
	IRobotSeismicAnalysis_PS_92_Params (see page 228)	Seismic analysis parameters for the French PS 92 code. .
	IRobotSeismicAnalysis_RPA_88_Params (see page 233)	Seismic analysis parameters for the Algerian RPA 88 code. .
	IRobotSeismicAnalysis_TURKISH_23098_Params (see page 236)	Seismic analysis parameters for the Turkish code. .
	IRobotSeismicAnalysis_UBC_97_Params (see page 239)	Seismic analysis parameters for the American UBC 97 code. .
	IRobotSeismicAnalysis_IBC_2000_Params (see page 242)	Seismic analysis parameters for the IBC 2000 code issued in the USA. .
	IRobotSeismicAnalysis_EAK_2000_Params (see page 246)	Seismic analysis parameters for the Greek E.A.K. 2000 code. .
	IRobotSeismicAnalysis_PS_92_SiteEnvelope (see page 265)	
	IRobotSeismicAnalysis_RPS_2000_Params (see page 267)	Parameters of the Moroccan seismic code R.P.S. 2000.
	IRobotSeismicAnalysis_RPA_2003_Params (see page 270)	Parameters of the Algerian seismic code RPA 99 (2003). .
	IRobotSeismicAnalysis_ITALY_ORDINANZA_Params (see page 273)	Seismic analysis parameters for the Italian code Ordinanza 3274.
	IRobotSeismicAnalysis_P_100_2006_Params (see page 276)	Parameters of seismic analysis for Romanian code (P100-1-2006).
	IRobotSeismicAnalysis_IBC_2006_Params (see page 279)	Seismic analysis parameters for IBC 2006 released in the USA.
	IRobotSeismicAnalysis_PS_92_2008_Params (see page 284)	Seismic analysis parameters for the French code PS 92 - 2008.
	IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope (see page 290)	
	IRobotSeismicResidualMode (see page 292)	
	IRobotSeismicAnalysis_EC8_General_Params (see page 293)	.
	IRobotSeismicAnalysis_EC8_Params (see page 299)	.

## II.1.5.1 IRobotSeismicAnalysis\_AFPS\_90\_Params

### Class Hierarchy

#### C++

```
interface IRobotSeismicAnalysis_AFPS_90_Params : IDispatch;
```

#### C#

```
public interface IRobotSeismicAnalysis_AFPS_90_Params;
```

### Visual Basic

```
Public Interface IRobotSeismicAnalysis_AFPS_90_Params
```

### Description

Seismic analysis parameters for the French AFPS code. .

### Version

Available since version 2.5.

## II.1.5.1.1 IRobotSeismicAnalysis\_AFPS\_90\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_AFPS\_90\_Params.

### Public Fields

	Name	Description
❖	BehaviorFactor ( <a href="#">see page 211</a> )	Behavior factor.
❖	Direction ( <a href="#">see page 211</a> )	Seismic excitation direction.
❖	DirectionType ( <a href="#">see page 211</a> )	Direction ( <a href="#">see page 211</a> ).
❖	Filter ( <a href="#">see page 212</a> )	Description of modes taken into account during structure dynamic analysis.
❖	ResidualMode ( <a href="#">see page 212</a> )	
❖	Site ( <a href="#">see page 212</a> )	Structure site type.
❖	SpectrumType ( <a href="#">see page 212</a> )	Spectrum type.
❖	StructureType ( <a href="#">see page 213</a> )	Structure type.
❖	Topography ( <a href="#">see page 213</a> )	Topography.
❖	ZoneType ( <a href="#">see page 213</a> )	Seismic zone number.

## II.1.5.1.2 IRobotSeismicAnalysis\_AFPS\_90\_Params Fields

The fields of the IRobotSeismicAnalysis\_AFPS\_90\_Params class are listed here.

### Public Fields

	Name	Description
❖	BehaviorFactor ( <a href="#">see page 211</a> )	Behavior factor.
❖	Direction ( <a href="#">see page 211</a> )	Seismic excitation direction.
❖	DirectionType ( <a href="#">see page 211</a> )	Direction ( <a href="#">see page 211</a> ).
❖	Filter ( <a href="#">see page 212</a> )	Description of modes taken into account during structure dynamic analysis.
❖	ResidualMode ( <a href="#">see page 212</a> )	
❖	Site ( <a href="#">see page 212</a> )	Structure site type.
❖	SpectrumType ( <a href="#">see page 212</a> )	Spectrum type.
❖	StructureType ( <a href="#">see page 213</a> )	Structure type.
❖	Topography ( <a href="#">see page 213</a> )	Topography.
❖	ZoneType ( <a href="#">see page 213</a> )	Seismic zone number.

### II.1.5.1.2.1 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double*);  
HRESULT put_BehaviorFactor(double);
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

#### Description

Behavior factor.

#### Version

Available since version 2.5.

### II.1.5.1.2.2 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D**);  
HRESULT put_Direction(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 2.5.

### II.1.5.1.2.3 DirectionType

#### C++

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType*);  
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

#### C#

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

#### Visual Basic

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

#### Description

Direction (see page 211).

#### Version

Available since version 2.5.

#### II.1.5.1.2.4 Filter

##### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

##### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

##### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

##### Description

Description of modes taken into account during structure dynamic analysis.

##### Version

Available since version 2.5.

#### II.1.5.1.2.5 ResidualMode

##### C++

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

##### C#

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

##### Visual Basic

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

##### Version

Available since version 14.7.

#### II.1.5.1.2.6 Site

##### C++

```
HRESULT get_Site(IRobotSeismicAnalysis_AFPS_90_SiteType* );
HRESULT put_Site(IRobotSeismicAnalysis_AFPS_90_SiteType);
```

##### C#

```
public IRobotSeismicAnalysis_AFPS_90_SiteType Site { get; set; }
```

##### Visual Basic

```
Public Site As IRobotSeismicAnalysis_AFPS_90_SiteType
```

##### Description

Structure site type.

##### Version

Available since version 2.5.

#### II.1.5.1.2.7 SpectrumType

##### C++

```
HRESULT get_SpectrumType(IRobotSeismicAnalysisSpectrumType* );
HRESULT put_SpectrumType(IRobotSeismicAnalysisSpectrumType);
```

**C#**

```
public IRobotSeismicAnalysisSpectrumType SpectrumType { get; set; }
```

**Visual Basic**

```
Public SpectrumType As IRobotSeismicAnalysisSpectrumType
```

**Description**

Spectrum type.

**Version**

Available since version 2.5.

## II.1.5.1.2.8 StructureType

**C++**

```
HRESULT get_StructureType(IRobotSeismicAnalysis_AFPS_90_StructureType* );
HRESULT put_StructureType(IRobotSeismicAnalysis_AFPS_90_StructureType);
```

**C#**

```
public IRobotSeismicAnalysis_AFPS_90_StructureType StructureType { get; set; }
```

**Visual Basic**

```
Public StructureType As IRobotSeismicAnalysis_AFPS_90_StructureType
```

**Description**

Structure type.

**Version**

Available since version 2.5.

## II.1.5.1.2.9 Topography

**C++**

```
HRESULT get_Topography(double* );
HRESULT put_Topography(double);
```

**C#**

```
public double Topography { get; set; }
```

**Visual Basic**

```
Public Topography As Double
```

**Description**

Topography.

**Version**

Available since version 2.5.

## II.1.5.1.2.10 ZoneType

**C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_AFPS_90_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_AFPS_90_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_AFPS_90_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_AFPS_90_ZoneType
```

**Description**

Seismic zone number.

**Version**

Available since version 2.5.

**II.1.5.2 IRobotSeismicAnalysis\_CHINESE\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_CHINESE_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_CHINESE_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_CHINESE_Params
```

**Description**

Seismic analysis parameters for the Chinese code. .

**Version**

Available since version 2.5.

**II.1.5.2.1 IRobotSeismicAnalysis\_CHINESE\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_CHINESE\_Params.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DesignStandard (see page 215)	Design standard.
◆	Direction (see page 215)	Seismic excitation direction.
◆	EarthquakeType (see page 215)	Earthquake distance.
◆	Factor (see page 216)	Factor.
◆	Filter (see page 216)	Description of modes taken into account during structure dynamic analysis.
◆	Intensity (see page 216)	Intensity.
◆	Site (see page 217)	Site.
◆	SiteTg (see page 217)	Site (see page 217).
◆	StructureType (see page 217)	Structure type.

**II.1.5.2.2 IRobotSeismicAnalysis\_CHINESE\_Params Fields**

The fields of the IRobotSeismicAnalysis\_CHINESE\_Params class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DesignStandard (see page 215)	Design standard.
◆	Direction (see page 215)	Seismic excitation direction.
◆	EarthquakeType (see page 215)	Earthquake distance.
◆	Factor (see page 216)	Factor.

◆	Filter (see page 216)	Description of modes taken into account during structure dynamic analysis.
◆	Intensity (see page 216)	Intensity.
◆	Site (see page 217)	Site.
◆	SiteTg (see page 217)	Site (see page 217).
◆	StructureType (see page 217)	Structure type.

### II.1.5.2.2.1 DesignStandard

#### C++

```
HRESULT get_DesignStandard(IRobotSeismicAnalysis_CHINESE_DesignType* );
HRESULT put_DesignStandard(IRobotSeismicAnalysis_CHINESE_DesignType);
```

#### C#

```
public IRobotSeismicAnalysis_CHINESE_DesignType DesignStandard { get; set; }
```

#### Visual Basic

```
Public DesignStandard As IRobotSeismicAnalysis_CHINESE_DesignType
```

#### Description

Design standard.

#### Version

Available since version 2.5.

### II.1.5.2.2.2 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 2.5.

### II.1.5.2.2.3 EarthquakeType

#### C++

```
HRESULT get_EarthquakeType(IRobotSeismicAnalysis_CHINESE_EarthquakeType* );
HRESULT put_EarthquakeType(IRobotSeismicAnalysis_CHINESE_EarthquakeType);
```

#### C#

```
public IRobotSeismicAnalysis_CHINESE_EarthquakeType EarthquakeType { get; set; }
```

#### Visual Basic

```
Public EarthquakeType As IRobotSeismicAnalysis_CHINESE_EarthquakeType
```

#### Description

Earthquake distance.

**Version**

Available since version 2.5.

**II.1.5.2.2.4 Factor****C++**

```
HRESULT get_Factor(double*);  
HRESULT put_Factor(double);
```

**C#**

```
public double Factor { get; set; }
```

**Visual Basic**

```
Public Factor As Double
```

**Description**

Factor.

**Version**

Available since version 2.5.

**II.1.5.2.2.5 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);  
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

**II.1.5.2.2.6 Intensity****C++**

```
HRESULT get_Intensity(IRobotSeismicAnalysis_CHINESE_IntensityType*);  
HRESULT put_Intensity(IRobotSeismicAnalysis_CHINESE_IntensityType);
```

**C#**

```
public IRobotSeismicAnalysis_CHINESE_IntensityType Intensity { get; set; }
```

**Visual Basic**

```
Public Intensity As IRobotSeismicAnalysis_CHINESE_IntensityType
```

**Description**

Intensity.

**Version**

Available since version 2.5.

### II.1.5.2.2.7 Site

#### C++

```
HRESULT get_Site(IRobotSeismicAnalysis_CHINESE_SiteType* );
HRESULT put_Site(IRobotSeismicAnalysis_CHINESE_SiteType);
```

#### C#

```
public IRobotSeismicAnalysis_CHINESE_SiteType Site { get; set; }
```

#### Visual Basic

```
Public Site As IRobotSeismicAnalysis_CHINESE_SiteType
```

#### Description

Site.

#### Version

Available since version 2.5.

### II.1.5.2.2.8 SiteTg

#### C++

```
HRESULT get_SiteTg(double* );
HRESULT put_SiteTg(double);
```

#### C#

```
public double SiteTg { get; set; }
```

#### Visual Basic

```
Public SiteTg As Double
```

#### Description

Site (see page 217).

#### Version

Available since version 2.5.

### II.1.5.2.2.9 StructureType

#### C++

```
HRESULT get_StructureType(IRobotSeismicAnalysis_CHINESE_StructureType* );
HRESULT put_StructureType(IRobotSeismicAnalysis_CHINESE_StructureType);
```

#### C#

```
public IRobotSeismicAnalysis_CHINESE_StructureType StructureType { get; set; }
```

#### Visual Basic

```
Public StructureType As IRobotSeismicAnalysis_CHINESE_StructureType
```

#### Description

Structure type.

#### Version

Available since version 2.5.

## II.1.5.3 IRobotSeismicAnalysis\_CIRSOC\_103\_Params

#### Class Hierarchy

**C++**

```
interface IRobotSeismicAnalysis_CIRSOC_103_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_CIRSOC_103_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_CIRSOC_103_Params
```

**Description**

Seismic analysis parameters for the Argentinian code. .

**Version**

Available since version 2.5.

**II.1.5.3.1 IRobotSeismicAnalysis\_CIRSOC\_103\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_CIRSOC\_103\_Params.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Direction ( <a href="#">see page 218</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 219</a> )	Direction ( <a href="#">see page 218</a> ).
◆	Filter ( <a href="#">see page 219</a> )	Description of modes taken into account during structure dynamic analysis.
◆	PlasticityCoeff ( <a href="#">see page 219</a> )	Global plasticity coefficient.
◆	Soil ( <a href="#">see page 220</a> )	
◆	StructureType ( <a href="#">see page 220</a> )	Structure type.
◆	ZoneType ( <a href="#">see page 220</a> )	Seismic zone number.

**II.1.5.3.2 IRobotSeismicAnalysis\_CIRSOC\_103\_Params Fields**

The fields of the IRobotSeismicAnalysis\_CIRSOC\_103\_Params class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Direction ( <a href="#">see page 218</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 219</a> )	Direction ( <a href="#">see page 218</a> ).
◆	Filter ( <a href="#">see page 219</a> )	Description of modes taken into account during structure dynamic analysis.
◆	PlasticityCoeff ( <a href="#">see page 219</a> )	Global plasticity coefficient.
◆	Soil ( <a href="#">see page 220</a> )	
◆	StructureType ( <a href="#">see page 220</a> )	Structure type.
◆	ZoneType ( <a href="#">see page 220</a> )	Seismic zone number.

**II.1.5.3.2.1 Direction****C++**

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

**II.1.5.3.2.2 DirectionType****C++**

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType* );
```

**C#**

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

**Visual Basic**

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

**Description**

Direction (see page 218).

**Version**

Available since version 2.5.

**II.1.5.3.2.3 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

**II.1.5.3.2.4 PlasticityCoeff****C++**

```
HRESULT get_PlasticityCoeff(double* );
HRESULT put_PlasticityCoeff(double );
```

**C#**

```
public double PlasticityCoeff { get; set; }
```

**Visual Basic**

```
Public PlasticityCoeff As double
```

**Description**

Global plasticity coefficient.

**Version**

Available since version 2.5.

**II.1.5.3.2.5 Soil****C++**

```
HRESULT get_Soil(RobotSeismicAnalysis_CIRSOC_103_SoilType* );
HRESULT put_Soil(RobotSeismicAnalysis_CIRSOC_103_SoilType);
```

**C#**

```
public RobotSeismicAnalysis_CIRSOC_103_SoilType Soil { get; set; }
```

**Visual Basic**

```
Public Soil As RobotSeismicAnalysis_CIRSOC_103_SoilType
```

**Version**

Available since version 2.5.

**II.1.5.3.2.6 StructureType****C++**

```
HRESULT get_StructureType(IRobotSeismicAnalysis_CIRSOC_103_StructureType* );
HRESULT put_StructureType(IRobotSeismicAnalysis_CIRSOC_103_StructureType);
```

**C#**

```
public IRobotSeismicAnalysis_CIRSOC_103_StructureType StructureType { get; set; }
```

**Visual Basic**

```
Public StructureType As IRobotSeismicAnalysis_CIRSOC_103_StructureType
```

**Description**

Structure type.

**Version**

Available since version 2.5.

**II.1.5.3.2.7 ZoneType****C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_CIRSOC_103_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_CIRSOC_103_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_CIRSOC_103_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_CIRSOC_103_ZoneType
```

**Description**

Seismic zone number.

**Version**

Available since version 2.5.

## II.1.5.4 IRobotSeismicAnalysis\_DM\_16\_1\_96\_Params

### Class Hierarchy

#### C++

```
interface IRobotSeismicAnalysis_DM_16_1_96_Params : IDispatch;
```

#### C#

```
public interface IRobotSeismicAnalysis_DM_16_1_96_Params;
```

### Visual Basic

```
Public Interface IRobotSeismicAnalysis_DM_16_1_96_Params
```

### Description

Seismic analysis parameters for the Italian DM 16.1.96 code.. .

### Version

Available since version 2.5.

## II.1.5.4.1 IRobotSeismicAnalysis\_DM\_16\_1\_96\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_DM\_16\_1\_96\_Params.

### Public Fields

	Name	Description
◆	Direction ( <a href="#">see page 221</a> )	Seismic excitation direction.
◆	Filter ( <a href="#">see page 222</a> )	Description of modes taken into account during structure dynamic analysis.
◆	SeismicCoeff ( <a href="#">see page 222</a> )	Seismic coefficient.
◆	SeismicProtectionCoeff ( <a href="#">see page 222</a> )	Seismic protection coefficient.

## II.1.5.4.2 IRobotSeismicAnalysis\_DM\_16\_1\_96\_Params Fields

The fields of the IRobotSeismicAnalysis\_DM\_16\_1\_96\_Params class are listed here.

### Public Fields

	Name	Description
◆	Direction ( <a href="#">see page 221</a> )	Seismic excitation direction.
◆	Filter ( <a href="#">see page 222</a> )	Description of modes taken into account during structure dynamic analysis.
◆	SeismicCoeff ( <a href="#">see page 222</a> )	Seismic coefficient.
◆	SeismicProtectionCoeff ( <a href="#">see page 222</a> )	Seismic protection coefficient.

## II.1.5.4.2.1 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

**II.1.5.4.2.2 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);  
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

**II.1.5.4.2.3 SeismicCoeff****C++**

```
HRESULT get_SeismicCoeff(double*);  
HRESULT put_SeismicCoeff(double);
```

**C#**

```
public double SeismicCoeff { get; set; }
```

**Visual Basic**

```
Public SeismicCoeff As Double
```

**Description**

Seismic coefficient.

**Version**

Available since version 2.5.

**II.1.5.4.2.4 SeismicProtectionCoeff****C++**

```
HRESULT get_SeismicProtectionCoeff(RobotSeismicAnalysisProtectionCoeffType*);  
HRESULT put_SeismicProtectionCoeff(RobotSeismicAnalysisProtectionCoeffType);
```

**C#**

```
public RobotSeismicAnalysisProtectionCoeffType SeismicProtectionCoeff { get; set; }
```

**Visual Basic**

```
Public SeismicProtectionCoeff As RobotSeismicAnalysisProtectionCoeffType
```

**Description**

Seismic protection coefficient.

**Version**

Available since version 2.5.

**II.1.5.5 IRobotSeismicAnalysis\_P\_100\_92\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_P_100_92_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_P_100_92_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_P_100_92_Params
```

**Description**

Seismic analysis parameters for the Romanian P100-92 code. .

**Version**

Available since version 2.5.

**II.1.5.5.1 IRobotSeismicAnalysis\_P\_100\_92\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_P\_100\_92\_Params.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Direction ( <a href="#">see page 224</a> )	Seismic excitation direction.
❖	ExcitationDir ( <a href="#">see page 224</a> )	
❖	Filter ( <a href="#">see page 224</a> )	Description of modes taken into account during structure dynamic analysis.
❖	ImportanceClass ( <a href="#">see page 224</a> )	Importance factor.
❖	Psi ( <a href="#">see page 225</a> )	
❖	Tc ( <a href="#">see page 225</a> )	
❖	ZoneType ( <a href="#">see page 225</a> )	Sesmic zone number.

**II.1.5.5.2 IRobotSeismicAnalysis\_P\_100\_92\_Params Fields**

The fields of the IRobotSeismicAnalysis\_P\_100\_92\_Params class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Direction ( <a href="#">see page 224</a> )	Seismic excitation direction.
❖	ExcitationDir ( <a href="#">see page 224</a> )	
❖	Filter ( <a href="#">see page 224</a> )	Description of modes taken into account during structure dynamic analysis.
❖	ImportanceClass ( <a href="#">see page 224</a> )	Importance factor.
❖	Psi ( <a href="#">see page 225</a> )	
❖	Tc ( <a href="#">see page 225</a> )	
❖	ZoneType ( <a href="#">see page 225</a> )	Sesmic zone number.

### II.1.5.5.2.1 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 2.5.

### II.1.5.5.2.2 ExcitationDir

#### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

#### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

#### Visual Basic

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

### II.1.5.5.2.3 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during structure dynamic analysis.

#### Version

Available since version 2.5.

### II.1.5.5.2.4 ImportanceClass

#### C++

```
HRESULT get_ImportanceClass(IRobotSeismicAnalysis_P_100_92_ImportanceClassType* );
HRESULT put_ImportanceClass(IRobotSeismicAnalysis_P_100_92_ImportanceClassType);
```

#### C#

```
public IRobotSeismicAnalysis_P_100_92_ImportanceClassType ImportanceClass { get; set; }
```

#### Visual Basic

```
Public ImportanceClass As IRobotSeismicAnalysis_P_100_92_ImportanceClassType
```

**Description**

Importance factor.

**Version**

Available since version 2.5.

**II.1.5.5.2.5 Psi****C++**

```
HRESULT get_Psi(double*);  
HRESULT put_Psi(double);
```

**C#**

```
public double Psi { get; set; }
```

**Visual Basic**

```
Public Psi As Double
```

**Version**

Available since version 2.5.

**II.1.5.5.2.6 Tc****C++**

```
HRESULT get_Tc(double*);  
HRESULT put_Tc(double);
```

**C#**

```
public double Tc { get; set; }
```

**Visual Basic**

```
Public Tc As Double
```

**Version**

Available since version 2.5.

**II.1.5.5.2.7 ZoneType****C++**

```
HRESULT get_ZoneType(RobotSeismicAnalysis_P100_92_ZoneType*);  
HRESULT put_ZoneType(RobotSeismicAnalysis_P100_92_ZoneType);
```

**C#**

```
public RobotSeismicAnalysis_P100_92_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As RobotSeismicAnalysis_P100_92_ZoneType
```

**Description**

Sesmic zone number.

**Version**

Available since version 2.5.

**II.1.5.6 IRobotSeismicAnalysis\_PS\_69\_Params****Class Hierarchy**

**C++**

```
interface IRobotSeismicAnalysis_PS_69_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_PS_69_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_PS_69_Params
```

**Description**

Seismic analysis parameters for the French PS69 code. .

**Version**

Available since version 2.5.

**II.1.5.6.1 IRobotSeismicAnalysis\_PS\_69\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_PS\_69\_Params.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Alpha (❑ see page 226)	Seismic intensity factor.
❖	Damping (❑ see page 227)	Damping.
❖	Delta (❑ see page 227)	Foundation coefficient.
❖	Direction (❑ see page 227)	Seismic excitation direction.
❖	Filter (❑ see page 228)	Description of modes taken into account during structure dynamic analysis.
❖	Soil (❑ see page 228)	Soil.

**II.1.5.6.2 IRobotSeismicAnalysis\_PS\_69\_Params Fields**

The fields of the IRobotSeismicAnalysis\_PS\_69\_Params class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Alpha (❑ see page 226)	Seismic intensity factor.
❖	Damping (❑ see page 227)	Damping.
❖	Delta (❑ see page 227)	Foundation coefficient.
❖	Direction (❑ see page 227)	Seismic excitation direction.
❖	Filter (❑ see page 228)	Description of modes taken into account during structure dynamic analysis.
❖	Soil (❑ see page 228)	Soil.

**II.1.5.6.2.1 Alpha****C++**

```
HRESULT get_Alpha(double* );
HRESULT put_Alpha(double);
```

**C#**

```
public double Alpha { get; set; }
```

**Visual Basic**

```
Public Alpha As Double
```

**Description**

Seismic intensity factor.

**Version**

Available since version 2.5.

**II.1.5.6.2.2 Damping****C++**

```
HRESULT get_Damping(IRobotSeismicAnalysis_PS_69_DampingType*);  
HRESULT put_Damping(IRobotSeismicAnalysis_PS_69_DampingType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_69_DampingType Damping { get; set; }
```

**Visual Basic**

```
Public Damping As IRobotSeismicAnalysis_PS_69_DampingType
```

**Description**

Damping.

**Version**

Available since version 2.5.

**II.1.5.6.2.3 Delta****C++**

```
HRESULT get_Delta(double*);  
HRESULT put_Delta(double);
```

**C#**

```
public double Delta { get; set; }
```

**Visual Basic**

```
Public Delta As Double
```

**Description**

Foundation coefficient.

**Version**

Available since version 2.5.

**II.1.5.6.2.4 Direction****C++**

```
HRESULT get_Direction(IRobotGeoPoint3D**);  
HRESULT put_Direction(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

**II.1.5.6.2.5 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);  
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

**II.1.5.6.2.6 Soil****C++**

```
HRESULT get_Soil(IRobotSeismicAnalysis_PS_69_SoilType*);  
HRESULT put_Soil(IRobotSeismicAnalysis_PS_69_SoilType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_69_SoilType Soil { get; set; }
```

**Visual Basic**

```
Public Soil As IRobotSeismicAnalysis_PS_69_SoilType
```

**Description**

Soil.

**Version**

Available since version 2.5.

**II.1.5.7 IRobotSeismicAnalysis\_PS\_92\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_PS_92_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_PS_92_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_PS_92_Params
```

**Description**

Seismic analysis parameters for the French PS 92 code. .

**Version**

Available since version 2.5.

### II.1.5.7.1 IRobotSeismicAnalysis\_PS\_92\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_PS\_92\_Params.

#### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 229</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 230</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 230</a> )	Direction ( <a href="#">see page 230</a> ).
◆	ExcitationDir ( <a href="#">see page 230</a> )	Definition of excitation direction.
◆	Filter ( <a href="#">see page 231</a> )	Description of modes taken into account during structure dynamic analysis.
◆	ResidualMode ( <a href="#">see page 231</a> )	
◆	Site ( <a href="#">see page 231</a> )	Site.
◆	SiteEnvelope ( <a href="#">see page 231</a> )	
◆	SpectrumType ( <a href="#">see page 232</a> )	Spectrum.
◆	StructureType ( <a href="#">see page 232</a> )	Structure type.
◆	Topography ( <a href="#">see page 232</a> )	Topography.
◆	ZoneType ( <a href="#">see page 233</a> )	Seismic zone number.

### II.1.5.7.2 IRobotSeismicAnalysis\_PS\_92\_Params Fields

The fields of the IRobotSeismicAnalysis\_PS\_92\_Params class are listed here.

#### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 229</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 230</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 230</a> )	Direction ( <a href="#">see page 230</a> ).
◆	ExcitationDir ( <a href="#">see page 230</a> )	Definition of excitation direction.
◆	Filter ( <a href="#">see page 231</a> )	Description of modes taken into account during structure dynamic analysis.
◆	ResidualMode ( <a href="#">see page 231</a> )	
◆	Site ( <a href="#">see page 231</a> )	Site.
◆	SiteEnvelope ( <a href="#">see page 231</a> )	
◆	SpectrumType ( <a href="#">see page 232</a> )	Spectrum.
◆	StructureType ( <a href="#">see page 232</a> )	Structure type.
◆	Topography ( <a href="#">see page 232</a> )	Topography.
◆	ZoneType ( <a href="#">see page 233</a> )	Seismic zone number.

### II.1.5.7.2.1 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

#### Description

Behavior factor.

**Version**

Available since version 2.5.

**II.1.5.7.2.2 Direction****C++**

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

**II.1.5.7.2.3 DirectionType****C++**

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType* );
```

**C#**

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

**Visual Basic**

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

**Description**

Direction (see page 230).

**Version**

Available since version 2.5.

**II.1.5.7.2.4 ExcitationDir****C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**Description**

Definition of excitation direction.

**Version**

Available since version 5.5.

### II.1.5.7.2.5 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during structure dynamic analysis.

#### Version

Available since version 2.5.

### II.1.5.7.2.6 ResidualMode

#### C++

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

#### C#

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

#### Visual Basic

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

#### Version

Available since version 14.7.

### II.1.5.7.2.7 Site

#### C++

```
HRESULT get_Site(IRobotSeismicAnalysis_PS_92_SiteType* );
HRESULT put_Site(IRobotSeismicAnalysis_PS_92_SiteType);
```

#### C#

```
public IRobotSeismicAnalysis_PS_92_SiteType Site { get; set; }
```

#### Visual Basic

```
Public Site As IRobotSeismicAnalysis_PS_92_SiteType
```

#### Description

Site.

#### Version

Available since version 2.5.

### II.1.5.7.2.8 SiteEnvelope

#### C++

```
HRESULT get_SiteEnvelope(IRobotSeismicAnalysis_PS_92_SiteEnvelope**);
```

#### C#

```
public IRobotSeismicAnalysis_PS_92_SiteEnvelope SiteEnvelope { get; }
```

**Visual Basic**

```
Public ReadOnly SiteEnvelope As IRobotSeismicAnalysis_PS_92_SiteEnvelope
```

**Version**

Available since version 5.5.

**II.1.5.7.2.9 SpectrumType****C++**

```
HRESULT get_SpectrumType(IRobotSeismicAnalysisSpectrumType* );
HRESULT put_SpectrumType(IRobotSeismicAnalysisSpectrumType);
```

**C#**

```
public IRobotSeismicAnalysisSpectrumType SpectrumType { get; set; }
```

**Visual Basic**

```
Public SpectrumType As IRobotSeismicAnalysisSpectrumType
```

**Description**

Spectrum.

**Version**

Available since version 2.5.

**II.1.5.7.2.10 StructureType****C++**

```
HRESULT get_StructureType(IRobotSeismicAnalysis_PS_92_StructureType* );
HRESULT put_StructureType(IRobotSeismicAnalysis_PS_92_StructureType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_92_StructureType StructureType { get; set; }
```

**Visual Basic**

```
Public StructureType As IRobotSeismicAnalysis_PS_92_StructureType
```

**Description**

Structure type.

**Version**

Available since version 2.5.

**II.1.5.7.2.11 Topography****C++**

```
HRESULT get_Topography(double* );
HRESULT put_Topography(double);
```

**C#**

```
public double Topography { get; set; }
```

**Visual Basic**

```
Public Topography As Double
```

**Description**

Topography.

**Version**

Available since version 2.5.

**II.1.5.7.2.12 ZoneType****C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_PS_92_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_PS_92_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_92_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_PS_92_ZoneType
```

**Description**

Seismic zone number.

**Version**

Available since version 2.5.

**II.1.5.8 IRobotSeismicAnalysis\_RPA\_88\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_RPA_88_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_RPA_88_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_RPA_88_Params
```

**Description**

Seismic analysis parameters for the Algerian RPA 88 code..

**Version**

Available since version 2.5.

**II.1.5.8.1 IRobotSeismicAnalysis\_RPA\_88\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_RPA\_88\_Params.

**Public Fields**

	Name	Description
❖	Category (see page 234)	Category.
❖	Direction (see page 234)	Seismic excitation direction.
❖	Filter (see page 234)	Description of modes taken into account during structure dynamic analysis.
❖	QualityFactor (see page 235)	Quality coefficient.
❖	Soil (see page 235)	Soil.
❖	Usage (see page 235)	Usage.
❖	ZoneType (see page 236)	Seismic zone number.

## II.1.5.8.2 IRobotSeismicAnalysis\_RPA\_88\_Params Fields

The fields of the IRobotSeismicAnalysis\_RPA\_88\_Params class are listed here.

### Public Fields

	Name	Description
◆	Category (see page 234)	Category.
◆	Direction (see page 234)	Seismic excitation direction.
◆	Filter (see page 234)	Description of modes taken into account during structure dynamic analysis.
◆	QualityFactor (see page 235)	Quality coefficient.
◆	Soil (see page 235)	Soil.
◆	Usage (see page 235)	Usage.
◆	ZoneType (see page 236)	Seismic zone number.

### II.1.5.8.2.1 Category

#### C++

```
HRESULT get_Category(IRobotSeismicAnalysis_RPA_88_CategoryType* );
HRESULT put_Category(IRobotSeismicAnalysis_RPA_88_CategoryType);
```

#### C#

```
public IRobotSeismicAnalysis_RPA_88_CategoryType Category { get; set; }
```

#### Visual Basic

```
Public Category As IRobotSeismicAnalysis_RPA_88_CategoryType
```

#### Description

Category.

#### Version

Available since version 2.5.

### II.1.5.8.2.2 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 2.5.

### II.1.5.8.2.3 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

### II.1.5.8.2.4 QualityFactor

**C++**

```
HRESULT get_QualityFactor(double*);  
HRESULT put_QualityFactor(double);
```

**C#**

```
public double QualityFactor { get; set; }
```

**Visual Basic**

```
Public QualityFactor As Double
```

**Description**

Quality coefficient.

**Version**

Available since version 2.5.

### II.1.5.8.2.5 Soil

**C++**

```
HRESULT get_Soil(IRobotSeismicAnalysis_RPA_88_SoilType*);  
HRESULT put_Soil(IRobotSeismicAnalysis_RPA_88_SoilType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_88_SoilType Soil { get; set; }
```

**Visual Basic**

```
Public Soil As IRobotSeismicAnalysis_RPA_88_SoilType
```

**Description**

Soil.

**Version**

Available since version 2.5.

### II.1.5.8.2.6 Usage

**C++**

```
HRESULT get_Usage(IRobotSeismicAnalysis_RPA_88_UsageType*);  
HRESULT put_Usage(IRobotSeismicAnalysis_RPA_88_UsageType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_88_UsageType Usage { get; set; }
```

**Visual Basic**

```
Public Usage As IRobotSeismicAnalysis_RPA_88_UsageType
```

**Description**

Usage.

**Version**

Available since version 2.5.

**II.1.5.8.2.7 ZoneType****C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_RPA_88_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_RPA_88_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_88_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_RPA_88_ZoneType
```

**Description**

Seismic zone number.

**Version**

Available since version 2.5.

**II.1.5.9 IRobotSeismicAnalysis\_TURKISH\_23098\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_TURKISH_23098_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_TURKISH_23098_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_TURKISH_23098_Params
```

**Description**

Seismic analysis parameters for the Turkish code. .

**Version**

Available since version 2.5.

**II.1.5.9.1 IRobotSeismicAnalysis\_TURKISH\_23098\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_TURKISH\_23098\_Params.

**Public Fields**

	Name	Description
◆	BehaviorFactor (see page 237)	Behavior factor.
◆	Direction (see page 237)	Seismic excitation direction.
◆	Filter (see page 238)	Description of modes taken into account during structure dynamic analysis.

❖	SoilType ( <a href="#">see page 238</a> )	Soil class.
❖	StructureImportance ( <a href="#">see page 238</a> )	Building importance coefficient.
❖	ZoneType ( <a href="#">see page 239</a> )	Seismic zone number.

### II.1.5.9.2 IRobotSeismicAnalysis\_TURKISH\_23098\_Params Fields

The fields of the IRobotSeismicAnalysis\_TURKISH\_23098\_Params class are listed here.

#### Public Fields

	Name	Description
❖	BehaviorFactor ( <a href="#">see page 237</a> )	Behavior factor.
❖	Direction ( <a href="#">see page 237</a> )	Seismic excitation direction.
❖	Filter ( <a href="#">see page 238</a> )	Description of modes taken into account during structure dynamic analysis.
❖	SoilType ( <a href="#">see page 238</a> )	Soil class.
❖	StructureImportance ( <a href="#">see page 238</a> )	Building importance coefficient.
❖	ZoneType ( <a href="#">see page 239</a> )	Seismic zone number.

#### II.1.5.9.2.1 BehaviorFactor

##### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

##### C#

```
public double BehaviorFactor { get; set; }
```

##### Visual Basic

```
Public BehaviorFactor As Double
```

##### Description

Behavior factor.

##### Version

Available since version 2.5.

#### II.1.5.9.2.2 Direction

##### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

##### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

##### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

##### Description

Seismic excitation direction.

##### Version

Available since version 2.5.

### II.1.5.9.2.3 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during structure dynamic analysis.

#### Version

Available since version 2.5.

### II.1.5.9.2.4 SoilType

#### C++

```
HRESULT get_SoilType(IRobotSeismicAnalysis_TURKISH_23098_SoilType* );
HRESULT put_SoilType(IRobotSeismicAnalysis_TURKISH_23098_SoilType);
```

#### C#

```
public IRobotSeismicAnalysis_TURKISH_23098_SoilType SoilType { get; set; }
```

#### Visual Basic

```
Public SoilType As IRobotSeismicAnalysis_TURKISH_23098_SoilType
```

#### Description

Soil class.

#### Version

Available since version 2.5.

### II.1.5.9.2.5 StructureImportance

#### C++

```
HRESULT get_StructureImportance(double* );
HRESULT put_StructureImportance(double);
```

#### C#

```
public double StructureImportance { get; set; }
```

#### Visual Basic

```
Public StructureImportance As Double
```

#### Description

Building importance coefficient.

#### Version

Available since version 2.5.

## II.1.5.9.2.6 ZoneType

### C++

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_TURKISH_23098_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_TURKISH_23098_ZoneType);
```

### C#

```
public IRobotSeismicAnalysis_TURKISH_23098_ZoneType ZoneType { get; set; }
```

### Visual Basic

```
Public ZoneType As IRobotSeismicAnalysis_TURKISH_23098_ZoneType
```

### Description

Seismic zone number.

### Version

Available since version 2.5.

## II.1.5.10 IRobotSeismicAnalysis\_UBC\_97\_Params

### Class Hierarchy

### C++

```
interface IRobotSeismicAnalysis_UBC_97_Params : IDispatch;
```

### C#

```
public interface IRobotSeismicAnalysis_UBC_97_Params;
```

### Visual Basic

```
Public Interface IRobotSeismicAnalysis_UBC_97_Params
```

### Description

Seismic analysis parameters for the American UBC 97 code..

### Version

Available since version 2.5.

## II.1.5.10.1 IRobotSeismicAnalysis\_UBC\_97\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_UBC\_97\_Params.

### Public Fields

	Name	Description
◆	BehaviorFactor (see page 240)	Behavior factor.
◆	ClosestDistance (see page 240)	Minimum distance to a known seismic source (in km).
◆	Direction (see page 240)	Seismic excitation direction.
◆	ExcitationDir (see page 241)	
◆	Filter (see page 241)	Description of modes taken into account during structure dynamic analysis.
◆	I (see page 241)	Structure importance factor.
◆	Soil (see page 242)	Soil.
◆	Source (see page 242)	Seismic source type.
◆	ZoneType (see page 242)	Seismic zone number.

## II.1.5.10.2 IRobotSeismicAnalysis\_UBC\_97\_Params Fields

The fields of the IRobotSeismicAnalysis\_UBC\_97\_Params class are listed here.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 240</a> )	Behavior factor.
◆	ClosestDistance ( <a href="#">see page 240</a> )	Minimum distance to a known seismic source (in km).
◆	Direction ( <a href="#">see page 240</a> )	Seismic excitation direction.
◆	ExcitationDir ( <a href="#">see page 241</a> )	
◆	Filter ( <a href="#">see page 241</a> )	Description of modes taken into account during structure dynamic analysis.
◆	I ( <a href="#">see page 241</a> )	Structure importance factor.
◆	Soil ( <a href="#">see page 242</a> )	Soil.
◆	Source ( <a href="#">see page 242</a> )	Seismic source type.
◆	ZoneType ( <a href="#">see page 242</a> )	Seismic zone number.

## II.1.5.10.2.1 BehaviorFactor

### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

### C#

```
public double BehaviorFactor { get; set; }
```

### Visual Basic

```
Public BehaviorFactor As Double
```

### Description

Behavior factor.

## II.1.5.10.2.2 ClosestDistance

### C++

```
HRESULT get_ClosestDistance(double* );
HRESULT put_ClosestDistance(double);
```

### C#

```
public double ClosestDistance { get; set; }
```

### Visual Basic

```
Public ClosestDistance As Double
```

### Description

Minimum distance to a known seismic source (in km).

### Version

Available since version 2.5.

## II.1.5.10.2.3 Direction

### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

**II.1.5.10.2.4 ExcitationDir****C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**II.1.5.10.2.5 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

**II.1.5.10.2.6 I****C++**

```
HRESULT get_I(double*);
HRESULT put_I(double);
```

**C#**

```
public double I { get; set; }
```

**Visual Basic**

```
Public I As double
```

**Description**

Structure importance factor.

**Version**

Available since version 8.

### II.1.5.10.2.7 Soil

#### C++

```
HRESULT get_Soil(IRobotSeismicAnalysis_UBC_97_SoilType* );
HRESULT put_Soil(IRobotSeismicAnalysis_UBC_97_SoilType);
```

#### C#

```
public IRobotSeismicAnalysis_UBC_97_SoilType Soil { get; set; }
```

#### Visual Basic

```
Public Soil As IRobotSeismicAnalysis_UBC_97_SoilType
```

#### Description

Soil.

#### Version

Available since version 2.5.

### II.1.5.10.2.8 Source

#### C++

```
HRESULT get_Source(IRobotSeismicAnalysis_UBC_97_SourceType* );
HRESULT put_Source(IRobotSeismicAnalysis_UBC_97_SourceType);
```

#### C#

```
public IRobotSeismicAnalysis_UBC_97_SourceType Source { get; set; }
```

#### Visual Basic

```
Public Source As IRobotSeismicAnalysis_UBC_97_SourceType
```

#### Description

Seismic source type.

#### Version

Available since version 2.5.

### II.1.5.10.2.9 ZoneType

#### C++

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_UBC_97_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_UBC_97_ZoneType);
```

#### C#

```
public IRobotSeismicAnalysis_UBC_97_ZoneType ZoneType { get; set; }
```

#### Visual Basic

```
Public ZoneType As IRobotSeismicAnalysis_UBC_97_ZoneType
```

#### Description

Seismic zone number.

#### Version

Available since version 2.5.

### II.1.5.11 IRobotSeismicAnalysis\_IBC\_2000\_Params

#### Class Hierarchy

**C++**

```
interface IRobotSeismicAnalysis_IBC_2000_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_IBC_2000_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_IBC_2000_Params
```

**Description**

Seismic analysis parameters for the IBC 2000 code issued in the USA. .

**Version**

Available since version 2.5.

**II.1.5.11.1 IRobotSeismicAnalysis\_IBC\_2000\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_IBC\_2000\_Params.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BehaviorFactor ( <a href="#">see page 243</a> )	Behaviour factor.
◆	Direction ( <a href="#">see page 244</a> )	Seismic excitation direction.
◆	ExcitationDir ( <a href="#">see page 244</a> )	
◆	Filter ( <a href="#">see page 244</a> )	Description of modes taken into account during structure dynamic analysis.
◆	Ie ( <a href="#">see page 245</a> )	Structure importance factor.
◆	S1 ( <a href="#">see page 245</a> )	Spectral acceleration for short periods.
◆	SiteClass ( <a href="#">see page 245</a> )	Soil.
◆	Ss ( <a href="#">see page 246</a> )	Spectral acceleration for 1-second period.

**II.1.5.11.2 IRobotSeismicAnalysis\_IBC\_2000\_Params Fields**

The fields of the IRobotSeismicAnalysis\_IBC\_2000\_Params class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BehaviorFactor ( <a href="#">see page 243</a> )	Behaviour factor.
◆	Direction ( <a href="#">see page 244</a> )	Seismic excitation direction.
◆	ExcitationDir ( <a href="#">see page 244</a> )	
◆	Filter ( <a href="#">see page 244</a> )	Description of modes taken into account during structure dynamic analysis.
◆	Ie ( <a href="#">see page 245</a> )	Structure importance factor.
◆	S1 ( <a href="#">see page 245</a> )	Spectral acceleration for short periods.
◆	SiteClass ( <a href="#">see page 245</a> )	Soil.
◆	Ss ( <a href="#">see page 246</a> )	Spectral acceleration for 1-second period.

**II.1.5.11.2.1 BehaviorFactor****C++**

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

**C#**

```
public double BehaviorFactor { get; set; }
```

**Visual Basic**

```
Public BehaviorFactor As Double
```

**Description**

Behaviour factor.

**Version**

Available since version 8.

## II.1.5.11.2.2 Direction

**C++**

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Description**

Seismic excitation direction.

**Version**

Available since version 2.5.

## II.1.5.11.2.3 ExcitationDir

**C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

## II.1.5.11.2.4 Filter

**C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during structure dynamic analysis.

**Version**

Available since version 2.5.

### II.1.5.11.2.5 Ie

#### C++

```
HRESULT get_Ie(double* );
HRESULT put_Ie(double );
```

#### C#

```
public double Ie { get; set; }
```

#### Visual Basic

```
Public Ie As Double
```

#### Description

Structure importance factor.

#### Version

Available since version 8.

### II.1.5.11.2.6 S1

#### C++

```
HRESULT get_S1(double* );
HRESULT put_S1(double );
```

#### C#

```
public double S1 { get; set; }
```

#### Visual Basic

```
Public S1 As Double
```

#### Description

Spectral acceleration for short periods.

#### Version

Available since version 2.5.

### II.1.5.11.2.7 SiteClass

#### C++

```
HRESULT get_SiteClass(IRobotSeismicAnalysis_IBC_2000_SiteClassType* );
HRESULT put_SiteClass(IRobotSeismicAnalysis_IBC_2000_SiteClassType );
```

#### C#

```
public IRobotSeismicAnalysis_IBC_2000_SiteClassType SiteClass { get; set; }
```

#### Visual Basic

```
Public SiteClass As IRobotSeismicAnalysis_IBC_2000_SiteClassType
```

#### Description

Soil.

#### Version

Available since version 2.5.

## II.1.5.11.2.8 Ss

### C++

```
HRESULT get_Ss(double* );
HRESULT put_Ss(double );
```

### C#

```
public double Ss { get; set; }
```

### Visual Basic

```
Public Ss As Double
```

### Description

Spectral acceleration for 1-second period.

### Version

Available since version 2.5.

## II.1.5.12 IRobotSeismicAnalysis\_EAK\_2000\_Params

### Class Hierarchy

### C++

```
interface IRobotSeismicAnalysis_EAK_2000_Params : IDispatch;
```

### C#

```
public interface IRobotSeismicAnalysis_EAK_2000_Params;
```

### Visual Basic

```
Public Interface IRobotSeismicAnalysis_EAK_2000_Params
```

### Description

Seismic analysis parameters for the Greek E.A.K. 2000 code. .

### Version

Available since version 2.5.

## II.1.5.12.1 IRobotSeismicAnalysis\_EAK\_2000\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_EAK\_2000\_Params.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 247</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 247</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 248</a> )	Direction ( <a href="#">see page 247</a> ).
◆	Filter ( <a href="#">see page 248</a> )	Description of modes taken into account during structure dynamic analysis.
◆	FoundationFactor ( <a href="#">see page 248</a> )	Foundation coefficient.
◆	GroundCategory ( <a href="#">see page 249</a> )	Ground category.
◆	ImportanceFactor ( <a href="#">see page 249</a> )	Structure importance.
◆	VerticalBehaviorFactor ( <a href="#">see page 249</a> )	Behavior coefficient for the vertical direction .
◆	VerticalFoundationFactor ( <a href="#">see page 249</a> )	Foundation coefficient for the vertical direction.
◆	ZoneType ( <a href="#">see page 250</a> )	Seismic zone number.

## II.1.5.12.2 IRobotSeismicAnalysis\_EAK\_2000\_Params Fields

The fields of the IRobotSeismicAnalysis\_EAK\_2000\_Params class are listed here.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 247</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 247</a> )	Seismic excitation direction.
◆	DirectionType ( <a href="#">see page 248</a> )	Direction ( <a href="#">see page 247</a> ).
◆	Filter ( <a href="#">see page 248</a> )	Description of modes taken into account during structure dynamic analysis.
◆	FoundationFactor ( <a href="#">see page 248</a> )	Foundation coefficient.
◆	GroundCategory ( <a href="#">see page 249</a> )	Ground category.
◆	ImportanceFactor ( <a href="#">see page 249</a> )	Structure importance.
◆	VerticalBehaviorFactor ( <a href="#">see page 249</a> )	Behavior coefficient for the vertical direction .
◆	VerticalFoundationFactor ( <a href="#">see page 249</a> )	Foundation coefficient for the vertical direction.
◆	ZoneType ( <a href="#">see page 250</a> )	Seismic zone number.

### II.1.5.12.2.1 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

#### Description

Behavior factor.

#### Version

Available since version 2.5.

### II.1.5.12.2.2 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 2.5.

### II.1.5.12.2.3 DirectionType

#### C++

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

#### C#

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

#### Visual Basic

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

#### Description

Direction (see page 247).

#### Version

Available since version 2.5.

### II.1.5.12.2.4 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during structure dynamic analysis.

#### Version

Available since version 2.5.

### II.1.5.12.2.5 FoundationFactor

#### C++

```
HRESULT get_FoundationFactor(double* );
HRESULT put_FoundationFactor(double);
```

#### C#

```
public double FoundationFactor { get; set; }
```

#### Visual Basic

```
Public FoundationFactor As Double
```

#### Description

Foundation coefficient.

#### Version

Available since version 2.5.

### II.1.5.12.2.6 GroundCategory

#### C++

```
HRESULT get_GroundCategory(IRobotSeismicAnalysis_EAK_2000_GroundCategoryType* );
HRESULT put_GroundCategory(IRobotSeismicAnalysis_EAK_2000_GroundCategoryType);
```

#### C#

```
public IRobotSeismicAnalysis_EAK_2000_GroundCategoryType GroundCategory { get; set; }
```

#### Visual Basic

```
Public GroundCategory As IRobotSeismicAnalysis_EAK_2000_GroundCategoryType
```

#### Description

Ground category.

#### Version

Available since version 2.5.

### II.1.5.12.2.7 ImportanceFactor

#### C++

```
HRESULT get_ImportanceFactor(IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType* );
HRESULT put_ImportanceFactor(IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType);
```

#### C#

```
public IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType ImportanceFactor { get; set; }
```

#### Visual Basic

```
Public ImportanceFactor As IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType
```

#### Description

Structure importance.

#### Version

Available since version 2.5.

### II.1.5.12.2.8 VerticalBehaviorFactor

#### C++

```
HRESULT get_VerticalBehaviorFactor(double* );
```

#### C#

```
public double VerticalBehaviorFactor { get; }
```

#### Visual Basic

```
Public ReadOnly VerticalBehaviorFactor As Double
```

#### Description

Behavior coefficient for the vertical direction .

#### Version

Available since version 2.5.

### II.1.5.12.2.9 VerticalFoundationFactor

#### C++

```
HRESULT get_VerticalFoundationFactor(double* );
```

**C#**

```
public double VerticalFoundationFactor { get; }
```

**Visual Basic**

```
Public ReadOnly VerticalFoundationFactor As Double
```

**Description**

Foundation coefficient for the vertical direction.

**Version**

Available since version 2.5.

**II.1.5.12.2.10 ZoneType****C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_EAK_2000_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_EAK_2000_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_EAK_2000_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_EAK_2000_ZoneType
```

**Description**

Seismic zone number.

**Version**

Available since version 2.5.

**II.1.5.13 IRobotSeismicAnalysis\_AFPS\_90\_ZoneType****C++**

```
enum IRobotSeismicAnalysis_AFPS_90_ZoneType;
```

**C#**

```
public enum IRobotSeismicAnalysis_AFPS_90_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_AFPS_90_ZoneType
```

**Members**

Members	Description
I_SAFT_AFPS_90_IA = 0	Available since version 2.5.
I_SAFT_AFPS_90_IB = 1	Available since version 2.5.
I_SAFT_AFPS_90_II = 2	Available since version 2.5.
I_SAFT_AFPS_90_III = 3	Available since version 2.5.

**Description**

Available numbers of seismic zones.

**Version**

Available since version 2.5.

## II.1.5.14 IRobotSeismicAnalysis\_CIRSOC\_103\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_CIRSOC_103_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_CIRSOC_103_ZoneType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_CIRSOC_103_ZoneType
```

### Members

Members	Description
I_SAQT_CIRSOC_103_0 = 0	Available since version 2.5.
I_SAQT_CIRSOC_103_1 = 1	Available since version 2.5.
I_SAQT_CIRSOC_103_2 = 2	Available since version 2.5.
I_SAQT_CIRSOC_103_3 = 3	Available since version 2.5.
I_SAQT_CIRSOC_103_4 = 4	Available since version 2.5.

### Description

Available numbers of seismic zones.

### Version

Available since version 2.5.

## II.1.5.15 IRobotSeismicAnalysis\_P\_100\_92\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_P_100_92_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_P_100_92_ZoneType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_P_100_92_ZoneType
```

### Members

Members	Description
I_SAQT_P_100_92_A = 0	Available since version 2.5.
I_SAQT_P_100_92_B = 1	Available since version 2.5.
I_SAQT_P_100_92_C = 2	Available since version 2.5.
I_SAQT_P_100_92_D = 3	Available since version 2.5.
I_SAQT_P_100_92_E = 4	Available since version 2.5.
I_SAQT_P_100_92_F = 5	Available since version 2.5.

### Description

Available numbers of seismic zones.

### Version

Available since version 2.5.

## II.1.5.16 IRobotSeismicAnalysis\_PS\_92\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_PS_92_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_PS_92_ZoneType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_PS_92_ZoneType
```

### Members

Members	Description
I_SAQT_PS_92_IA = 0	Available since version 2.5.
I_SAQT_PS_92_IB = 1	Available since version 2.5.
I_SAQT_PS_92_II = 2	Available since version 2.5.
I_SAQT_PS_92_III = 3	Available since version 2.5.

### Description

Available numbers of seismic zones.

### Version

Available since version 2.5.

## II.1.5.17 IRobotSeismicAnalysis\_RPA\_88\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_RPA_88_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_RPA_88_ZoneType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPA_88_ZoneType
```

### Members

Members	Description
I_SAQT_RPA_88_I = 0	Available since version 2.5.
I_SAQT_RPA_88_II = 1	Available since version 2.5.
I_SAQT_RPA_88_III = 2	Available since version 2.5.

### Description

Available numbers of seismic zones.

### Version

Available since version 2.5.

## II.1.5.18 IRobotSeismicAnalysis\_UBC\_97\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_UBC_97_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_UBC_97_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_UBC_97_ZoneType
```

**Members**

Members	Description
I_SAQT_UBC_97_1 = 0	Available since version 2.5.
I_SAQT_UBC_97_2A = 1	Available since version 2.5.
I_SAQT_UBC_97_2B = 2	Available since version 2.5.
I_SAQT_UBC_97_3 = 3	Available since version 2.5.
I_SAQT_UBC_97_4 = 4	Available since version 2.5.

**Description**

Available numbers of seismic zones.

**Version**

Available since version 2.5.

**II.1.5.19 IRobotSeismicAnalysis\_EAK\_2000\_ZoneType****C++**

```
enum IRobotSeismicAnalysis_EAK_2000_ZoneType;
```

**C#**

```
public enum IRobotSeismicAnalysis_EAK_2000_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_EAK_2000_ZoneType
```

**Members**

Members	Description
I_SAQT_EAK_2000_I = 0	Available since version 2.5.
I_SAQT_EAK_2000_II = 1	Available since version 2.5.
I_SAQT_EAK_2000_III = 2	Available since version 2.5.
I_SAQT_EAK_2000_IV = 3	Available since version 2.5.

**Description**

Available numbers of seismic zones.

**Version**

Available since version 2.5.

**II.1.5.20 IRobotSeismicAnalysis\_TURKISH\_23098\_ZoneType****C++**

```
enum IRobotSeismicAnalysis_TURKISH_23098_ZoneType;
```

**C#**

```
public enum IRobotSeismicAnalysis_TURKISH_23098_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_TURKISH_23098_ZoneType
```

## Members

Members	Description
I_SAFT_TURKISH_23098_1 = 0	Available since version 2.5.
I_SAFT_TURKISH_23098_2 = 1	Available since version 2.5.
I_SAFT_TURKISH_23098_3 = 2	Available since version 2.5.
I_SAFT_TURKISH_23098_4 = 3	Available since version 2.5.

## Description

Available numbers of seismic zones.

## Version

Available since version 2.5.

## II.1.5.21 IRobotSeismicAnalysis\_AFPS\_90\_StructureType

### C++

```
enum IRobotSeismicAnalysis_AFPS_90_StructureType;
```

### C#

```
public enum IRobotSeismicAnalysis_AFPS_90_StructureType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_AFPS_90_StructureType
```

## Members

Members	Description
I_SAST_AFPS_90_A = 0	Available since version 2.5.
I_SAST_AFPS_90_B = 1	Available since version 2.5.
I_SAST_AFPS_90_C = 2	Available since version 2.5.

## Description

Available structure types.

## Version

Available since version 2.5.

## II.1.5.22 IRobotSeismicAnalysis\_AFPS\_90\_SiteType

### C++

```
enum IRobotSeismicAnalysis_AFPS_90_SiteType;
```

### C#

```
public enum IRobotSeismicAnalysis_AFPS_90_SiteType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_AFPS_90_SiteType
```

## Members

Members	Description
I_SAST_AFPS_90_S0 = 0	Available since version 2.5.
I_SAST_AFPS_90_S1 = 1	Available since version 2.5.
I_SAST_AFPS_90_S2 = 2	Available since version 2.5.
I_SAST_AFPS_90_S3 = 3	Available since version 2.5.

**Description**

Available types of structure site.

**Version**

Available since version 2.5.

**II.1.5.23 IRobotSeismicAnalysisSpectrumType****C++**

```
enum IRobotSeismicAnalysisSpectrumType;
```

**C#**

```
public enum IRobotSeismicAnalysisSpectrumType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysisSpectrumType
```

**Members**

Members	Description
I_SAST_DIMENSIONING = 0	Available since version 2.5.
I_SAST_ELASITC = 1	Available since version 2.5.

**Description**

Available spectrum types. .

**Version**

Available since version 2.5.

**II.1.5.24 IRobotSeismicAnalysis\_CHINESE\_StructureType****C++**

```
enum IRobotSeismicAnalysis_CHINESE_StructureType;
```

**C#**

```
public enum IRobotSeismicAnalysis_CHINESE_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_CHINESE_StructureType
```

**Members**

Members	Description
I_SAST_CHINESE_SPECIAL = 0	Available since version 2.5.
I_SAST_CHINESE_BUILDINGS = 1	Available since version 2.5.
I_SAST_CHINESE_BRIDGES = 2	Available since version 2.5.
I_SAST_CHINESE_HARBOR_BUILDINGS = 3	Available since version 2.5.

**Description**

Available structure types.

**Version**

Available since version 2.5.

## II.1.5.25 IRobotSeismicAnalysis\_CHINESE\_SiteType

### C++

```
enum IRobotSeismicAnalysis_CHINESE_SiteType;
```

### C#

```
public enum IRobotSeismicAnalysis_CHINESE_SiteType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_CHINESE_SiteType
```

### Members

Members	Description
I_SAST_CHINESE_I = 0	Available since version 2.5.
I_SAST_CHINESE_II = 1	Available since version 2.5.
I_SAST_CHINESE_III = 2	Available since version 2.5.
I_SAST_CHINESE_IV = 3	Available since version 2.5.

### Description

Available types of structure site.

### Version

Available since version 2.5.

## II.1.5.26 IRobotSeismicAnalysis\_CHINESE\_IntensityType

### C++

```
enum IRobotSeismicAnalysis_CHINESE_IntensityType;
```

### C#

```
public enum IRobotSeismicAnalysis_CHINESE_IntensityType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_CHINESE_IntensityType
```

### Members

Members	Description
I_SAINT_CHINESE_6 = 0	Available since version 2.5.
I_SAINT_CHINESE_7 = 1	Available since version 2.5.
I_SAINT_CHINESE_8 = 2	Available since version 2.5.
I_SAINT_CHINESE_9 = 3	Available since version 2.5.

### Description

Available intensity types.

### Version

Available since version 2.5.

## II.1.5.27 IRobotSeismicAnalysis\_CHINESE\_DesignType

### C++

```
enum IRobotSeismicAnalysis_CHINESE_DesignType;
```

**C#**

```
public enum IRobotSeismicAnalysis_CHINESE_DesignType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_CHINESE_DesignType
```

**Members**

Members	Description
I_SADT_CHINESE_A = 0	Available since version 2.5.
I_SADT_CHINESE_B = 1	Available since version 2.5.

**Description**

Available design standards.

**Version**

Available since version 2.5.

**II.1.5.28 IRobotSeismicAnalysis\_CHINESE\_EarthquakeType****C++**

```
enum IRobotSeismicAnalysis_CHINESE_EarthquakeType;
```

**C#**

```
public enum IRobotSeismicAnalysis_CHINESE_EarthquakeType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_CHINESE_EarthquakeType
```

**Members**

Members	Description
I_SAET_CHINESE_NEAR = 0	Available since version 2.5.
I_SAET_CHINESE_FAR = 1	Available since version 2.5.

**Description**

Available types of earthquake distance.

**Version**

Available since version 2.5.

**II.1.5.29 IRobotSeismicAnalysis\_CIRSOC\_103\_SoilType****C++**

```
enum IRobotSeismicAnalysis_CIRSOC_103_SoilType;
```

**C#**

```
public enum IRobotSeismicAnalysis_CIRSOC_103_SoilType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_CIRSOC_103_SoilType
```

**Members**

Members	Description
I_SAST_CIRSOC_103_I = 0	Available since version 2.5.

I_SAST_CIRSOC_103_II = 1	Available since version 2.5.
I_SAST_CIRSOC_103_III = 2	Available since version 2.5.

**Description**

Available ground types. .

**Version**

Available since version 2.5.

**II.1.5.30 IRobotSeismicAnalysis\_CIRSOC\_103\_StructureType****C++**

```
enum IRobotSeismicAnalysis_CIRSOC_103_StructureType;
```

**C#**

```
public enum IRobotSeismicAnalysis_CIRSOC_103_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_CIRSOC_103_StructureType
```

**Members**

Members	Description
I_SAST_CIRSOC_103_A0 = 0	Available since version 2.5.
I_SAST_CIRSOC_103_A = 1	Available since version 2.5.
I_SAST_CIRSOC_103_B = 2	Available since version 2.5.

**Description**

Available structure types.

**Version**

Available since version 2.5.

**II.1.5.31 IRobotSeismicAnalysisDirectionType****C++**

```
enum IRobotSeismicAnalysisDirectionType;
```

**C#**

```
public enum IRobotSeismicAnalysisDirectionType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysisDirectionType
```

**Members**

Members	Description
I_SADT_HORIZONTAL = 0	Available since version 2.5.
I_SADT_VERTICAL = 1	Available since version 2.5.

**Description**

Available direction types.

**Version**

Available since version 2.5.

### II.1.5.32 IRobotSeismicAnalysis\_DM\_16\_1\_96\_ProtectionCoeffType

#### C++

```
enum IRobotSeismicAnalysis_DM_16_1_96_ProtectionCoeffType;
```

#### C#

```
public enum IRobotSeismicAnalysis_DM_16_1_96_ProtectionCoeffType;
```

#### Visual Basic

```
Public Enum IRobotSeismicAnalysis_DM_16_1_96_ProtectionCoeffType
```

#### Members

Members	Description
I_SAPCT_DM_16_1_96_1 = 0	Available since version 2.5.
I_SAPCT_DM_16_1_96_12 = 1	Available since version 2.5.
I_SAPCT_DM_16_1_96_14 = 2	Available since version 2.5.

#### Description

Available values of seismic protection coefficients.. .

#### Version

Available since version 2.5.

### II.1.5.33 IRobotSeismicAnalysis\_P\_100\_92\_ImportanceClassType

#### C++

```
enum IRobotSeismicAnalysis_P_100_92_ImportanceClassType;
```

#### C#

```
public enum IRobotSeismicAnalysis_P_100_92_ImportanceClassType;
```

#### Visual Basic

```
Public Enum IRobotSeismicAnalysis_P_100_92_ImportanceClassType
```

#### Members

Members	Description
I_SAICT_100_92_I = 0	Available since version 2.5.
I_SAICT_100_92_II = 1	Available since version 2.5.
I_SAICT_100_92_III = 2	Available since version 2.5.
I_SAICT_100_92_IV = 3	Available since version 2.5.

#### Description

Available importance classes.. .

#### Version

Available since version 2.5.

### II.1.5.34 IRobotSeismicAnalysis\_PS\_69\_DampingType

#### C++

```
enum IRobotSeismicAnalysis_PS_69_DampingType;
```

#### C#

```
public enum IRobotSeismicAnalysis_PS_69_DampingType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_PS_69_DampingType
```

**Members**

Members	Description
I_SADT_PS_69_NORMAL = 0	Available since version 2.5.
I_SADT_PS_69_AVERAGE = 1	Available since version 2.5.
I_SADT_PS_69_WEAK = 2	Available since version 2.5.

**Description**

Available damping types. .

**Version**

Available since version 2.5.

**II.1.5.35 IRobotSeismicAnalysis\_PS\_69\_SoilType****C++**

```
enum IRobotSeismicAnalysis_PS_69_SoilType;
```

**C#**

```
public enum IRobotSeismicAnalysis_PS_69_SoilType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_PS_69_SoilType
```

**Members**

Members	Description
I_SAST_PS_69_RIGID = 0	Available since version 2.5.
I_SAST_PS_69_FLEXIBLE = 1	Available since version 2.5.

**Description**

Available ground types. .

**Version**

Available since version 2.5.

**II.1.5.36 IRobotSeismicAnalysis\_PS\_92\_StructureType****C++**

```
enum IRobotSeismicAnalysis_PS_92_StructureType;
```

**C#**

```
public enum IRobotSeismicAnalysis_PS_92_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_PS_92_StructureType
```

**Members**

Members	Description
I_SAST_PS_92_B = 0	Available since version 2.5.
I_SAST_PS_92_C = 1	Available since version 2.5.
I_SAST_PS_92_D = 2	Available since version 2.5.

**Description**

Available structure types.

**Version**

Available since version 2.5.

**II.1.5.37 IRobotSeismicAnalysis\_PS\_92\_SiteType****C++**

```
enum IRobotSeismicAnalysis_PS_92_SiteType;
```

**C#**

```
public enum IRobotSeismicAnalysis_PS_92_SiteType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_PS_92_SiteType
```

**Members**

Members	Description
I_SAST_PS_92_S0 = 0	Available since version 2.5.
I_SAST_PS_92_S1 = 1	Available since version 2.5.
I_SAST_PS_92_S2 = 2	Available since version 2.5.
I_SAST_PS_92_S3 = 3	Available since version 2.5.
I_SAST_PS_92_ENVELOPE = 4	Available since version 5.5.

**Description**

Available site types.

**Version**

Available since version 2.5.

**II.1.5.38 IRobotSeismicAnalysis\_RPA\_88\_UsageType****C++**

```
enum IRobotSeismicAnalysis_RPA_88_UsageType;
```

**C#**

```
public enum IRobotSeismicAnalysis_RPA_88_UsageType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_RPA_88_UsageType
```

**Members**

Members	Description
I_SAUT_RPA_88_1 = 0	Available since version 2.5.
I_SAUT_RPA_88_2 = 1	Available since version 2.5.
I_SAUT_RPA_88_3 = 2	Available since version 2.5.

**Description**

Available usage types.

**Version**

Available since version 2.5.

## II.1.5.39 IRobotSeismicAnalysis\_RPA\_88\_CategoryType

### C++

```
enum IRobotSeismicAnalysis_RPA_88_CategoryType;
```

### C#

```
public enum IRobotSeismicAnalysis_RPA_88_CategoryType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPA_88_CategoryType
```

### Members

Members	Description
I_SACT_RPA_88_1 = 0	Available since version 2.5.
I_SACT_RPA_88_2 = 1	Available since version 2.5.
I_SACT_RPA_88_3 = 2	Available since version 2.5.
I_SACT_RPA_88_4 = 3	Available since version 2.5.
I_SACT_RPA_88_5 = 4	Available since version 2.5.
I_SACT_RPA_88_6 = 5	Available since version 2.5.
I_SACT_RPA_88_7 = 6	Available since version 2.5.
I_SACT_RPA_88_8 = 7	Available since version 2.5.

### Description

Available categories.

### Version

Available since version 2.5.

## II.1.5.40 IRobotSeismicAnalysis\_RPA\_88\_SoilType

### C++

```
enum IRobotSeismicAnalysis_RPA_88_SoilType;
```

### C#

```
public enum IRobotSeismicAnalysis_RPA_88_SoilType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPA_88_SoilType
```

### Members

Members	Description
I_SAST_RPA_88_RIGID = 0	Available since version 2.5.
I_SAST_RPA_88_FLEXIBLE = 1	Available since version 2.5.

### Description

Available soil types.

### Version

Available since version 2.5.

## II.1.5.41 IRobotSeismicAnalysis\_TURKISH\_23098\_SoilType

### C++

```
enum IRobotSeismicAnalysis_TURKISH_23098_SoilType;
```

### C#

```
public enum IRobotSeismicAnalysis_TURKISH_23098_SoilType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_TURKISH_23098_SoilType
```

### Members

Members	Description
I_SAST_TURKISH_23098_Z1 = 0	Available since version 2.5.
I_SAST_TURKISH_23098_Z2 = 1	Available since version 2.5.
I_SAST_TURKISH_23098_Z3 = 2	Available since version 2.5.
I_SAST_TURKISH_23098_Z4 = 3	Available since version 2.5.

### Description

Available soil types.

### Version

Available since version 2.5.

## II.1.5.42 IRobotSeismicAnalysis\_UBC\_97\_SoilType

### C++

```
enum IRobotSeismicAnalysis_UBC_97_SoilType;
```

### C#

```
public enum IRobotSeismicAnalysis_UBC_97_SoilType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_UBC_97_SoilType
```

### Members

Members	Description
I_SAST_UBC_97_Sa = 0	Available since version 2.5.
I_SAST_UBC_97_Sb = 1	Available since version 2.5.
I_SAST_UBC_97_Sc = 2	Available since version 2.5.
I_SAST_UBC_97_Sd = 3	Available since version 2.5.
I_SAST_UBC_97_Se = 4	Available since version 2.5.
I_SAST_UBC_97_Sf = 5	Available since version 2.5.

### Description

Available soil types.

### Version

Available since version 2.5.

## II.1.5.43 IRobotSeismicAnalysis\_UBC\_97\_SourceType

### C++

```
enum IRobotSeismicAnalysis_UBC_97_SourceType;
```

### C#

```
public enum IRobotSeismicAnalysis_UBC_97_SourceType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_UBC_97_SourceType
```

### Members

Members	Description
I_SAST_UBC_97_A = 0	Available since version 2.5.
I_SAST_UBC_97_B = 1	Available since version 2.5.
I_SAST_UBC_97_C = 2	Available since version 2.5.

### Description

Available seismic source types.

### Version

Available since version 2.5.

## II.1.5.44 IRobotSeismicAnalysis\_IBC\_2000\_SiteClassType

### C++

```
enum IRobotSeismicAnalysis_IBC_2000_SiteClassType;
```

### C#

```
public enum IRobotSeismicAnalysis_IBC_2000_SiteClassType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_IBC_2000_SiteClassType
```

### Members

Members	Description
I_SASCT_IBC_2000_A = 0	Available since version 2.5.
I_SASCT_IBC_2000_B = 1	Available since version 2.5.
I_SASCT_IBC_2000_C = 2	Available since version 2.5.
I_SASCT_IBC_2000_D = 3	Available since version 2.5.
I_SASCT_IBC_2000_E = 4	Available since version 2.5.
I_SASCT_IBC_2000_F = 5	Available since version 2.5.

### Description

Available soil types.

### Version

Available since version 2.5.

## II.1.5.45 IRobotSeismicAnalysis\_EAK\_2000\_ImportanceFactorType

### C++

```
enum IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType;
```

**C#**

```
public enum IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType
```

**Members**

Members	Description
I_SAIFT_EAK_2000_SIGMA1 = 0	Available since version 2.5.
I_SAIFT_EAK_2000_SIGMA2 = 1	Available since version 2.5.
I_SAIFT_EAK_2000_SIGMA3 = 2	Available since version 2.5.
I_SAIFT_EAK_2000_SIGMA4 = 3	Available since version 2.5.

**Description**

Available importance factors. .

**Version**

Available since version 2.5.

**II.1.5.46 IRobotSeismicAnalysis\_EAK\_2000\_GroundCategoryType****C++**

```
enum IRobotSeismicAnalysis_EAK_2000_GroundCategoryType;
```

**C#**

```
public enum IRobotSeismicAnalysis_EAK_2000_GroundCategoryType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_EAK_2000_GroundCategoryType
```

**Members**

Members	Description
I_SAGCT_EAK_2000_ALPHA = 0	Available since version 2.5.
I_SAGCT_EAK_2000_BETA = 1	Available since version 2.5.
I_SAGCT_EAK_2000_GAMMA = 2	Available since version 2.5.
I_SAGCT_EAK_2000_DELTA = 3	Available since version 2.5.

**Description**

Available ground categories.

**Version**

Available since version 2.5.

**II.1.5.47 IRobotSeismicAnalysis\_PS\_92\_SiteEnvelope****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_PS_92_SiteEnvelope : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_PS_92_SiteEnvelope;
```

## Visual Basic

```
Public Interface IRobotSeismicAnalysis_PS_92_SiteEnvelope
```

### Version

Available since version 5.5.

#### II.1.5.47.1 IRobotSeismicAnalysis\_PS\_92\_SiteEnvelope Members

The following tables list the members exposed by IRobotSeismicAnalysis\_PS\_92\_SiteEnvelope.

### Public Methods

	Name	Description
💡	IsActive (🔗 see page 266)	Function returns the TRUE value if the indicated site type has been considered in the envelope. .
💡	SetActive (🔗 see page 266)	Function adds/deletes the indicated site type to/from the envelope. .

#### II.1.5.47.2 IRobotSeismicAnalysis\_PS\_92\_SiteEnvelope Methods

The methods of the IRobotSeismicAnalysis\_PS\_92\_SiteEnvelope class are listed here.

### Public Methods

	Name	Description
💡	IsActive (🔗 see page 266)	Function returns the TRUE value if the indicated site type has been considered in the envelope. .
💡	SetActive (🔗 see page 266)	Function adds/deletes the indicated site type to/from the envelope. .

#### II.1.5.47.2.1 IsActive

##### C++

```
HRESULT IsActive(IRobotSeismicAnalysis_PS_92_SiteType _site, VARIANT_BOOL* ret);
```

##### C#

```
public bool IsActive(IRobotSeismicAnalysis_PS_92_SiteType _site);
```

## Visual Basic

```
Public Function IsActive(_site As IRobotSeismicAnalysis_PS_92_SiteType) As Boolean
```

### Description

Function returns the TRUE value if the indicated site type has been considered in the envelope. .

#### II.1.5.47.2.2 SetActive

##### C++

```
HRESULT SetActive(IRobotSeismicAnalysis_PS_92_SiteType _site, VARIANT_BOOL _is_active = -1);
```

##### C#

```
public void SetActive(IRobotSeismicAnalysis_PS_92_SiteType _site, bool _is_active = -1);
```

## Visual Basic

```
Public Sub SetActive(_site As IRobotSeismicAnalysis_PS_92_SiteType, Optional _is_active As Boolean = -1)
```

### Description

Function adds/deletes the indicated site type to/from the envelope. .

## II.1.5.48 IRobotSeismicAnalysis\_RPS\_2000\_ZoneType

C++

```
enum IRobotSeismicAnalysis_RPS_2000_ZoneType;
```

C#

```
public enum IRobotSeismicAnalysis_RPS_2000_ZoneType;
```

Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPS_2000_ZoneType
```

## II.1.5.49 IRobotSeismicAnalysis\_RPS\_2000\_SiteType

C++

```
enum IRobotSeismicAnalysis_RPS_2000_SiteType;
```

C#

```
public enum IRobotSeismicAnalysis_RPS_2000_SiteType;
```

Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPS_2000_SiteType
```

## II.1.5.50 IRobotSeismicAnalysis\_RPS\_2000\_StructureClass

C++

```
enum IRobotSeismicAnalysis_RPS_2000_StructureClass;
```

C#

```
public enum IRobotSeismicAnalysis_RPS_2000_StructureClass;
```

Visual Basic

```
Public Enum IRobotSeismicAnalysis_RPS_2000_StructureClass
```

## II.1.5.51 IRobotSeismicAnalysis\_RPS\_2000\_Params

Class Hierarchy

C++

```
interface IRobotSeismicAnalysis_RPS_2000_Params : IDispatch;
```

C#

```
public interface IRobotSeismicAnalysis_RPS_2000_Params;
```

Visual Basic

```
Public Interface IRobotSeismicAnalysis_RPS_2000_Params
```

Description

Parameters of the Moroccan seismic code R.P.S. 2000.

### II.1.5.51.1 IRobotSeismicAnalysis\_RPS\_2000\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_RPS\_2000\_Params.

## Public Fields

	Name	Description
◆	BehaviorFactor (see page 268)	
◆	DirectionType (see page 268)	
◆	ExcitationDir (see page 268)	
◆	Filter (see page 269)	
◆	ResidualMode (see page 269)	
◆	Site (see page 269)	
◆	StructureClass (see page 269)	
◆	Zone (see page 270)	

### II.1.5.51.2 IRobotSeismicAnalysis\_RPS\_2000\_Params Fields

The fields of the IRobotSeismicAnalysis\_RPS\_2000\_Params class are listed here.

## Public Fields

	Name	Description
◆	BehaviorFactor (see page 268)	
◆	DirectionType (see page 268)	
◆	ExcitationDir (see page 268)	
◆	Filter (see page 269)	
◆	ResidualMode (see page 269)	
◆	Site (see page 269)	
◆	StructureClass (see page 269)	
◆	Zone (see page 270)	

### II.1.5.51.2.1 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

### II.1.5.51.2.2 DirectionType

#### C++

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

#### C#

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

#### Visual Basic

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

### II.1.5.51.2.3 ExcitationDir

#### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**II.1.5.51.2.4 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);  
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**II.1.5.51.2.5 ResidualMode****C++**

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

**C#**

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

**Visual Basic**

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

**Version**

Available since version 14.7.

**II.1.5.51.2.6 Site****C++**

```
HRESULT get_Site(IRobotSeismicAnalysis_RPS_2000_SiteType*);  
HRESULT put_Site(IRobotSeismicAnalysis_RPS_2000_SiteType*);
```

**C#**

```
public IRobotSeismicAnalysis_RPS_2000_SiteType Site { get; set; }
```

**Visual Basic**

```
Public Site As IRobotSeismicAnalysis_RPS_2000_SiteType
```

**II.1.5.51.2.7 StructureClass****C++**

```
HRESULT get_StructureClass(IRobotSeismicAnalysis_RPS_2000_StructureClass*);  
HRESULT put_StructureClass(IRobotSeismicAnalysis_RPS_2000_StructureClass*);
```

**C#**

```
public IRobotSeismicAnalysis_RPS_2000_StructureClass StructureClass { get; set; }
```

**Visual Basic**

```
Public StructureClass As IRobotSeismicAnalysis_RPS_2000_StructureClass
```

### II.1.5.51.2.8 Zone

**C++**

```
HRESULT get_Zone(IRobotSeismicAnalysis_RPS_2000_ZoneType* );
HRESULT put_Zone(IRobotSeismicAnalysis_RPS_2000_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_RPS_2000_ZoneType Zone { get; set; }
```

**Visual Basic**

```
Public Zone As IRobotSeismicAnalysis_RPS_2000_ZoneType
```

### II.1.5.52 IRobotSeismicAnalysis\_RPA\_2003\_ZoneType

**C++**

```
enum IRobotSeismicAnalysis_RPA_2003_ZoneType;
```

**C#**

```
public enum IRobotSeismicAnalysis_RPA_2003_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_RPA_2003_ZoneType
```

### II.1.5.53 IRobotSeismicAnalysis\_RPA\_2003\_UsageType

**C++**

```
enum IRobotSeismicAnalysis_RPA_2003_UsageType;
```

**C#**

```
public enum IRobotSeismicAnalysis_RPA_2003_UsageType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_RPA_2003_UsageType
```

### II.1.5.54 IRobotSeismicAnalysis\_RPA\_2003\_SiteType

**C++**

```
enum IRobotSeismicAnalysis_RPA_2003_SiteType;
```

**C#**

```
public enum IRobotSeismicAnalysis_RPA_2003_SiteType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_RPA_2003_SiteType
```

### II.1.5.55 IRobotSeismicAnalysis\_RPA\_2003\_Params

**Class Hierarchy**

**C++**

```
interface IRobotSeismicAnalysis_RPA_2003_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_RPA_2003_Params;
```

## Visual Basic

```
Public Interface IRobotSeismicAnalysis_RPA_2003_Params
```

### Description

Parameters of the Algerian seismic code RPA 99 (2003). .

#### II.1.5.55.1 IRobotSeismicAnalysis\_RPA\_2003\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_RPA\_2003\_Params.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 271</a> )	
◆	ExcitationDir ( <a href="#">see page 271</a> )	
◆	Filter ( <a href="#">see page 272</a> )	
◆	QualityCoef ( <a href="#">see page 272</a> )	
◆	ResidualMode ( <a href="#">see page 272</a> )	
◆	Site ( <a href="#">see page 272</a> )	
◆	Usage ( <a href="#">see page 272</a> )	
◆	Zone ( <a href="#">see page 273</a> )	

#### II.1.5.55.2 IRobotSeismicAnalysis\_RPA\_2003\_Params Fields

The fields of the IRobotSeismicAnalysis\_RPA\_2003\_Params class are listed here.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 271</a> )	
◆	ExcitationDir ( <a href="#">see page 271</a> )	
◆	Filter ( <a href="#">see page 272</a> )	
◆	QualityCoef ( <a href="#">see page 272</a> )	
◆	ResidualMode ( <a href="#">see page 272</a> )	
◆	Site ( <a href="#">see page 272</a> )	
◆	Usage ( <a href="#">see page 272</a> )	
◆	Zone ( <a href="#">see page 273</a> )	

#### II.1.5.55.2.1 BehaviorFactor

##### C++

```
HRESULT get_BehaviorFactor(double*);  
HRESULT put_BehaviorFactor(double);
```

##### C#

```
public double BehaviorFactor { get; set; }
```

## Visual Basic

```
Public BehaviorFactor As Double
```

#### II.1.5.55.2.2 ExcitationDir

##### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

##### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**II.1.5.55.2.3 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**II.1.5.55.2.4 QualityCoef****C++**

```
HRESULT get_QualityCoef(double* );
HRESULT put_QualityCoef(double);
```

**C#**

```
public double QualityCoef { get; set; }
```

**Visual Basic**

```
Public QualityCoef As double
```

**II.1.5.55.2.5 ResidualMode****C++**

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

**C#**

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

**Visual Basic**

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

**Version**

Available since version 14.7.

**II.1.5.55.2.6 Site****C++**

```
HRESULT get_Site(IRobotSeismicAnalysis_RPA_2003_SiteType* );
HRESULT put_Site(IRobotSeismicAnalysis_RPA_2003_SiteType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_2003_SiteType Site { get; set; }
```

**Visual Basic**

```
Public Site As IRobotSeismicAnalysis_RPA_2003_SiteType
```

**II.1.5.55.2.7 Usage****C++**

```
HRESULT get_Usage(IRobotSeismicAnalysis_RPA_2003_UsageType* );
HRESULT put_Usage(IRobotSeismicAnalysis_RPA_2003_UsageType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_2003_UsageType Usage { get; set; }
```

**Visual Basic**

```
Public Usage As IRobotSeismicAnalysis_RPA_2003_UsageType
```

**II.1.5.55.2.8 Zone****C++**

```
HRESULT get_Zone(IRobotSeismicAnalysis_RPA_2003_ZoneType* );
HRESULT put_Zone(IRobotSeismicAnalysis_RPA_2003_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_RPA_2003_ZoneType Zone { get; set; }
```

**Visual Basic**

```
Public Zone As IRobotSeismicAnalysis_RPA_2003_ZoneType
```

**II.1.5.56 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_ITALY_ORDINANZA_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_ITALY_ORDINANZA_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_ITALY_ORDINANZA_Params
```

**Description**

Seismic analysis parameters for the Italian code Ordinanza 3274.

**II.1.5.56.1 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Params.

**Public Fields**

	Name	Description
❖	Direction ( [ see page 274)	
❖	ExcitationDir ( [ see page 274)	
❖	FactorQ ( [ see page 274)	
❖	Filter ( [ see page 274)	
❖	Soil ( [ see page 274)	
❖	Spectrum ( [ see page 275)	
❖	Zone ( [ see page 275)	

**II.1.5.56.2 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Params Fields**

The fields of the IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Params class are listed here.

**Public Fields**

	Name	Description
❖	Direction ( [ see page 274)	
❖	ExcitationDir ( [ see page 274)	

◆	FactorQ (see page 274)	
◆	Filter (see page 274)	
◆	Soil (see page 274)	
◆	Spectrum (see page 275)	
◆	Zone (see page 275)	

### II.1.5.56.2.1 Direction

**C++**

```
HRESULT get_Direction(IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction* );
HRESULT put_Direction(IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction);
```

**C#**

```
public IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction
```

### II.1.5.56.2.2 ExcitationDir

**C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

### II.1.5.56.2.3 FactorQ

**C++**

```
HRESULT get_FactorQ(double* );
HRESULT put_FactorQ(double);
```

**C#**

```
public double FactorQ { get; set; }
```

**Visual Basic**

```
Public FactorQ As Double
```

### II.1.5.56.2.4 Filter

**C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

### II.1.5.56.2.5 Soil

**C++**

```
HRESULT get_Soil(IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType* );
HRESULT put_Soil(IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType);
```

**C#**

```
public IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType Soil { get; set; }
```

**Visual Basic**

```
Public Soil As IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType
```

**II.1.5.6.2.6 Spectrum****C++**

```
HRESULT get_Spectrum(IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum* );
HRESULT put_Spectrum(IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum);
```

**C#**

```
public IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum Spectrum { get; set; }
```

**Visual Basic**

```
Public Spectrum As IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum
```

**II.1.5.6.2.7 Zone****C++**

```
HRESULT get_Zone(IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType* );
HRESULT put_Zone(IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType Zone { get; set; }
```

**Visual Basic**

```
Public Zone As IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType
```

**II.1.5.7 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_SoilType****C++**

```
enum IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType;
```

**C#**

```
public enum IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType
```

**Description**

Soil type according to the Italian code Ordinanza 3274.

**II.1.5.8 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_ZoneType****C++**

```
enum IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType;
```

**C#**

```
public enum IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType
```

## II.1.5.59 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Spectrum

**C++**

```
enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum;
```

**C#**

```
public enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum
```

## II.1.5.60 IRobotSeismicAnalysis\_ITALY\_ORDINANZA\_Direction

**C++**

```
enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction;
```

**C#**

```
public enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction
```

## II.1.5.61 IRobotSeismicAnalysis\_P\_100\_2006\_Params

**Class Hierarchy**

**C++**

```
interface IRobotSeismicAnalysis_P_100_2006_Params : IDispatch,
```

**C#**

```
public interface IRobotSeismicAnalysis_P_100_2006_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_P_100_2006_Params
```

**Description**

Parameters of seismic analysis for Romanian code (P100-1-2006).

**Version**

Available since version 8.1.

### II.1.5.61.1 IRobotSeismicAnalysis\_P\_100\_2006\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_P\_100\_2006\_Params.

**Public Fields**

	Name	Description
◆	Agg (see page 277)	
◆	B0 (see page 277)	
◆	BehaviorFactor (see page 277)	
◆	ExcitationDir (see page 278)	
◆	ImportanceFactor (see page 278)	
◆	SpectrumType (see page 278)	
◆	Tb (see page 278)	

◆	Tc (see page 279)	
◆	Td (see page 279)	

### II.1.5.61.2 IRobotSeismicAnalysis\_P\_100\_2006\_Params Fields

The fields of the IRobotSeismicAnalysis\_P\_100\_2006\_Params class are listed here.

#### Public Fields

	Name	Description
◆	Agg (see page 277)	
◆	B0 (see page 277)	
◆	BehaviorFactor (see page 277)	
◆	ExcitationDir (see page 278)	
◆	ImportanceFactor (see page 278)	
◆	SpectrumType (see page 278)	
◆	Tb (see page 278)	
◆	Tc (see page 279)	
◆	Td (see page 279)	

### II.1.5.61.2.1 Agg

#### C++

```
HRESULT get_Agg(double* );
HRESULT put_Agg(double);
```

#### C#

```
public double Agg { get; set; }
```

#### Visual Basic

```
Public Agg As double
```

#### Version

Available since version 8.1.

### II.1.5.61.2.2 B0

#### C++

```
HRESULT get_B0(double* );
HRESULT put_B0(double);
```

#### C#

```
public double B0 { get; set; }
```

#### Visual Basic

```
Public B0 As double
```

#### Version

Available since version 8.1.

### II.1.5.61.2.3 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

**C#**

```
public double BehaviorFactor { get; set; }
```

**Visual Basic**

```
Public BehaviorFactor As Double
```

**Version**

Available since version 8.1.

**II.1.5.61.2.4 ExcitationDir****C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**Version**

Available since version 8.1.

**II.1.5.61.2.5 ImportanceFactor****C++**

```
HRESULT get_ImportanceFactor(double*);  
HRESULT put_ImportanceFactor(double);
```

**C#**

```
public double ImportanceFactor { get; set; }
```

**Visual Basic**

```
Public ImportanceFactor As Double
```

**Version**

Available since version 8.1.

**II.1.5.61.2.6 SpectrumType****C++**

```
HRESULT get_SpectrumType(IRobotSeismicAnalysisSpectrumType*);  
HRESULT put_SpectrumType(IRobotSeismicAnalysisSpectrumType);
```

**C#**

```
public IRobotSeismicAnalysisSpectrumType SpectrumType { get; set; }
```

**Visual Basic**

```
Public SpectrumType As IRobotSeismicAnalysisSpectrumType
```

**Version**

Available since version 8.1.

**II.1.5.61.2.7 Tb****C++**

```
HRESULT get_Tb(double*);  
HRESULT put_Tb(double);
```

**C#**

```
public double Tb { get; set; }
```

**Visual Basic**

```
Public Tb As Double
```

**Version**

Available since version 8.1.

**II.1.5.61.2.8 Tc****C++**

```
HRESULT get_Tc(double*);  
HRESULT put_Tc(double);
```

**C#**

```
public double Tc { get; set; }
```

**Visual Basic**

```
Public Tc As Double
```

**Version**

Available since version 8.1.

**II.1.5.61.2.9 Td****C++**

```
HRESULT get_Td(double*);  
HRESULT put_Td(double);
```

**C#**

```
public double Td { get; set; }
```

**Visual Basic**

```
Public Td As Double
```

**Version**

Available since version 8.1.

**II.1.5.62 IRobotSeismicAnalysis\_IBC\_2006\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_IBC_2006_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_IBC_2006_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_IBC_2006_Params
```

**Description**

Seismic analysis parameters for IBC 2006 released in the USA.

**Version**

Available since version 8.1.

## II.1.5.62.1 IRobotSeismicAnalysis\_IBC\_2006\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_IBC\_2006\_Params.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 280</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 281</a> )	Seismic excitation direction.
◆	ExcitationDir ( <a href="#">see page 281</a> )	
◆	Filter ( <a href="#">see page 281</a> )	Description of modes taken into account during the structure dynamic analysis.
◆	I ( <a href="#">see page 281</a> )	Structure importance factor.
◆	S1 ( <a href="#">see page 282</a> )	Spectral acceleration for short periods.
◆	SiteClass ( <a href="#">see page 282</a> )	Soil.
◆	Ss ( <a href="#">see page 282</a> )	Spectral acceleration for the period 1s.
◆	TL ( <a href="#">see page 283</a> )	term period.

## II.1.5.62.2 IRobotSeismicAnalysis\_IBC\_2006\_Params Fields

The fields of the IRobotSeismicAnalysis\_IBC\_2006\_Params class are listed here.

### Public Fields

	Name	Description
◆	BehaviorFactor ( <a href="#">see page 280</a> )	Behavior factor.
◆	Direction ( <a href="#">see page 281</a> )	Seismic excitation direction.
◆	ExcitationDir ( <a href="#">see page 281</a> )	
◆	Filter ( <a href="#">see page 281</a> )	Description of modes taken into account during the structure dynamic analysis.
◆	I ( <a href="#">see page 281</a> )	Structure importance factor.
◆	S1 ( <a href="#">see page 282</a> )	Spectral acceleration for short periods.
◆	SiteClass ( <a href="#">see page 282</a> )	Soil.
◆	Ss ( <a href="#">see page 282</a> )	Spectral acceleration for the period 1s.
◆	TL ( <a href="#">see page 283</a> )	term period.

## II.1.5.62.2.1 BehaviorFactor

### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

### C#

```
public double BehaviorFactor { get; set; }
```

### Visual Basic

```
Public BehaviorFactor As Double
```

### Description

Behavior factor.

### Version

Available since version 8.1.

### II.1.5.62.2.2 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

#### Version

Available since version 8.1.

### II.1.5.62.2.3 ExcitationDir

#### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

#### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

#### Visual Basic

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

#### Version

Available since version 8.1.

### II.1.5.62.2.4 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during the structure dynamic analysis.

#### Version

Available since version 8.1.

### II.1.5.62.2.5 I

#### C++

```
HRESULT get_I(double* );
HRESULT put_I(double );
```

**C#**

```
public double I { get; set; }
```

**Visual Basic**

```
Public I As Double
```

**Description**

Structure importance factor.

**Version**

Available since version 8.1.

**II.1.5.62.2.6 S1****C++**

```
HRESULT get_S1(double* );
HRESULT put_S1(double);
```

**C#**

```
public double S1 { get; set; }
```

**Visual Basic**

```
Public S1 As Double
```

**Description**

Spectral acceleration for short periods.

**Version**

Available since version 8.1.

**II.1.5.62.2.7 SiteClass****C++**

```
HRESULT get_SiteClass(IRobotSeismicAnalysis_IBC_2006_SiteClassType* );
HRESULT put_SiteClass(IRobotSeismicAnalysis_IBC_2006_SiteClassType);
```

**C#**

```
public IRobotSeismicAnalysis_IBC_2006_SiteClassType SiteClass { get; set; }
```

**Visual Basic**

```
Public SiteClass As IRobotSeismicAnalysis_IBC_2006_SiteClassType
```

**Description**

Soil.

**Version**

Available since version 8.1.

**II.1.5.62.2.8 Ss****C++**

```
HRESULT get_Ss(double* );
HRESULT put_Ss(double);
```

**C#**

```
public double Ss { get; set; }
```

**Visual Basic**

```
Public Ss As double
```

**Description**

Spectral acceleration for the period 1s.

**Version**

Available since version 8.1.

**II.1.5.62.2.9 TL****C++**

```
HRESULT get_TL(double* );
HRESULT put_TL(double);
```

**C#**

```
public double TL { get; set; }
```

**Visual Basic**

```
Public TL As double
```

**Description**

term period.

**Version**

Available since version 8.1.

**II.1.5.63 IRobotSeismicAnalysis\_IBC\_2006\_SiteClassType****C++**

```
enum IRobotSeismicAnalysis_IBC_2006_SiteClassType;
```

**C#**

```
public enum IRobotSeismicAnalysis_IBC_2006_SiteClassType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_IBC_2006_SiteClassType
```

**Members**

Members	Description
I_SASCT_IBC_2006_A = 0	Available since version 8.1.
I_SASCT_IBC_2006_B = 1	Available since version 8.1.
I_SASCT_IBC_2006_C = 2	Available since version 8.1.
I_SASCT_IBC_2006_D = 3	Available since version 8.1.
I_SASCT_IBC_2006_E = 4	Available since version 8.1.
I_SASCT_IBC_2006_F = 5	Available since version 8.1.

**Description**

Available soil types.

**Version**

Available since version 8.1.

## II.1.5.64 IRobotSeismicAnalysis\_PS\_92\_2008\_Params

### Class Hierarchy

#### C++

```
interface IRobotSeismicAnalysis_PS_92_2008_Params : IDispatch;
```

#### C#

```
public interface IRobotSeismicAnalysis_PS_92_2008_Params;
```

#### Visual Basic

```
Public Interface IRobotSeismicAnalysis_PS_92_2008_Params
```

### Description

Seismic analysis parameters for the French code PS 92 - 2008.

### Version

Available since version 8.7.

## II.1.5.64.1 IRobotSeismicAnalysis\_PS\_92\_2008\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_PS\_92\_2008\_Params.

### Public Fields

	Name	Description
◆	Ag (see page 285)	Acceleration factor.
◆	BehaviorFactor (see page 285)	Behavior factor.
◆	Direction (see page 285)	Seismic excitation direction.
◆	DirectionType (see page 286)	Direction (see page 285).
◆	ExcitationDir (see page 286)	Definition of the excitation direction.
◆	Filter (see page 286)	Description of modes taken into account during the structure dynamic analysis.
◆	ResidualMode (see page 287)	
◆	Site (see page 287)	Site.
◆	SiteEnvelope (see page 287)	
◆	SpectrumType (see page 287)	Spectrum.
◆	StructureType (see page 288)	
◆	Topography (see page 288)	Topography.
◆	ZoneType (see page 288)	Seismic zone no.

## II.1.5.64.2 IRobotSeismicAnalysis\_PS\_92\_2008\_Params Fields

The fields of the IRobotSeismicAnalysis\_PS\_92\_2008\_Params class are listed here.

### Public Fields

	Name	Description
◆	Ag (see page 285)	Acceleration factor.
◆	BehaviorFactor (see page 285)	Behavior factor.
◆	Direction (see page 285)	Seismic excitation direction.
◆	DirectionType (see page 286)	Direction (see page 285).
◆	ExcitationDir (see page 286)	Definition of the excitation direction.
◆	Filter (see page 286)	Description of modes taken into account during the structure dynamic analysis.
◆	ResidualMode (see page 287)	

◆	Site (see page 287)	Site.
◆	SiteEnvelope (see page 287)	
◆	SpectrumType (see page 287)	Spectrum.
◆	StructureType (see page 288)	
◆	Topography (see page 288)	Topography.
◆	ZoneType (see page 288)	Seismic zone no.

### II.1.5.64.2.1 Ag

#### C++

```
HRESULT get_Ag( double* );
HRESULT put_Ag( double );
```

#### C#

```
public double Ag { get; set; }
```

#### Visual Basic

```
Public Ag As Double
```

#### Description

Acceleration factor.

#### Version

Available since version 9.

### II.1.5.64.2.2 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double );
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

#### Description

Behavior factor.

#### Version

Available since version 8.7.

### II.1.5.64.2.3 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Seismic excitation direction.

**Version**

Available since version 8.7.

**II.1.5.64.2.4 DirectionType****C++**

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

**C#**

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

**Visual Basic**

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

**Description**

Direction (see page 285).

**Version**

Available since version 8.7.

**II.1.5.64.2.5 ExcitationDir****C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**Description**

Definition of the excitation direction.

**Version**

Available since version 8.7.

**II.1.5.64.2.6 Filter****C++**

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

**C#**

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

**Visual Basic**

```
Public Filter As IRobotCaseAnalysisModesFilter
```

**Description**

Description of modes taken into account during the structure dynamic analysis.

**Version**

Available since version 8.7.

### II.1.5.64.2.7 ResidualMode

#### C++

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

#### C#

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

#### Visual Basic

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

#### Version

Available since version 14.7.

### II.1.5.64.2.8 Site

#### C++

```
HRESULT get_Site(IRobotSeismicAnalysis_PS_92_2008_SiteType*);  
HRESULT put_Site(IRobotSeismicAnalysis_PS_92_2008_SiteType);
```

#### C#

```
public IRobotSeismicAnalysis_PS_92_2008_SiteType Site { get; set; }
```

#### Visual Basic

```
Public Site As IRobotSeismicAnalysis_PS_92_2008_SiteType
```

#### Description

Site.

#### Version

Available since version 8.7.

### II.1.5.64.2.9 SiteEnvelope

#### C++

```
HRESULT get_SiteEnvelope(IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope**);
```

#### C#

```
public IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope SiteEnvelope { get; }
```

#### Visual Basic

```
Public ReadOnly SiteEnvelope As IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope
```

#### Version

Available since version 8.7.

### II.1.5.64.2.10 SpectrumType

#### C++

```
HRESULT get_SpectrumType(IRobotSeismicAnalysisSpectrumType*);  
HRESULT put_SpectrumType(IRobotSeismicAnalysisSpectrumType);
```

#### C#

```
public IRobotSeismicAnalysisSpectrumType SpectrumType { get; set; }
```

#### Visual Basic

```
Public SpectrumType As IRobotSeismicAnalysisSpectrumType
```

**Description**

Spectrum.

**Version**

Available since version 8.7.

**II.1.5.64.2.11 StructureType****C++**

```
HRESULT get_StructureType(IRobotSeismicAnalysis_PS_92_2008_StructureType* );
HRESULT put_StructureType(IRobotSeismicAnalysis_PS_92_2008_StructureType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_92_2008_StructureType StructureType { get; set; }
```

**Visual Basic**

```
Public StructureType As IRobotSeismicAnalysis_PS_92_2008_StructureType
```

**Version**

Available since version 8.7.

**II.1.5.64.2.12 Topography****C++**

```
HRESULT get_Topography(double* );
HRESULT put_Topography(double);
```

**C#**

```
public double Topography { get; set; }
```

**Visual Basic**

```
Public Topography As Double
```

**Description**

Topography.

**Version**

Available since version 8.7.

**II.1.5.64.2.13 ZoneType****C++**

```
HRESULT get_ZoneType(IRobotSeismicAnalysis_PS_92_2008_ZoneType* );
HRESULT put_ZoneType(IRobotSeismicAnalysis_PS_92_2008_ZoneType);
```

**C#**

```
public IRobotSeismicAnalysis_PS_92_2008_ZoneType ZoneType { get; set; }
```

**Visual Basic**

```
Public ZoneType As IRobotSeismicAnalysis_PS_92_2008_ZoneType
```

**Description**

Seismic zone no.

**Version**

Available since version 8.7.

## II.1.5.65 IRobotSeismicAnalysis\_PS\_92\_2008\_ZoneType

### C++

```
enum IRobotSeismicAnalysis_PS_92_2008_ZoneType;
```

### C#

```
public enum IRobotSeismicAnalysis_PS_92_2008_ZoneType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_PS_92_2008_ZoneType
```

### Members

Members	Description
I_SAFT_PS_92_2008_2 = 0	Available since version 8.7.
I_SAFT_PS_92_2008_3 = 1	Available since version 8.7.
I_SAFT_PS_92_2008_4 = 2	Available since version 8.7.
I_SAFT_PS_92_2008_5 = 4	Available since version 8.7.

### Description

Available numbers of seismic zones.

### Version

Available since version 8.7.

## II.1.5.66 IRobotSeismicAnalysis\_PS\_92\_2008\_StructureType

### C++

```
enum IRobotSeismicAnalysis_PS_92_2008_StructureType;
```

### C#

```
public enum IRobotSeismicAnalysis_PS_92_2008_StructureType;
```

### Visual Basic

```
Public Enum IRobotSeismicAnalysis_PS_92_2008_StructureType
```

### Members

Members	Description
I_SAST_PS_92_2008_B = 0	Available since version 8.7.
I_SAST_PS_92_2008_C = 1	Available since version 8.7.
I_SAST_PS_92_2008_D = 2	Available since version 8.7.

### Description

Available structure types.

### Version

Available since version 8.7.

## II.1.5.67 IRobotSeismicAnalysis\_PS\_92\_2008\_SiteType

### C++

```
enum IRobotSeismicAnalysis_PS_92_2008_SiteType;
```

### C#

```
public enum IRobotSeismicAnalysis_PS_92_2008_SiteType;
```

## Visual Basic

```
Public Enum IRobotSeismicAnalysis_PS_92_2008_SiteType
```

## Members

Members	Description
I_SAST_PS_92_2008_S0 = 0	Available since version 8.7.
I_SAST_PS_92_2008_S1 = 1	Available since version 8.7.
I_SAST_PS_92_2008_S2 = 2	Available since version 8.7.
I_SAST_PS_92_2008_S3 = 3	Available since version 8.7.
I_SAST_PS_92_2008_ENVELOPE = 4	Available since version 8.7.

## Description

Available site types.

## Version

Available since version 8.7.

## II.1.5.68 IRobotSeismicAnalysis\_PS\_92\_2008\_SiteEnvelope

### Class Hierarchy

#### C++

```
interface IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope : IDispatch;
```

#### C#

```
public interface IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope;
```

## Visual Basic

```
Public Interface IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope
```

## Version

Available since version 8.7.

## II.1.5.68.1 IRobotSeismicAnalysis\_PS\_92\_2008\_SiteEnvelope Members

The following tables list the members exposed by IRobotSeismicAnalysis\_PS\_92\_2008\_SiteEnvelope.

### Public Methods

	Name	Description
≡	IsActive ( [ see page 291])	Function returns the TRUE value if the indicated site type has been included in the envelope.
≡	SetActive ( [ see page 291])	Function adds/deletes an indicated site type to/from the envelope.

## II.1.5.68.2 IRobotSeismicAnalysis\_PS\_92\_2008\_SiteEnvelope Methods

The methods of the IRobotSeismicAnalysis\_PS\_92\_2008\_SiteEnvelope class are listed here.

### Public Methods

	Name	Description
≡	IsActive ( [ see page 291])	Function returns the TRUE value if the indicated site type has been included in the envelope.
≡	SetActive ( [ see page 291])	Function adds/deletes an indicated site type to/from the envelope.

### II.1.5.68.2.1 IsActive

**C++**

```
HRESULT IsActive(IRobotSeismicAnalysis_PS_92_2008_SiteType _site, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsActive(IRobotSeismicAnalysis_PS_92_2008_SiteType _site);
```

**Visual Basic**

```
Public Function IsActive(_site As IRobotSeismicAnalysis_PS_92_2008_SiteType) As Boolean
```

#### Description

Function returns the TRUE value if the indicated site type has been included in the envelope.

#### Version

Available since version 8.7.

### II.1.5.68.2.2 SetActive

**C++**

```
HRESULT SetActive(IRobotSeismicAnalysis_PS_92_2008_SiteType _site, VARIANT_BOOL _is_active = -1);
```

**C#**

```
public void SetActive(IRobotSeismicAnalysis_PS_92_2008_SiteType _site, bool _is_active = -1);
```

**Visual Basic**

```
Public Sub SetActive(_site As IRobotSeismicAnalysis_PS_92_2008_SiteType, Optional _is_active As Boolean = -1)
```

#### Description

Function adds/deletes an indicated site type to/from the envelope.

#### Version

Available since version 8.7.

### II.1.5.69 IRobotSeismicResidualModeDefinitionType

**C++**

```
enum IRobotSeismicResidualModeDefinitionType;
```

**C#**

```
public enum IRobotSeismicResidualModeDefinitionType;
```

**Visual Basic**

```
Public Enum IRobotSeismicResidualModeDefinitionType
```

#### Members

Members	Description
I_SRMDT_LAST_CALC_MODE = 0	Available since version 14.7.
I_SRMDT_LIMIT_FREQUENCY = 1	Available since version 14.7.

#### Version

Available since version 14.7.

## II.1.5.70 IRobotSeismicResidualMode

### Class Hierarchy

#### C++

```
interface IRobotSeismicResidualMode : IDispatch;
```

#### C#

```
public interface IRobotSeismicResidualMode;
```

### Visual Basic

```
Public Interface IRobotSeismicResidualMode
```

### Version

Available since version 14.7.

## II.1.5.70.1 IRobotSeismicResidualMode Members

The following tables list the members exposed by IRobotSeismicResidualMode.

### Public Fields

	Name	Description
◆	AugmentationFactor ( <a href="#">see page 292</a> )	
◆	DefinitionMethod ( <a href="#">see page 293</a> )	
◆	IsActive ( <a href="#">see page 293</a> )	
◆	LimitFrequency ( <a href="#">see page 293</a> )	

## II.1.5.70.2 IRobotSeismicResidualMode Fields

The fields of the IRobotSeismicResidualMode class are listed here.

### Public Fields

	Name	Description
◆	AugmentationFactor ( <a href="#">see page 292</a> )	
◆	DefinitionMethod ( <a href="#">see page 293</a> )	
◆	IsActive ( <a href="#">see page 293</a> )	
◆	LimitFrequency ( <a href="#">see page 293</a> )	

## II.1.5.70.2.1 AugmentationFactor

#### C++

```
HRESULT get_AugmentationFactor(double* );
HRESULT put_AugmentationFactor(double);
```

#### C#

```
public double AugmentationFactor { get; set; }
```

### Visual Basic

```
Public AugmentationFactor As Double
```

### Version

Available since version 14.7.

### II.1.5.70.2.2 DefinitionMethod

#### C++

```
HRESULT get_DefinitionMethod(IRobotSeismicResidualModeDefinitionType* );
HRESULT put_DefinitionMethod(IRobotSeismicResidualModeDefinitionType);
```

#### C#

```
public IRobotSeismicResidualModeDefinitionType DefinitionMethod { get; set; }
```

#### Visual Basic

```
Public DefinitionMethod As IRobotSeismicResidualModeDefinitionType
```

#### Version

Available since version 14.7.

### II.1.5.70.2.3 IsActive

#### C++

```
HRESULT get_IsActive(VARIANT_BOOL* );
HRESULT put_IsActive(VARIANT_BOOL);
```

#### C#

```
public bool IsActive { get; set; }
```

#### Visual Basic

```
Public IsActive As Boolean
```

#### Version

Available since version 14.7.

### II.1.5.70.2.4 LimitFrequency

#### C++

```
HRESULT get_LimitFrequency(double* );
HRESULT put_LimitFrequency(double);
```

#### C#

```
public double LimitFrequency { get; set; }
```

#### Visual Basic

```
Public LimitFrequency As Double
```

#### Version

Available since version 14.7.

### II.1.5.71 IRobotSeismicAnalysis\_EC8\_General\_Params

#### Class Hierarchy

#### C++

```
interface IRobotSeismicAnalysis_EC8_General_Params : IDispatch;
```

#### C#

```
public interface IRobotSeismicAnalysis_EC8_General_Params;
```

#### Visual Basic

```
Public Interface IRobotSeismicAnalysis_EC8_General_Params
```

## Description

### Version

Available since version 14.7.

### II.1.5.71.1 IRobotSeismicAnalysis\_EC8\_General\_Params Members

The following tables list the members exposed by IRobotSeismicAnalysis\_EC8\_General\_Params.

#### Public Fields

	Name	Description
◆	Ag ( <a href="#">see page 294</a> )	.
◆	B ( <a href="#">see page 295</a> )	.
◆	BehaviorFactor ( <a href="#">see page 295</a> )	.
◆	Direction ( <a href="#">see page 295</a> )	.
◆	DirectionType ( <a href="#">see page 296</a> )	.
◆	ExcitationDir ( <a href="#">see page 296</a> )	.
◆	Filter ( <a href="#">see page 296</a> )	.
◆	ResidualMode ( <a href="#">see page 296</a> )	.
◆	S ( <a href="#">see page 297</a> )	.
◆	Spectrum ( <a href="#">see page 297</a> )	.
◆	Tb ( <a href="#">see page 297</a> )	.
◆	Tc ( <a href="#">see page 297</a> )	.
◆	Td ( <a href="#">see page 298</a> )	.

### II.1.5.71.2 IRobotSeismicAnalysis\_EC8\_General\_Params Fields

The fields of the IRobotSeismicAnalysis\_EC8\_General\_Params class are listed here.

#### Public Fields

	Name	Description
◆	Ag ( <a href="#">see page 294</a> )	.
◆	B ( <a href="#">see page 295</a> )	.
◆	BehaviorFactor ( <a href="#">see page 295</a> )	.
◆	Direction ( <a href="#">see page 295</a> )	.
◆	DirectionType ( <a href="#">see page 296</a> )	.
◆	ExcitationDir ( <a href="#">see page 296</a> )	.
◆	Filter ( <a href="#">see page 296</a> )	.
◆	ResidualMode ( <a href="#">see page 296</a> )	.
◆	S ( <a href="#">see page 297</a> )	.
◆	Spectrum ( <a href="#">see page 297</a> )	.
◆	Tb ( <a href="#">see page 297</a> )	.
◆	Tc ( <a href="#">see page 297</a> )	.
◆	Td ( <a href="#">see page 298</a> )	.

### II.1.5.71.2.1 Ag

#### C++

```
HRESULT get_Ag(double* );
HRESULT put_Ag(double );
```

**C#**

```
public double Ag { get; set; }
```

**Visual Basic**

```
Public Ag As Double
```

**Description****Version**

Available since version 14.7.

**II.1.5.71.2.2 B****C++**

```
HRESULT get_B(double*);  
HRESULT put_B(double);
```

**C#**

```
public double B { get; set; }
```

**Visual Basic**

```
Public B As Double
```

**Version**

Available since version 14.7.

**II.1.5.71.2.3 BehaviorFactor****C++**

```
HRESULT get_BehaviorFactor(double*);  
HRESULT put_BehaviorFactor(double);
```

**C#**

```
public double BehaviorFactor { get; set; }
```

**Visual Basic**

```
Public BehaviorFactor As Double
```

**Version**

Available since version 14.7.

**II.1.5.71.2.4 Direction****C++**

```
HRESULT get_Direction(IRobotGeoPoint3D**);  
HRESULT put_Direction(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Direction { get; set; }
```

**Visual Basic**

```
Public Direction As IRobotGeoPoint3D
```

**Version**

Available since version 14.7.

### II.1.5.71.2.5 DirectionType

#### C++

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

#### C#

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

#### Visual Basic

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

#### Version

Available since version 14.7.

### II.1.5.71.2.6 ExcitationDir

#### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

#### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

#### Visual Basic

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

#### Description

#### Version

Available since version 14.7.

### II.1.5.71.2.7 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Version

Available since version 14.7.

### II.1.5.71.2.8 ResidualMode

#### C++

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode** );
```

#### C#

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

#### Visual Basic

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

**Version**

Available since version 14.7.

**II.1.5.71.2.9 S****C++**

```
HRESULT get_S(double*);  
HRESULT put_S(double);
```

**C#**

```
public double S { get; set; }
```

**Visual Basic**

```
Public S As Double
```

**Version**

Available since version 14.7.

**II.1.5.71.2.10 Spectrum****C++**

```
HRESULT get_Spectrum(IRobotSeismicAnalysisSpectrumType*);  
HRESULT put_Spectrum(IRobotSeismicAnalysisSpectrumType);
```

**C#**

```
public IRobotSeismicAnalysisSpectrumType Spectrum { get; set; }
```

**Visual Basic**

```
Public Spectrum As IRobotSeismicAnalysisSpectrumType
```

**Version**

Available since version 14.7.

**II.1.5.71.2.11 Tb****C++**

```
HRESULT get_Tb(double*);  
HRESULT put_Tb(double);
```

**C#**

```
public double Tb { get; set; }
```

**Visual Basic**

```
Public Tb As Double
```

**Version**

Available since version 14.7.

**II.1.5.71.2.12 Tc****C++**

```
HRESULT get_Tc(double*);  
HRESULT put_Tc(double);
```

**C#**

```
public double Tc { get; set; }
```

**Visual Basic**

```
Public Tc As double
```

**Version**

Available since version 14.7.

**II.1.5.71.2.13 Td****C++**

```
HRESULT get_Td(double* );
HRESULT put_Td(double);
```

**C#**

```
public double Td { get; set; }
```

**Visual Basic**

```
Public Td As double
```

**Version**

Available since version 14.7.

**II.1.5.72 IRobotSeismicAnalysis\_EC8\_SpectrumType****C++**

```
enum IRobotSeismicAnalysis_EC8_SpectrumType;
```

**C#**

```
public enum IRobotSeismicAnalysis_EC8_SpectrumType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_EC8_SpectrumType
```

**Members**

Members	Description
I_EC8_SAST_TYPE_1 = 0	Available since version 14.7.
I_EC8_SAST_TYPE_2 = 1	Available since version 14.7.

**Version**

Available since version 14.7.

**II.1.5.73 IRobotSeismicAnalysis\_EC8\_GroundType****C++**

```
enum IRobotSeismicAnalysis_EC8_GroundType;
```

**C#**

```
public enum IRobotSeismicAnalysis_EC8_GroundType;
```

**Visual Basic**

```
Public Enum IRobotSeismicAnalysis_EC8_GroundType
```

**Members**

Members	Description
I_EC8_SAGT_A = 0	Available since version 14.7.

I_EC8_SAGT_B = 1	Available since version 14.7.
I_EC8_SAGT_C = 2	Available since version 14.7.
I_EC8_SAGT_D = 3	Available since version 14.7.
I_EC8_SAGT_E = 4	Available since version 14.7.

**Version**

Available since version 14.7.

**II.1.5.74 IRobotSeismicAnalysis\_EC8\_Params****Class Hierarchy****C++**

```
interface IRobotSeismicAnalysis_EC8_Params : IDispatch;
```

**C#**

```
public interface IRobotSeismicAnalysis_EC8_Params;
```

**Visual Basic**

```
Public Interface IRobotSeismicAnalysis_EC8_Params
```

**Description****Version**

Available since version 14.7.

**II.1.5.74.1 IRobotSeismicAnalysis\_EC8\_Params Members**

The following tables list the members exposed by IRobotSeismicAnalysis\_EC8\_Params.

**Public Fields**

	Name	Description
◆	Ag ( <a href="#">see page 300</a> )	.
◆	BehaviorFactor ( <a href="#">see page 300</a> )	.
◆	Direction ( <a href="#">see page 300</a> )	.
◆	DirectionType ( <a href="#">see page 301</a> )	.
◆	ExcitationDir ( <a href="#">see page 301</a> )	.
◆	Filter ( <a href="#">see page 301</a> )	.
◆	GroundType ( <a href="#">see page 301</a> )	.
◆	ResidualMode ( <a href="#">see page 302</a> )	.
◆	Spectrum ( <a href="#">see page 302</a> )	.
◆	SpectrumType ( <a href="#">see page 302</a> )	.

**II.1.5.74.2 IRobotSeismicAnalysis\_EC8\_Params Fields**

The fields of the IRobotSeismicAnalysis\_EC8\_Params class are listed here.

**Public Fields**

	Name	Description
◆	Ag ( <a href="#">see page 300</a> )	.
◆	BehaviorFactor ( <a href="#">see page 300</a> )	.
◆	Direction ( <a href="#">see page 300</a> )	.
◆	DirectionType ( <a href="#">see page 301</a> )	.

❖	ExcitationDir (see page 301)	.
❖	Filter (see page 301)	
❖	GroundType (see page 301)	
❖	ResidualMode (see page 302)	
❖	Spectrum (see page 302)	
❖	SpectrumType (see page 302)	

### II.1.5.74.2.1 Ag

#### C++

```
HRESULT get_Ag(double* );
HRESULT put_Ag(double);
```

#### C#

```
public double Ag { get; set; }
```

#### Visual Basic

```
Public Ag As Double
```

#### Description

#### Version

Available since version 14.7.

### II.1.5.74.2.2 BehaviorFactor

#### C++

```
HRESULT get_BehaviorFactor(double* );
HRESULT put_BehaviorFactor(double);
```

#### C#

```
public double BehaviorFactor { get; set; }
```

#### Visual Basic

```
Public BehaviorFactor As Double
```

#### Version

Available since version 14.7.

### II.1.5.74.2.3 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D** );
HRESULT put_Direction(IRobotGeoPoint3D* );
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Version

Available since version 14.7.

#### II.1.5.74.2.4 DirectionType

##### C++

```
HRESULT get_DirectionType(IRobotSeismicAnalysisDirectionType* );
HRESULT put_DirectionType(IRobotSeismicAnalysisDirectionType);
```

##### C#

```
public IRobotSeismicAnalysisDirectionType DirectionType { get; set; }
```

##### Visual Basic

```
Public DirectionType As IRobotSeismicAnalysisDirectionType
```

##### Version

Available since version 14.7.

#### II.1.5.74.2.5 ExcitationDir

##### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection** );
```

##### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

##### Visual Basic

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

##### Description

##### Version

Available since version 14.7.

#### II.1.5.74.2.6 Filter

##### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter** );
HRESULT put_Filter(IRobotCaseAnalysisModesFilter* );
```

##### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

##### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

##### Version

Available since version 14.7.

#### II.1.5.74.2.7 GroundType

##### C++

```
HRESULT get_GroundType(IRobotSeismicAnalysis_EC8_GroundType* );
HRESULT put_GroundType(IRobotSeismicAnalysis_EC8_GroundType);
```

##### C#

```
public IRobotSeismicAnalysis_EC8_GroundType GroundType { get; set; }
```

**Visual Basic**

```
Public GroundType As IRobotSeismicAnalysis_EC8_GroundType
```

**Version**

Available since version 14.7.

**II.1.5.74.2.8 ResidualMode****C++**

```
HRESULT get_ResidualMode(IRobotSeismicResidualMode**);
```

**C#**

```
public IRobotSeismicResidualMode ResidualMode { get; }
```

**Visual Basic**

```
Public ReadOnly ResidualMode As IRobotSeismicResidualMode
```

**Version**

Available since version 14.7.

**II.1.5.74.2.9 Spectrum****C++**

```
HRESULT get_Spectrum(IRobotSeismicAnalysisSpectrumType*);  
HRESULT put_Spectrum(IRobotSeismicAnalysisSpectrumType);
```

**C#**

```
public IRobotSeismicAnalysisSpectrumType Spectrum { get; set; }
```

**Visual Basic**

```
Public Spectrum As IRobotSeismicAnalysisSpectrumType
```

**Version**

Available since version 14.7.

**II.1.5.74.2.10 SpectrumType****C++**

```
HRESULT get_SpectrumType(IRobotSeismicAnalysis_EC8_SpectrumType*);  
HRESULT put_SpectrumType(IRobotSeismicAnalysis_EC8_SpectrumType);
```

**C#**

```
public IRobotSeismicAnalysis_EC8_SpectrumType SpectrumType { get; set; }
```

**Visual Basic**

```
Public SpectrumType As IRobotSeismicAnalysis_EC8_SpectrumType
```

**Version**

Available since version 14.7.

**II.1.6 Spectral analysis parameters**

Available since version 2.5.

## Enumerations

	Name	Description
	IRobotSpectralAnalysisAbscissaXAxisType ( <a href="#">see page 313</a> )	Available quantities on the abscissa axis. .
	IRobotSpectralAnalysisOrdinateYAxisType ( <a href="#">see page 314</a> )	Available quantities on the ordinate axis. .

## Interfaces

	Name	Description
	IRobotSpectralAnalysisParams ( <a href="#">see page 303</a> )	Parameters of spectral analysis.
	IRobotSpectralAnalysisSpectrum ( <a href="#">see page 305</a> )	Spectrum definition.
	IRobotSpectralAnalysisPointsCollection ( <a href="#">see page 310</a> )	Collection of points defining spectrum. .

### II.1.6.1 IRobotSpectralAnalysisParams

#### Class Hierarchy

#### C++

```
interface IRobotSpectralAnalysisParams : IDispatch;
```

#### C#

```
public interface IRobotSpectralAnalysisParams;
```

#### Visual Basic

```
Public Interface IRobotSpectralAnalysisParams
```

#### Description

Parameters of spectral analysis.

#### Version

Available since version 2.5.

### II.1.6.1.1 IRobotSpectralAnalysisParams Members

The following tables list the members exposed by IRobotSpectralAnalysisParams.

#### Public Fields

	Name	Description
	Direction ( <a href="#">see page 304</a> )	Spectral excitation direction.
	ExcitationDir ( <a href="#">see page 304</a> )	
	Filter ( <a href="#">see page 304</a> )	Description of modes taken into account during structure dynamic analysis.
	Spectrum ( <a href="#">see page 304</a> )	Spectrum definition.

### II.1.6.1.2 IRobotSpectralAnalysisParams Fields

The fields of the IRobotSpectralAnalysisParams class are listed here.

#### Public Fields

	Name	Description
	Direction ( <a href="#">see page 304</a> )	Spectral excitation direction.
	ExcitationDir ( <a href="#">see page 304</a> )	

	Filter ( <a href="#">see page 304</a> )	Description of modes taken into account during structure dynamic analysis.
	Spectrum ( <a href="#">see page 304</a> )	Spectrum definition.

### II.1.6.1.2.1 Direction

#### C++

```
HRESULT get_Direction(IRobotGeoPoint3D**);
HRESULT put_Direction(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotGeoPoint3D
```

#### Description

Spectral excitation direction.

#### Version

Available since version 2.5.

### II.1.6.1.2.2 ExcitationDir

#### C++

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

#### C#

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

#### Visual Basic

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

### II.1.6.1.2.3 Filter

#### C++

```
HRESULT get_Filter(IRobotCaseAnalysisModesFilter**);
HRESULT put_Filter(IRobotCaseAnalysisModesFilter*);
```

#### C#

```
public IRobotCaseAnalysisModesFilter Filter { get; set; }
```

#### Visual Basic

```
Public Filter As IRobotCaseAnalysisModesFilter
```

#### Description

Description of modes taken into account during structure dynamic analysis.

#### Version

Available since version 2.5.

### II.1.6.1.2.4 Spectrum

#### C++

```
HRESULT get_Spectrum(IRobotSpectralAnalysisSpectrum**);
HRESULT put_Spectrum(IRobotSpectralAnalysisSpectrum*);
```

**C#**

```
public IRobotSpectralAnalysisSpectrum Spectrum { get; set; }
```

**Visual Basic**

```
Public Spectrum As IRobotSpectralAnalysisSpectrum
```

**Description**

Spectrum definition.

**Version**

Available since version 2.5.

**II.1.6.2 IRobotSpectralAnalysisSpectrum****Class Hierarchy****C++**

```
interface IRobotSpectralAnalysisSpectrum : IDispatch;
```

**C#**

```
public interface IRobotSpectralAnalysisSpectrum;
```

**Visual Basic**

```
Public Interface IRobotSpectralAnalysisSpectrum
```

**Description**

Spectrum definition.

**Version**

Available since version 2.5.

**II.1.6.2.1 IRobotSpectralAnalysisSpectrum Members**

The following tables list the members exposed by IRobotSpectralAnalysisSpectrum.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AbscissaXAxis ( <a href="#">see page 306</a> )	Quantity on the abscissa axis .
◆	AbscissaXAxisLogarithmicScale ( <a href="#">see page 306</a> )	Logarithmic scale on the abscissa axis.
◆	Damping ( <a href="#">see page 306</a> )	Damping.
◆	Name ( <a href="#">see page 307</a> )	Spectrum name.
◆	OrdinateYAxis ( <a href="#">see page 307</a> )	Quantity on the ordinate axis.
◆	OrdinateYAxisLogarithmicScale ( <a href="#">see page 307</a> )	Logarithmic scale on the ordinate axis.
◆	Points ( <a href="#">see page 308</a> )	Points defining spectrum.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	AddFromTimeHistory ( <a href="#">see page 308</a> )	To the collection of points defining spectrum the function adds points of the spectrum calculated based on the time-dependent function and specified parameters. .
◆	Average ( <a href="#">see page 309</a> )	Function calculates the spectrum by averaging the indicated spectra.
◆	LoadFromFile ( <a href="#">see page 309</a> )	Function reads the spectrum from a file. .
◆	SaveToFile ( <a href="#">see page 309</a> )	Function saves the spectrum to a file.

## II.1.6.2.2 IRobotSpectralAnalysisSpectrum Fields

The fields of the IRobotSpectralAnalysisSpectrum class are listed here.

### Public Fields

	Name	Description
◆	AbscissaXAxis (see page 306)	Quantity on the abscissa axis .
◆	AbscissaXAxisLogarithmicScale (see page 306)	Logarithmic scale on the abscissa axis.
◆	Damping (see page 306)	Damping.
◆	Name (see page 307)	Spectrum name.
◆	OrdinateYAxis (see page 307)	Quantity on the ordinate axis.
◆	OrdinateYAxisLogarithmicScale (see page 307)	Logarithmic scale on the ordinate axis.
◆	Points (see page 308)	Points defining spectrum.

### II.1.6.2.2.1 AbscissaXAxis

#### C++

```
HRESULT get_AbscissaXAxis(IRobotSpectralAnalysisAbscissaXAxisType* );
HRESULT put_AbscissaXAxis(IRobotSpectralAnalysisAbscissaXAxisType);
```

#### C#

```
public IRobotSpectralAnalysisAbscissaXAxisType AbscissaXAxis { get; set; }
```

#### Visual Basic

```
Public AbscissaXAxis As IRobotSpectralAnalysisAbscissaXAxisType
```

#### Description

Quantity on the abscissa axis .

#### Version

Available since version 2.5.

### II.1.6.2.2.2 AbscissaXAxisLogarithmicScale

#### C++

```
HRESULT get_AbscissaXAxisLogarithmicScale(VARIANT_BOOL* );
HRESULT put_AbscissaXAxisLogarithmicScale(VARIANT_BOOL);
```

#### C#

```
public bool AbscissaXAxisLogarithmicScale { get; set; }
```

#### Visual Basic

```
Public AbscissaXAxisLogarithmicScale As Boolean
```

#### Description

Logarithmic scale on the abscissa axis.

#### Version

Available since version 2.5.

### II.1.6.2.2.3 Damping

#### C++

```
HRESULT get_Damping(double* );
HRESULT put_Damping(double);
```

**C#**

```
public double Damping { get; set; }
```

**Visual Basic**

```
Public Damping As Double
```

**Description**

Damping.

**Version**

Available since version 2.5.

**II.1.6.2.2.4 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public BSTR Name { get; set; }
```

**Visual Basic**

```
Public Name As BSTR
```

**Description**

Spectrum name.

**Version**

Available since version 2.5.

**II.1.6.2.2.5 OrdinateYAxis****C++**

```
HRESULT get_OrdinateYAxis(IRobotSpectralAnalysisOrdinateYAxisType* );
HRESULT put_OrdinateYAxis(IRobotSpectralAnalysisOrdinateYAxisType);
```

**C#**

```
public IRobotSpectralAnalysisOrdinateYAxisType OrdinateYAxis { get; set; }
```

**Visual Basic**

```
Public OrdinateYAxis As IRobotSpectralAnalysisOrdinateYAxisType
```

**Description**

Quantity on the ordinate axis.

**Version**

Available since version 2.5.

**II.1.6.2.2.6 OrdinateYAxisLogarithmicScale****C++**

```
HRESULT get_OrdinateYAxisLogarithmicScale(VARIANT_BOOL* );
HRESULT put_OrdinateYAxisLogarithmicScale(VARIANT_BOOL);
```

**C#**

```
public bool OrdinateYAxisLogarithmicScale { get; set; }
```

**Visual Basic**

```
Public OrdinateYAxisLogarithmicScale As Boolean
```

**Description**

Logarithmic scale on the ordinate axis.

**Version**

Available since version 2.5.

**II.1.6.2.2.7 Points****C++**

```
HRESULT get_Points(IRobotSpectralAnalysisPointsCollection**);
HRESULT put_Points(IRobotSpectralAnalysisPointsCollection*);
```

**C#**

```
public IRobotSpectralAnalysisPointsCollection Points { get; set; }
```

**Visual Basic**

```
Public Points As IRobotSpectralAnalysisPointsCollection
```

**Description**

Points defining spectrum.

**Version**

Available since version 2.5.

**II.1.6.2.3 IRobotSpectralAnalysisSpectrum Methods**

The methods of the IRobotSpectralAnalysisSpectrum class are listed here.

**Public Methods**

	Name	Description
新加	AddFromTimeHistory (见 page 308)	To the collection of points defining spectrum the function adds points of the spectrum calculated based on the time-dependent function and specified parameters. .
新加	Average (见 page 309)	Function calculates the spectrum by averaging the indicated spectra.
新加	LoadFromFile (见 page 309)	Function reads the spectrum from a file. .
新加	SaveToFile (见 page 309)	Function saves the spectrum to a file.

**II.1.6.2.3.1 AddFromTimeHistory****C++**

```
HRESULT AddFromTimeHistory(IRobotSpectralAnalysisPointsCollection* _time_history, double
_t_min, double _t_max, long _points_count, double _damping);
```

**C#**

```
public void AddFromTimeHistory(IRobotSpectralAnalysisPointsCollection _time_history, double
_t_min, double _t_max, long _points_count, double _damping);
```

**Visual Basic**

```
Public Sub AddFromTimeHistory(ByRef _time_history As
IRobotSpectralAnalysisPointsCollection, _t_min As Double, _t_max As Double, _points_count
As Long, _damping As Double)
```

**Description**

To the collection of points defining spectrum the function adds points of the spectrum calculated based on the

time-dependent function and specified parameters. .

#### Version

Available since version 2.5.

### II.1.6.2.3.2 Average

#### C++

```
HRESULT Average(IRobotSpectralAnalysisSpectrum* _spectrum_1,  
IRobotSpectralAnalysisSpectrum* _spectrum_2, double _damping);
```

#### C#

```
public void Average(IRobotSpectralAnalysisSpectrum _spectrum_1,  
IRobotSpectralAnalysisSpectrum _spectrum_2, double _damping);
```

#### Visual Basic

```
Public Sub Average(ByRef _spectrum_1 As IRobotSpectralAnalysisSpectrum, ByRef _spectrum_2  
As IRobotSpectralAnalysisSpectrum, _damping As Double)
```

#### Description

Function calculates the spectrum by averaging the indicated spectra.

#### Version

Available since version 2.5.

### II.1.6.2.3.3 LoadFromFile

#### C++

```
HRESULT LoadFromFile(BSTR _file_path, BSTR _spectrum_name);
```

#### C#

```
public void LoadFromFile(BSTR _file_path, BSTR _spectrum_name);
```

#### Visual Basic

```
Public Sub LoadFromFile(_file_path As BSTR, _spectrum_name As BSTR)
```

#### Description

Function reads the spectrum from a file. .

#### Version

Available since version 2.5.

### II.1.6.2.3.4 SaveToFile

#### C++

```
HRESULT SaveToFile(BSTR _file_path);
```

#### C#

```
public void SaveToFile(BSTR _file_path);
```

#### Visual Basic

```
Public Sub SaveToFile(_file_path As BSTR)
```

#### Description

Function saves the spectrum to a file.

**Version**

Available since version 2.5.

**II.1.6.3 IRobotSpectralAnalysisPointsCollection****Class Hierarchy****C++**

```
interface IRobotSpectralAnalysisPointsCollection : IDispatch;
```

**C#**

```
public interface IRobotSpectralAnalysisPointsCollection;
```

**Visual Basic**

```
Public Interface IRobotSpectralAnalysisPointsCollection
```

**Description**

Collection of points defining spectrum. .

**Version**

Available since version 2.5.

**II.1.6.3.1 IRobotSpectralAnalysisPointsCollection Members**

The following tables list the members exposed by IRobotSpectralAnalysisPointsCollection.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 310</a> )	Number of points defining spectrum.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add ( <a href="#">see page 311</a> )	Function adds a new point to the spectrum definition. .
◆	Clear ( <a href="#">see page 311</a> )	Function deletes all the points from the collection defining spectrum. .
◆	Get ( <a href="#">see page 312</a> )	Function returns coordinates of the selected point defining spectrum.
◆	LoadFromFile ( <a href="#">see page 312</a> )	Function reads out the point collection from a file. .
◆	Remove ( <a href="#">see page 312</a> )	Function deletes the point from the collection defining spectrum.
◆	SaveToFile ( <a href="#">see page 313</a> )	Function saves the point collection to a file. .
◆	Set ( <a href="#">see page 313</a> )	Function modifies the selected point defining the spectrum. .

**II.1.6.3.2 IRobotSpectralAnalysisPointsCollection Fields**

The fields of the IRobotSpectralAnalysisPointsCollection class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 310</a> )	Number of points defining spectrum.

**II.1.6.3.2.1 Count****C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of points defining spectrum.

**Version**

Available since version 2.5.

**II.1.6.3.3 IRobotSpectralAnalysisPointsCollection Methods**

The methods of the IRobotSpectralAnalysisPointsCollection class are listed here.

**Public Methods**

	Name	Description
>Add ( <a href="#">see page 311</a> )		Function adds a new point to the spectrum definition. .
Clear ( <a href="#">see page 311</a> )		Function deletes all the points from the collection defining spectrum. .
Get ( <a href="#">see page 312</a> )		Function returns coordinates of the selected point defining spectrum.
LoadFromFile ( <a href="#">see page 312</a> )		Function reads out the point collection from a file. .
Remove ( <a href="#">see page 312</a> )		Function deletes the point from the collection defining spectrum.
SaveToFile ( <a href="#">see page 313</a> )		Function saves the point collection to a file. .
Set ( <a href="#">see page 313</a> )		Function modifies the selected point defining the spectrum. .

**II.1.6.3.3.1 Add****C++**

```
HRESULT Add(double _x, double _y);
```

**C#**

```
public void Add(double _x, double _y);
```

**Visual Basic**

```
Public Sub Add(_x As double, _y As double)
```

**Description**

Function adds a new point to the spectrum definition. .

**Version**

Available since version 2.5.

**II.1.6.3.3.2 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

Function deletes all the points from the collection defining spectrum. .

**Version**

Available since version 2.5.

**II.1.6.3.3 Get****C++**

```
HRESULT Get(long _pos, double* _x, double* _y);
```

**C#**

```
public void Get(long _pos, double* _x, double* _y);
```

**Visual Basic**

```
Public Sub Get(_pos As long, ByRef _x As double*, ByRef _y As double*)
```

**Description**

Function returns coordinates of the selected point defining spectrum.

**Version**

Available since version 2.5.

**II.1.6.3.4 LoadFromFile****C++**

```
HRESULT LoadFromFile(BSTR _file_path);
```

**C#**

```
public void LoadFromFile(BSTR _file_path);
```

**Visual Basic**

```
Public Sub LoadFromFile(_file_path As BSTR)
```

**Description**

Function reads out the point collection from a file..

**Version**

Available since version 2.5.

**II.1.6.3.5 Remove****C++**

```
HRESULT Remove(long _pos);
```

**C#**

```
public void Remove(long _pos);
```

**Visual Basic**

```
Public Sub Remove(_pos As long)
```

**Description**

Function deletes the point from the collection defining spectrum.

**Version**

Available since version 2.5.

### II.1.6.3.3.6 SaveToFile

#### C++

```
HRESULT SaveToFile(BSTR _file_path);
```

#### C#

```
public void SaveToFile(BSTR _file_path);
```

#### Visual Basic

```
Public Sub SaveToFile(_file_path As BSTR)
```

#### Description

Function saves the point collection to a file. .

#### Version

Available since version 2.5.

### II.1.6.3.3.7 Set

#### C++

```
HRESULT Set(long _pos, double _x, double _y);
```

#### C#

```
public void Set(long _pos, double _x, double _y);
```

#### Visual Basic

```
Public Sub Set(_pos As long, _x As double, _y As double)
```

#### Description

Function modifies the selected point defining the spectrum. .

#### Version

Available since version 2.5.

### II.1.6.4 IRobotSpectralAnalysisAbscissaXAxisType

#### C++

```
enum IRobotSpectralAnalysisAbscissaXAxisType;
```

#### C#

```
public enum IRobotSpectralAnalysisAbscissaXAxisType;
```

#### Visual Basic

```
Public Enum IRobotSpectralAnalysisAbscissaXAxisType
```

#### Members

Members	Description
I_SAXAT_PERIOD = 0	Period. Available since version 2.5.
I_SAXAT_PULSATION = 1	Pulsation. Available since version 2.5.
I_SAXAT_FREQUENCY = 2	Frequency. Available since version 2.5.

**Description**

Available quantities on the abscissa axis. .

**Version**

Available since version 2.5.

**II.1.6.5 IRobotSpectralAnalysisOrdinateYAxisType****C++**

```
enum IRobotSpectralAnalysisOrdinateYAxisType;
```

**C#**

```
public enum IRobotSpectralAnalysisOrdinateYAxisType;
```

**Visual Basic**

```
Public Enum IRobotSpectralAnalysisOrdinateYAxisType
```

**Members**

Members	Description
I_SAOYAT_VELOCITY = 0	Velocity. Available since version 2.5.
I_SAOYAT_ACCELERATION = 1	Acceleration. Available since version 2.5.
I_SAOYAT_EXCITATION = 2	Displacement. Available since version 2.5.

**Description**

Available quantities on the ordinate axis. .

**Version**

Available since version 2.5.

**II.1.7 Non-linear analysis parameters**

Available since version 2.5.

**Enumerations**

	Name	Description
	IRobotNonlinearAnalysisAlgorithmType ( <a href="#">see page 314</a> )	Methods of non-linear analysis.

**Interfaces**

	Name	Description
	IRobotNonlinearAnalysisParams ( <a href="#">see page 315</a> )	Parameters of non-linear analysis.

**II.1.7.1 IRobotNonlinearAnalysisAlgorithmType****C++**

```
enum IRobotNonlinearAnalysisAlgorithmType;
```

**C#**

```
public enum IRobotNonlinearAnalysisAlgorithmType;
```

## Visual Basic

```
Public Enum IRobotNonlinearAnalysisAlgorithmType
```

### Members

Members	Description
I_NAAT_INCREMENTAL_METHOD = 0	Incremental method of non-linear analysis. Available since version 2.5.
I_NAAT_DIRECT_ITERATION_METHOD = 1	Direct iteration method. Available since version 2.5.
I_NAAT_ARC_LENGTH_METHOD = 2	Available since version 2.5.
I_NAAT_PREDICTOR_CORRECTOR_METHOD = 3	Available since version 3.

### Description

Methods of non-linear analysis.

### Version

Available since version 2.5.

## II.1.7.2 IRobotNonlinearAnalysisParams

### Class Hierarchy

#### C++

```
interface IRobotNonlinearAnalysisParams : IDispatch;
```

#### C#

```
public interface IRobotNonlinearAnalysisParams;
```

### Visual Basic

```
Public Interface IRobotNonlinearAnalysisParams
```

### Description

Parameters of non-linear analysis.

### Version

Available since version 2.5.

## II.1.7.2.1 IRobotNonlinearAnalysisParams Members

The following tables list the members exposed by IRobotNonlinearAnalysisParams.

### Public Fields

	Name	Description
◆	Algorithm ( <a href="#">see page 317</a> )	Method of non-linear analysis.
◆	DegreeOfFreedom ( <a href="#">see page 317</a> )	Definition of degree of freedom (for the "Arc-length" method) .
◆	DisplacementsRelativeCodeTolerance ( <a href="#">see page 317</a> )	Relative code tolerance for displacements.
◆	IncrementLengthReductionFactor ( <a href="#">see page 318</a> )	Increment length reduction factor.
◆	IncrementLengthReductionNumber ( <a href="#">see page 318</a> )	Increment length reduction number .
◆	LineSearchMethodFactor ( <a href="#">see page 318</a> )	Control parameter for line-search method .
◆	LoadIncrementNumber ( <a href="#">see page 318</a> )	Number of load increments.

◆	MatrixUpdateAfterEachIteration ( <a href="#">see page 319</a> )	Matrix update after each iteration.
◆	MatrixUpdateAfterEachSubdivision ( <a href="#">see page 319</a> )	Matrix update after each subdivision.
◆	MaxDisplacement ( <a href="#">see page 319</a> )	Maximum displacement for the selected degree of freedom (for the "Arc-length" method) .
◆	MaximumIterationNumberForOneIncrement ( <a href="#">see page 320</a> )	Maximum iteration number for one increment.
◆	MaximumNumberOfBFGSCorrections ( <a href="#">see page 320</a> )	Maximum number of BFGS corrections.
◆	MaximumNumberOfLineSearches ( <a href="#">see page 320</a> )	Maximum number of line searches.
◆	MaxLoadFactor ( <a href="#">see page 320</a> )	Maximum load factor (for the "Arc-length" method) .
◆	NodeNumber ( <a href="#">see page 321</a> )	Node number (for the "Arc-length" method) .
◆	PDelta ( <a href="#">see page 321</a> )	Definition of P-delta analysis.
◆	ResidualForcesRelativeCodeTolerance ( <a href="#">see page 321</a> )	Relative code tolerance for residual forces.
◆	ResultListEachIteration ( <a href="#">see page 322</a> )	Relative code tolerance for residual forces.
◆	Stiff ( <a href="#">see page 322</a> )	(for "Arc-length" method).

## Public Methods

	Name	Description
◆	GetSettingsFromPreferences ( <a href="#">see page 322</a> )	Function takes settings from preferences.
◆	SaveSettingsInPreferences ( <a href="#">see page 323</a> )	Function saves settings to preferences as default ones. .

### II.1.7.2.2 IRobotNonlinearAnalysisParams Fields

The fields of the IRobotNonlinearAnalysisParams class are listed here.

## Public Fields

	Name	Description
◆	Algorithm ( <a href="#">see page 317</a> )	Method of non-linear analysis.
◆	DegreeOfFreedom ( <a href="#">see page 317</a> )	Definition of degree of freedom (for the "Arc-length" method) .
◆	DisplacementsRelativeCodeTolerance ( <a href="#">see page 317</a> )	Relative code tolerance for displacements.
◆	IncrementLengthReductionFactor ( <a href="#">see page 318</a> )	Increment length reduction factor.
◆	IncrementLengthReductionNumber ( <a href="#">see page 318</a> )	Increment length reduction number .
◆	LineSearchMethodFactor ( <a href="#">see page 318</a> )	Control parameter for line-search method .
◆	LoadIncrementNumber ( <a href="#">see page 318</a> )	Number of load increments.
◆	MatrixUpdateAfterEachIteration ( <a href="#">see page 319</a> )	Matrix update after each iteration.
◆	MatrixUpdateAfterEachSubdivision ( <a href="#">see page 319</a> )	Matrix update after each subdivision.
◆	MaxDisplacement ( <a href="#">see page 319</a> )	Maximum displacement for the selected degree of freedom (for the "Arc-length" method) .
◆	MaximumIterationNumberForOneIncrement ( <a href="#">see page 320</a> )	Maximum iteration number for one increment.

◆	MaximumNumberOfBFGSCorrections ( [ see page 320) )	Maximum number of BFGS corrections.
◆	MaximumNumberOfLineSearches ( [ see page 320)	Maximum number of line searches.
◆	MaxLoadFactor ( [ see page 320)	Maximum load factor (for the "Arc-length" method) .
◆	NodeNumber ( [ see page 321)	Node number (for the "Arc-length" method) .
◆	PDelta ( [ see page 321)	Definition of P-delta analysis.
◆	ResidualForcesRelativeCodeTolerance ( [ see page 321)	Relative code tolerance for residual forces.
◆	ResultListEachIteration ( [ see page 322)	Relative code tolerance for residual forces.
◆	Stiff ( [ see page 322)	(for "Arc-length" method).

### II.1.7.2.2.1 Algorithm

#### C++

```
HRESULT get_Algorithm(IRobotNonlinearAnalysisAlgorithmType* );
HRESULT put_Algorithm(IRobotNonlinearAnalysisAlgorithmType);
```

#### C#

```
public IRobotNonlinearAnalysisAlgorithmType Algorithm { get; set; }
```

#### Visual Basic

```
Public Algorithm As IRobotNonlinearAnalysisAlgorithmType
```

#### Description

Method of non-linear analysis.

### II.1.7.2.2.2 DegreeOfFreedom

#### C++

```
HRESULT get_DegreeOfFreedom(IRobotDegreeOfFreedom* );
HRESULT put_DegreeOfFreedom(IRobotDegreeOfFreedom);
```

#### C#

```
public IRobotDegreeOfFreedom DegreeOfFreedom { get; set; }
```

#### Visual Basic

```
Public DegreeOfFreedom As IRobotDegreeOfFreedom
```

#### Description

Definition of degree of freedom (for the "Arc-length" method) .

#### Version

Available since version 2.5.

### II.1.7.2.2.3 DisplacementsRelativeCodeTolerance

#### C++

```
HRESULT get_DisplacementsRelativeCodeTolerance(double* );
HRESULT put_DisplacementsRelativeCodeTolerance(double);
```

#### C#

```
public double DisplacementsRelativeCodeTolerance { get; set; }
```

#### Visual Basic

```
Public DisplacementsRelativeCodeTolerance As double
```

**Description**

Relative code tolerance for displacements.

**Version**

Available since version 2.5.

**II.1.7.2.2.4 IncrementLengthReductionFactor****C++**

```
HRESULT get_IncrementLengthReductionFactor(double*);  
HRESULT put_IncrementLengthReductionFactor(double);
```

**C#**

```
public double IncrementLengthReductionFactor { get; set; }
```

**Visual Basic**

```
Public IncrementLengthReductionFactor As Double
```

**Description**

Increment length reduction factor.

**II.1.7.2.2.5 IncrementLengthReductionNumber****C++**

```
HRESULT get_IncrementLengthReductionNumber(long*);  
HRESULT put_IncrementLengthReductionNumber(long);
```

**C#**

```
public long IncrementLengthReductionNumber { get; set; }
```

**Visual Basic**

```
Public IncrementLengthReductionNumber As Long
```

**Description**

Increment length reduction number .

**II.1.7.2.2.6 LineSearchMethodFactor****C++**

```
HRESULT get_LineSearchMethodFactor(double*);  
HRESULT put_LineSearchMethodFactor(double);
```

**C#**

```
public double LineSearchMethodFactor { get; set; }
```

**Visual Basic**

```
Public LineSearchMethodFactor As Double
```

**Description**

Control parameter for line-search method .

**II.1.7.2.2.7 LoadIncrementNumber****C++**

```
HRESULT get_LoadIncrementNumber(long*);  
HRESULT put_LoadIncrementNumber(long);
```

**C#**

```
public long LoadIncrementNumber { get; set; }
```

**Visual Basic**

```
Public LoadIncrementNumber As long
```

**Description**

Number of load increments.

**II.1.7.2.2.8 MatrixUpdateAfterEachIteration****C++**

```
HRESULT get_MatrixUpdateAfterEachIteration(VARIANT_BOOL* );
HRESULT put_MatrixUpdateAfterEachIteration(VARIANT_BOOL);
```

**C#**

```
public bool MatrixUpdateAfterEachIteration { get; set; }
```

**Visual Basic**

```
Public MatrixUpdateAfterEachIteration As Boolean
```

**Description**

Matrix update after each iteration.

**II.1.7.2.2.9 MatrixUpdateAfterEachSubdivision****C++**

```
HRESULT get_MatrixUpdateAfterEachSubdivision(VARIANT_BOOL* );
HRESULT put_MatrixUpdateAfterEachSubdivision(VARIANT_BOOL);
```

**C#**

```
public bool MatrixUpdateAfterEachSubdivision { get; set; }
```

**Visual Basic**

```
Public MatrixUpdateAfterEachSubdivision As Boolean
```

**Description**

Matrix update after each subdivision.

**II.1.7.2.2.10 MaxDisplacement****C++**

```
HRESULT get_MaxDisplacement(double* );
HRESULT put_MaxDisplacement(double);
```

**C#**

```
public double MaxDisplacement { get; set; }
```

**Visual Basic**

```
Public MaxDisplacement As double
```

**Description**

Maximum displacement for the selected degree of freedom (for the "Arc-length" method) .

**Version**

Available since version 2.5.

### II.1.7.2.2.11 MaximumIterationNumberForOneIncrement

**C++**

```
HRESULT get_MaximumIterationNumberForOneIncrement(long* );
HRESULT put_MaximumIterationNumberForOneIncrement(long);
```

**C#**

```
public long MaximumIterationNumberForOneIncrement { get; set; }
```

**Visual Basic**

```
Public MaximumIterationNumberForOneIncrement As long
```

**Description**

Maximum iteration number for one increment.

### II.1.7.2.2.12 MaximumNumberOfBFGSCorrections

**C++**

```
HRESULT get_MaximumNumberOfBFGSCorrections(long* );
HRESULT put_MaximumNumberOfBFGSCorrections(long);
```

**C#**

```
public long MaximumNumberOfBFGSCorrections { get; set; }
```

**Visual Basic**

```
Public MaximumNumberOfBFGSCorrections As long
```

**Description**

Maximum number of BFGS corrections.

### II.1.7.2.2.13 MaximumNumberOfLineSearches

**C++**

```
HRESULT get_MaximumNumberOfLineSearches(long* );
HRESULT put_MaximumNumberOfLineSearches(long);
```

**C#**

```
public long MaximumNumberOfLineSearches { get; set; }
```

**Visual Basic**

```
Public MaximumNumberOfLineSearches As long
```

**Description**

Maximum number of line searches.

### II.1.7.2.2.14 MaxLoadFactor

**C++**

```
HRESULT get_MaxLoadFactor(double* );
HRESULT put_MaxLoadFactor(double);
```

**C#**

```
public double MaxLoadFactor { get; set; }
```

**Visual Basic**

```
Public MaxLoadFactor As double
```

**Description**

Maximum load factor (for the "Arc-length" method) .

**Version**

Available since version 2.5.

**II.1.7.2.2.15 NodeNumber****C++**

```
HRESULT get_NodeNumber(long* );
HRESULT put_NodeNumber(long);
```

**C#**

```
public long NodeNumber { get; set; }
```

**Visual Basic**

```
Public NodeNumber As long
```

**Description**

Node number (for the "Arc-length" method) .

**Version**

Available since version 2.5.

**II.1.7.2.2.16 PDelta****C++**

```
HRESULT get_PDelta(VARIANT_BOOL* );
HRESULT put_PDelta(VARIANT_BOOL);
```

**C#**

```
public bool PDelta { get; set; }
```

**Visual Basic**

```
Public PDelta As Boolean
```

**Description**

Definition of P-delta analysis.

**II.1.7.2.2.17 ResidualForcesRelativeCodeTolerance****C++**

```
HRESULT get_ResidualForcesRelativeCodeTolerance(double* );
HRESULT put_ResidualForcesRelativeCodeTolerance(double);
```

**C#**

```
public double ResidualForcesRelativeCodeTolerance { get; set; }
```

**Visual Basic**

```
Public ResidualForcesRelativeCodeTolerance As double
```

**Description**

Relative code tolerance for residual forces.

### II.1.7.2.2.18 ResultListEachIteration

#### C++

```
HRESULT get_ResultListEachIteration(VARIANT_BOOL* );
HRESULT put_ResultListEachIteration(VARIANT_BOOL);
```

#### C#

```
public bool ResultListEachIteration { get; set; }
```

#### Visual Basic

```
Public ResultListEachIteration As Boolean
```

#### Description

Relative code tolerance for residual forces.

### II.1.7.2.2.19 Stiff

#### C++

```
HRESULT get_Stiff(VARIANT_BOOL* );
HRESULT put_Stiff(VARIANT_BOOL);
```

#### C#

```
public bool Stiff { get; set; }
```

#### Visual Basic

```
Public Stiff As Boolean
```

#### Description

(for "Arc-length" method).

#### Version

Available since version 2.5.

### II.1.7.2.3 IRobotNonlinearAnalysisParams Methods

The methods of the IRobotNonlinearAnalysisParams class are listed here.

#### Public Methods

	Name	Description
💡	GetSettingsFromPreferences (↗ see page 322)	Function takes settings from preferences.
💡	SaveSettingsInPreferences (↗ see page 323)	Function saves settings to preferences as default ones. .

### II.1.7.2.3.1 GetSettingsFromPreferences

#### C++

```
HRESULT GetSettingsFromPreferences( );
```

#### C#

```
public void GetSettingsFromPreferences( );
```

#### Visual Basic

```
Public Sub GetSettingsFromPreferences( )
```

#### Description

Function takes settings from preferences.

## Version

Available since version 2.5.

### II.1.7.2.3.2 SaveSettingsInPreferences

#### C++

```
HRESULT SaveSettingsInPreferences();
```

#### C#

```
public void SaveSettingsInPreferences();
```

#### Visual Basic

```
Public Sub SaveSettingsInPreferences()
```

#### Description

Function saves settings to preferences as default ones. .

## Version

Available since version 2.5.

## II.1.8 Moving loads

Available since version 2.5.

### Enumerations

	Name	Description
	IRobotMobileCaseFlag ( <a href="#">see page 332</a> )	Available types of a moving load case. .
	IRobotMobileCaseApplicationPlaneType ( <a href="#">see page 340</a> )	Type of application plane.
	IRobotVehicleLoadType ( <a href="#">see page 348</a> )	Load types.

### Interfaces

	Name	Description
	IRobotMobileCase ( <a href="#">see page 324</a> )	Moving load case.
	IRobotMobileCaseComponentMngr ( <a href="#">see page 329</a> )	Component list of moving load.
	IRobotMobileCaseComponent ( <a href="#">see page 331</a> )	Simple component of a moving load case. This component corresponds to the simple case generated at the next point of the route. .
	IRobotMobileCaseRoute ( <a href="#">see page 333</a> )	Definition of vehicle route.
	IRobotMobileCaseSegmentFactors ( <a href="#">see page 336</a> )	A set of coefficients assigned to a route segment, correcting vehicle loads. .
	IRobotVehicleData ( <a href="#">see page 340</a> )	Parameter set for vehicle definition.
	IRobotVehicleLoadMngr ( <a href="#">see page 343</a> )	List of loads that define a vehicle.
	IRobotVehicleLoad ( <a href="#">see page 345</a> )	Record describing a single load associated with the vehicle. .
	IRobotVehicleDatabaseList ( <a href="#">see page 349</a> )	Interface describing the list of vehicle databases.
	IRobotVehicleDatabase ( <a href="#">see page 353</a> )	Interface for vehicle database support.

## II.1.8.1 IRobotMobileCase

### Class Hierarchy

#### C++

```
interface IRobotMobileCase : IRobotCase;
```

#### C#

```
public interface IRobotMobileCase : IRobotCase;
```

### Visual Basic

```
Public Interface IRobotMobileCase
```

### Description

Moving load case.

### Version

Available since version 2.5.

## II.1.8.1.1 IRobotMobileCase Members

The following tables list the members exposed by IRobotMobileCase.

### Public Fields

	Name	Description
◆	AnalyzeType ( <a href="#">see page 405</a> )	Analysis type.
◆	ApplicationPlaneBars ( <a href="#">see page 325</a> )	List of bars defining the plane to which the vehicle is applied (active parameter for ApplicationPlaneType ( <a href="#">see page 325</a> ) = I_MCPT_SELECTION).
◆	ApplicationPlaneType ( <a href="#">see page 325</a> )	Type ( <a href="#">see page 406</a> ) of the plane to which the vehicle is applied.
◆	Components ( <a href="#">see page 326</a> )	List of moving load case components; Each component corresponds to a simple case generated due to applying the vehicle to the determined node on the route. .
◆	Flag ( <a href="#">see page 326</a> )	Flag determining type of a moving load case; Possible flag values are described by the IRobotMobileCaseFlag ( <a href="#">see page 332</a> ) interface; In practice, this field is used for navigation between the base moving load case and its envelopes. .
◆	IsAuxiliary ( <a href="#">see page 326</a> )	
◆	Label ( <a href="#">see page 327</a> )	
◆	Name ( <a href="#">see page 405</a> )	User-defined case name .
◆	Nature ( <a href="#">see page 405</a> )	Case nature .
◆	NatureName ( <a href="#">see page 327</a> )	Name ( <a href="#">see page 405</a> ) of nature.
◆	Number ( <a href="#">see page 406</a> )	User-defined number linked with the load case .
◆	Type ( <a href="#">see page 406</a> )	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .
◆	UniqueId ( <a href="#">see page 327</a> )	Unique identifier of a load case.
◆	Vehicle ( <a href="#">see page 327</a> )	Name ( <a href="#">see page 405</a> ) identifying the vehicle.

### Public Methods

	Name	Description
◆	FindByFlag ( <a href="#">see page 328</a> )	Function returns number of the case of the determined type, associated with this moving load case. It allows navigation between the base moving load case and its envelopes. .
◆	GetRoute ( <a href="#">see page 328</a> )	Function takes the interface of route definition.

	SetNatureExt (see page 329)	Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .
	SetRoute (see page 329)	Function updates the route definition based on the indicated interface. .

### II.1.8.1.2 IRobotMobileCase Fields

The fields of the IRobotMobileCase class are listed here.

#### Public Fields

	Name	Description
	ApplicationPlaneBars (see page 325)	List of bars defining the plane to which the vehicle is applied (active parameter for ApplicationPlaneType (see page 325) = I_MCPT_SELECTION).
	ApplicationPlaneType (see page 325)	Type (see page 406) of the plane to which the vehicle is applied.
	Components (see page 326)	List of moving load case components; Each component corresponds to a simple case generated due to applying the vehicle to the determined node on the route. .
	Flag (see page 326)	Flag determining type of a moving load case; Possible flag values are described by the IRobotMobileCaseFlag (see page 332) interface; In practice, this field is used for navigation between the base moving load case and its envelopes. .
	IsAuxiliary (see page 326)	
	Label (see page 327)	
	NatureName (see page 327)	Name (see page 405) of nature.
	UniqueID (see page 327)	Unique identifier of a load case.
	Vehicle (see page 327)	Name (see page 405) identifying the vehicle.

#### II.1.8.1.2.1 ApplicationPlaneBars

##### C++

```
HRESULT get_ApplicationPlaneBars(BSTR* );
HRESULT put_ApplicationPlaneBars(BSTR);
```

##### C#

```
public String ApplicationPlaneBars { get; set; }
```

##### Visual Basic

```
Public ApplicationPlaneBars As String
```

##### Description

List of bars defining the plane to which the vehicle is applied (active parameter for ApplicationPlaneType (see page 325) = I\_MCPT\_SELECTION).

##### Version

Available since version 3.

#### II.1.8.1.2.2 ApplicationPlaneType

##### C++

```
HRESULT get_ApplicationPlaneType(IRobotMobileCaseApplicationPlaneType* );
HRESULT put_ApplicationPlaneType(IRobotMobileCaseApplicationPlaneType);
```

##### C#

```
public IRobotMobileCaseApplicationPlaneType ApplicationPlaneType { get; set; }
```

**Visual Basic**

```
Public ApplicationPlaneType As IRobotMobileCaseApplicationPlaneType
```

**Description**

Type (see page 406) of the plane to which the vehicle is applied.

**Version**

Available since version 3.

**II.1.8.1.2.3 Components****C++**

```
HRESULT get_Components(IRobotMobileCaseComponentMngr**);
```

**C#**

```
public IRobotMobileCaseComponentMngr Components { get; }
```

**Visual Basic**

```
Public ReadOnly Components As IRobotMobileCaseComponentMngr
```

**Description**

List of moving load case components; Each component corresponds to a simple case generated due to applying the vehicle to the determined node on the route. .

**Version**

Available since version 2.5.

**II.1.8.1.2.4 Flag****C++**

```
HRESULT get_Flag(IRobotMobileCaseFlag*);
```

**C#**

```
public IRobotMobileCaseFlag Flag { get; }
```

**Visual Basic**

```
Public ReadOnly Flag As IRobotMobileCaseFlag
```

**Description**

Flag determining type of a moving load case; Possible flag values are described by the IRobotMobileCaseFlag (see page 332) interface; In practice, this field is used for navigation between the base moving load case and its envelopes. .

**Version**

Available since version 2.5.

**II.1.8.1.2.5 IsAuxiliary****C++**

```
HRESULT get_IsAuxiliary(VARIANT_BOOL*);  
HRESULT put_IsAuxiliary(VARIANT_BOOL);
```

**C#**

```
public bool IsAuxiliary { get; set; }
```

**Visual Basic**

```
Public IsAuxiliary As Boolean
```

**Version**

Available since version 8.1.

**II.1.8.1.2.6 Label****C++**

```
HRESULT get_Label(BSTR* );
HRESULT put_Label(BSTR);
```

**C#**

```
public String Label { get; set; }
```

**Visual Basic**

```
Public Label As String
```

**Version**

Available since version 8.

**II.1.8.1.2.7 NatureName****C++**

```
HRESULT get_NatureName(BSTR* );
```

**C#**

```
public String NatureName { get; }
```

**Visual Basic**

```
Public ReadOnly NatureName As String
```

**Description**

Name (see page 405) of nature.

**Version**

Available since version 3.

**II.1.8.1.2.8 UniqueId****C++**

```
HRESULT get_UniqueId(long* );
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique identifier of a load case.

**Version**

Available since version 2.5.

**II.1.8.1.2.9 Vehicle****C++**

```
HRESULT get_Vehicle(BSTR* );
HRESULT put_Vehicle(BSTR);
```

**C#**

```
public String Vehicle { get; set; }
```

**Visual Basic**

```
Public Vehicle As String
```

**Description**

Name (see page 405) identifying the vehicle.

**Version**

Available since version 3.

**II.1.8.1.3 IRobotMobileCase Methods**

The methods of the IRobotMobileCase class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	FindByFlag (see page 328)	Function returns number of the case of the determined type, associated with this moving load case. It allows navigation between the base moving load case and its envelopes. .
💡	GetRoute (see page 328)	Function takes the interface of route definition.
💡	SetNatureExt (see page 329)	Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .
💡	SetRoute (see page 329)	Function updates the route definition based on the indicated interface. .

**II.1.8.1.3.1 FindByFlag****C++**

```
HRESULT FindByFlag(IRobotMobileCaseFlag _mc_flag, long* ret);
```

**C#**

```
public long FindByFlag(IRobotMobileCaseFlag _mc_flag);
```

**Visual Basic**

```
Public Function FindByFlag(_mc_flag As IRobotMobileCaseFlag) As long
```

**Description**

Function returns number of the case of the determined type, associated with this moving load case. It allows navigation between the base moving load case and its envelopes. .

**Version**

Available since version 2.5.

**II.1.8.1.3.2 GetRoute****C++**

```
HRESULT GetRoute(IRobotMobileCaseRoute** ret);
```

**C#**

```
public IRobotMobileCaseRoute GetRoute();
```

**Visual Basic**

```
Public Function GetRoute() As IRobotMobileCaseRoute
```

**Description**

Function takes the interface of route definition.

**Version**

Available since version 3.

**II.1.8.1.3.3 SetNatureExt****C++**

```
HRESULT SetNatureExt(long _rgl_action);
```

**C#**

```
public void SetNatureExt(long _rgl_action);
```

**Visual Basic**

```
Public Sub SetNatureExt(_rgl_action As long)
```

**Description**

Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .

**Version**

Available since version 3.

**II.1.8.1.3.4 SetRoute****C++**

```
HRESULT SetRoute(IRobotMobileCaseRoute _route);
```

**C#**

```
public void SetRoute(IRobotMobileCaseRoute _route);
```

**Visual Basic**

```
Public Sub SetRoute(_route As IRobotMobileCaseRoute)
```

**Description**

Function updates the route definition based on the indicated interface. .

**Version**

Available since version 3.

**II.1.8.2 IRobotMobileCaseComponentMngr****Class Hierarchy****C++**

```
interface IRobotMobileCaseComponentMngr : IDispatch;
```

**C#**

```
public interface IRobotMobileCaseComponentMngr;
```

**Visual Basic**

```
Public Interface IRobotMobileCaseComponentMngr
```

## Description

Component list of moving load.

## Version

Available since version 2.5.

### II.1.8.2.1 IRobotMobileCaseComponentMngr Members

The following tables list the members exposed by IRobotMobileCaseComponentMngr.

#### Public Fields

	Name	Description
◆	Count (see page 330)	Number of components on the list .

#### Public Methods

	Name	Description
✿	Get (see page 330)	Function returns the component with the specified index. Components of a moving load case are indexed from 1 to Count (see page 330). .

### II.1.8.2.2 IRobotMobileCaseComponentMngr Fields

The fields of the IRobotMobileCaseComponentMngr class are listed here.

#### Public Fields

	Name	Description
◆	Count (see page 330)	Number of components on the list .

### II.1.8.2.2.1 Count

#### C++

```
HRESULT get_Count( long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of components on the list .

#### Version

Available since version 2.5.

### II.1.8.2.3 IRobotMobileCaseComponentMngr Methods

The methods of the IRobotMobileCaseComponentMngr class are listed here.

#### Public Methods

	Name	Description
✿	Get (see page 330)	Function returns the component with the specified index. Components of a moving load case are indexed from 1 to Count (see page 330). .

### II.1.8.2.3.1 Get

#### C++

```
HRESULT Get( long _cmpnt_idx, IRobotMobileCaseComponent** ret );
```

**C#**

```
public IRobotMobileCaseComponent Get(long _cmpnt_idx);
```

**Visual Basic**

```
Public Function Get(_cmpnt_idx As long) As IRobotMobileCaseComponent
```

**Description**

Function returns the component with the specified index. Components of a moving load case are indexed from 1 to Count (see page 330). .

**Version**

Available since version 2.5.

**II.1.8.3 IRobotMobileCaseComponent****Class Hierarchy****C++**

```
interface IRobotMobileCaseComponent : IDispatch;
```

**C#**

```
public interface IRobotMobileCaseComponent;
```

**Visual Basic**

```
Public Interface IRobotMobileCaseComponent
```

**Description**

Simple component of a moving load case. This component corresponds to the simple case generated at the next point of the route. .

**Version**

Available since version 2.5.

**II.1.8.3.1 IRobotMobileCaseComponent Members**

The following tables list the members exposed by IRobotMobileCaseComponent.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Point (see page 331)	Number of the point on the route .
◆	Records (see page 332)	List of load records.

**II.1.8.3.2 IRobotMobileCaseComponent Fields**

The fields of the IRobotMobileCaseComponent class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Point (see page 331)	Number of the point on the route .
◆	Records (see page 332)	List of load records.

**II.1.8.3.2.1 Point****C++**

```
HRESULT get_Point(long* );
```

**C#**

```
public long Point { get; }
```

**Visual Basic**

```
Public ReadOnly Point As Long
```

**Description**

Number of the point on the route .

**Version**

Available since version 2.5.

**II.1.8.3.2.2 Records****C++**

```
HRESULT get_Records(IRobotLoadRecordMngr**);
```

**C#**

```
public IRobotLoadRecordMngr Records { get; }
```

**Visual Basic**

```
Public ReadOnly Records As IRobotLoadRecordMngr
```

**Description**

List of load records.

**Version**

Available since version 2.5.

**II.1.8.4 IRobotMobileCaseFlag****C++**

```
enum IRobotMobileCaseFlag;
```

**C#**

```
public enum IRobotMobileCaseFlag;
```

**Visual Basic**

```
Public Enum IRobotMobileCaseFlag
```

**Members**

<b>Members</b>	<b>Description</b>
I_MCF_MAIN = 0	Base moving load case. Available since version 2.5.
I_MCF_MAX = 1	Top envelope. Available since version 2.5.
I_MCF_MIN = 2	Bottom envelope. Available since version 2.5.

**Description**

Available types of a moving load case. .

**Version**

Available since version 2.5.

## II.1.8.5 IRobotMobileCaseRoute

### Class Hierarchy

#### C++

```
interface IRobotMobileCaseRoute : IDispatch;
```

#### C#

```
public interface IRobotMobileCaseRoute;
```

### Visual Basic

```
Public Interface IRobotMobileCaseRoute
```

### Description

Definition of vehicle route.

### Version

Available since version 3.

## II.1.8.5.1 IRobotMobileCaseRoute Members

The following tables list the members exposed by IRobotMobileCaseRoute.

### Public Fields

	Name	Description
❖	BeginningRouteLimit ( <a href="#">see page 334</a> )	Limit of vehicle position at the route beginning.
❖	EndRouteLimit ( <a href="#">see page 334</a> )	Limit of vehicle position at the route end.
❖	Geometry ( <a href="#">see page 334</a> )	Number of the object that determines route geometry .
❖	LoadDirection ( <a href="#">see page 335</a> )	Global direction of load application.
❖	Step ( <a href="#">see page 335</a> )	Step - the distance between two successive vehicle positions on a route, expressed in meters.
❖	Tolerance ( <a href="#">see page 335</a> )	Value determining conditions of generating the loads associated with the vehicle in the indicated structure .

### Public Methods

	Name	Description
❖	GetFactors ( <a href="#">see page 336</a> )	Function takes the vector of coefficients correcting loads for a route segment of the indicated number.
❖	SetFactors ( <a href="#">see page 336</a> )	Function updates the vector of coefficients correcting loads for a route segment of the indicated number and on the basis of the indicated interface .

## II.1.8.5.2 IRobotMobileCaseRoute Fields

The fields of the IRobotMobileCaseRoute class are listed here.

### Public Fields

	Name	Description
❖	BeginningRouteLimit ( <a href="#">see page 334</a> )	Limit of vehicle position at the route beginning.
❖	EndRouteLimit ( <a href="#">see page 334</a> )	Limit of vehicle position at the route end.
❖	Geometry ( <a href="#">see page 334</a> )	Number of the object that determines route geometry .
❖	LoadDirection ( <a href="#">see page 335</a> )	Global direction of load application.
❖	Step ( <a href="#">see page 335</a> )	Step - the distance between two successive vehicle positions on a route, expressed in meters.

	Tolerance (see page 335)	Value determining conditions of generating the loads associated with the vehicle in the indicated structure .
-----------------------------------------------------------------------------------	--------------------------	---------------------------------------------------------------------------------------------------------------

### II.1.8.5.2.1 BeginingRouteLimit

#### C++

```
HRESULT get_BeginingRouteLimit(VARIANT_BOOL* );
HRESULT put_BeginingRouteLimit(VARIANT_BOOL);
```

#### C#

```
public bool BeginingRouteLimit { get; set; }
```

#### Visual Basic

```
Public BeginingRouteLimit As Boolean
```

#### Description

Limit of vehicle position at the route beginning.

#### Version

Available since version 3.

### II.1.8.5.2.2 EndRouteLimit

#### C++

```
HRESULT get_EndRouteLimit(VARIANT_BOOL* );
HRESULT put_EndRouteLimit(VARIANT_BOOL);
```

#### C#

```
public bool EndRouteLimit { get; set; }
```

#### Visual Basic

```
Public EndRouteLimit As Boolean
```

#### Description

Limit of vehicle position at the route end.

#### Version

Available since version 3.

### II.1.8.5.2.3 Geometry

#### C++

```
HRESULT get_Geometry(long* );
HRESULT put_Geometry(long);
```

#### C#

```
public long Geometry { get; set; }
```

#### Visual Basic

```
Public Geometry As long
```

#### Description

Number of the object that determines route geometry .

#### Version

Available since version 3.

#### II.1.8.5.2.4 LoadDirection

##### C++

```
HRESULT get_LoadDirection(IRobotGeoPoint3D**);
```

##### C#

```
public IRobotGeoPoint3D LoadDirection { get; }
```

##### Visual Basic

```
Public ReadOnly LoadDirection As IRobotGeoPoint3D
```

##### Description

Global direction of load application.

##### Version

Available since version 3.

#### II.1.8.5.2.5 Step

##### C++

```
HRESULT get_Step(double*);  
HRESULT put_Step(double);
```

##### C#

```
public double Step { get; set; }
```

##### Visual Basic

```
Public Step As Double
```

##### Description

Step - the distance between two successive vehicle positions on a route, expressed in meters.

##### Version

Available since version 3.

#### II.1.8.5.2.6 Tolerance

##### C++

```
HRESULT get_Tolerance(double*);  
HRESULT put_Tolerance(double);
```

##### C#

```
public double Tolerance { get; set; }
```

##### Visual Basic

```
Public Tolerance As Double
```

##### Description

Value determining conditions of generating the loads associated with the vehicle in the indicated structure .

##### Version

Available since version 3.

#### II.1.8.5.3 IRobotMobileCaseRoute Methods

The methods of the IRobotMobileCaseRoute class are listed here.

## Public Methods

	Name	Description
💡	GetFactors (see page 336)	Function takes the vector of coefficients correcting loads for a route segment of the indicated number.
💡	SetFactors (see page 336)	Function updates the vector of coefficients correcting loads for a route segment of the indicated number and on the basis of the indicated interface .

### II.1.8.5.3.1 GetFactors

#### C++

```
HRESULT GetFactors(long _segment, IRobotMobileCaseSegmentFactors** ret);
```

#### C#

```
public IRobotMobileCaseSegmentFactors GetFactors(long _segment);
```

#### Visual Basic

```
Public Function GetFactors(_segment As long) As IRobotMobileCaseSegmentFactors
```

#### Description

Function takes the vector of coefficients correcting loads for a route segment of the indicated number.

#### Version

Available since version 3.

### II.1.8.5.3.2 SetFactors

#### C++

```
HRESULT SetFactors(long _segment, IRobotMobileCaseSegmentFactors _factors);
```

#### C#

```
public void SetFactors(long _segment, IRobotMobileCaseSegmentFactors _factors);
```

#### Visual Basic

```
Public Sub SetFactors(_segment As long, _factors As IRobotMobileCaseSegmentFactors)
```

#### Description

Function updates the vector of coefficients correcting loads for a route segment of the indicated number and on the basis of the indicated interface .

#### Version

Available since version 3.

### II.1.8.6 IRobotMobileCaseSegmentFactors

#### Class Hierarchy

#### C++

```
interface IRobotMobileCaseSegmentFactors : IDispatch;
```

#### C#

```
public interface IRobotMobileCaseSegmentFactors;
```

#### Visual Basic

```
Public Interface IRobotMobileCaseSegmentFactors
```

## Description

A set of coefficients assigned to a route segment, correcting vehicle loads. .

## Version

Available since version 3.

### II.1.8.6.1 IRobotMobileCaseSegmentFactors Members

The following tables list the members exposed by IRobotMobileCaseSegmentFactors.

#### Public Fields

	Name	Description
◆	Gamma ( <a href="#">see page 337</a> )	Angle determining vehicle rotation about the vertical axis.
◆	HL ( <a href="#">see page 338</a> )	Scaling factor for horizontal transversal force H (from left side).
◆	HR ( <a href="#">see page 338</a> )	Scaling factor for horizontal transversal force H (from right side).
◆	LL ( <a href="#">see page 338</a> )	Scaling factor for horizontal longitudinal force L (from left side).
◆	LR ( <a href="#">see page 339</a> )	Scaling factor for horizontal longitudinal force L (from right side).
◆	VL ( <a href="#">see page 339</a> )	Scaling factor for vertical force V (from left side).
◆	VR ( <a href="#">see page 339</a> )	Scaling factor for vertical force V (from right side).

### II.1.8.6.2 IRobotMobileCaseSegmentFactors Fields

The fields of the IRobotMobileCaseSegmentFactors class are listed here.

#### Public Fields

	Name	Description
◆	Gamma ( <a href="#">see page 337</a> )	Angle determining vehicle rotation about the vertical axis.
◆	HL ( <a href="#">see page 338</a> )	Scaling factor for horizontal transversal force H (from left side).
◆	HR ( <a href="#">see page 338</a> )	Scaling factor for horizontal transversal force H (from right side).
◆	LL ( <a href="#">see page 338</a> )	Scaling factor for horizontal longitudinal force L (from left side).
◆	LR ( <a href="#">see page 339</a> )	Scaling factor for horizontal longitudinal force L (from right side).
◆	VL ( <a href="#">see page 339</a> )	Scaling factor for vertical force V (from left side).
◆	VR ( <a href="#">see page 339</a> )	Scaling factor for vertical force V (from right side).

### II.1.8.6.2.1 Gamma

#### C++

```
HRESULT get_Gamma( double* );
HRESULT put_Gamma( double );
```

#### C#

```
public double Gamma { get; set; }
```

#### Visual Basic

```
Public Gamma As Double
```

## Description

Angle determining vehicle rotation about the vertical axis.

## Version

Available since version 3.

### II.1.8.6.2.2 HL

#### C++

```
HRESULT get_HL( double* );
HRESULT put_HL( double );
```

#### C#

```
public double HL { get; set; }
```

#### Visual Basic

```
Public HL As Double
```

#### Description

Scaling factor for horizontal transversal force H (from left side).

#### Version

Available since version 3.

### II.1.8.6.2.3 HR

#### C++

```
HRESULT get_HR( double* );
HRESULT put_HR( double );
```

#### C#

```
public double HR { get; set; }
```

#### Visual Basic

```
Public HR As Double
```

#### Description

Scaling factor for horizontal transversal force H (from right side).

#### Version

Available since version 3.

### II.1.8.6.2.4 LL

#### C++

```
HRESULT get_LL( double* );
HRESULT put_LL( double );
```

#### C#

```
public double LL { get; set; }
```

#### Visual Basic

```
Public LL As Double
```

#### Description

Scaling factor for horizontal longitudinal force L (from left side).

#### Version

Available since version 3.

### II.1.8.6.2.5 LR

#### C++

```
HRESULT get_LR( double* );
HRESULT put_LR( double );
```

#### C#

```
public double LR { get; set; }
```

#### Visual Basic

```
Public LR As Double
```

#### Description

Scaling factor for horizontal longitudinal force L (from right side).

#### Version

Available since version 3.

### II.1.8.6.2.6 VL

#### C++

```
HRESULT get_VL( double* );
HRESULT put_VL( double );
```

#### C#

```
public double VL { get; set; }
```

#### Visual Basic

```
Public VL As Double
```

#### Description

Scaling factor for vertical force V (from left side).

#### Version

Available since version 3.

### II.1.8.6.2.7 VR

#### C++

```
HRESULT get_VR( double* );
HRESULT put_VR( double );
```

#### C#

```
public double VR { get; set; }
```

#### Visual Basic

```
Public VR As Double
```

#### Description

Scaling factor for vertical force V (from right side).

#### Version

Available since version 3.

## II.1.8.7 IRobotMobileCaseApplicationPlaneType

### C++

```
enum IRobotMobileCaseApplicationPlaneType;
```

### C#

```
public enum IRobotMobileCaseApplicationPlaneType;
```

### Visual Basic

```
Public Enum IRobotMobileCaseApplicationPlaneType
```

### Members

Members	Description
I_MCPT_AUTOMATIC = 1	Application - automatic. Available since version 3.
I_MCPT_SELECTION = 2	Application - to selected bars. Available since version 3.

### Description

Type of application plane.

### Version

Available since version 3.

## II.1.8.8 IRobotVehicleData

### Class Hierarchy

### C++

```
interface IRobotVehicleData : IDispatch;
```

### C#

```
public interface IRobotVehicleData;
```

### Visual Basic

```
Public Interface IRobotVehicleData
```

### Description

Parameter set for vehicle definition.

### Version

Available since version 3.

## II.1.8.8.1 IRobotVehicleData Members

The following tables list the members exposed by IRobotVehicleData.

### Public Fields

	Name	Description
◆	b (↗ see page 341)	Vehicle width.
◆	d1 (↗ see page 341)	Distance between load and vehicle contour at the vehicle front .
◆	d2 (↗ see page 341)	Distance between load and vehicle contour at the vehicle back.
◆	Loads (↗ see page 342)	List of loads associated with the vehicle.

## Public Methods

	Name	Description
💡	LoadFromDBase (see page 342)	Function reads vehicle parameters from the database.
💡	StoreToDBase (see page 343)	Function saves vehicle parameters to the database.

### II.1.8.8.2 IRobotVehicleData Fields

The fields of the IRobotVehicleData class are listed here.

#### Public Fields

	Name	Description
💡	b (see page 341)	Vehicle width.
💡	d1 (see page 341)	Distance between load and vehicle contour at the vehicle front .
💡	d2 (see page 341)	Distance between load and vehicle contour at the vehicle back.
💡	Loads (see page 342)	List of loads associated with the vehicle.

#### II.1.8.8.2.1 b

##### C++

```
HRESULT get_b( double* );
HRESULT put_b( double );
```

##### C#

```
public double b { get; set; }
```

##### Visual Basic

```
Public b As Double
```

##### Description

Vehicle width.

##### Version

Available since version 3.

#### II.1.8.8.2.2 d1

##### C++

```
HRESULT get_d1( double* );
HRESULT put_d1( double );
```

##### C#

```
public double d1 { get; set; }
```

##### Visual Basic

```
Public d1 As Double
```

##### Description

Distance between load and vehicle contour at the vehicle front .

##### Version

Available since version 3.

#### II.1.8.8.2.3 d2

##### C++

```
HRESULT get_d2( double* );
```

```
HRESULT put_d2( double );
```

**C#**

```
public double d2 { get; set; }
```

**Visual Basic**

```
Public d2 As Double
```

**Description**

Distance between load and vehicle contour at the vehicle back.

**Version**

Available since version 3.

**II.1.8.2.4 Loads****C++**

```
HRESULT get_Loads( IRobotVehicleLoadMngr** );
```

**C#**

```
public IRobotVehicleLoadMngr Loads { get; }
```

**Visual Basic**

```
Public ReadOnly Loads As IRobotVehicleLoadMngr
```

**Description**

List of loads associated with the vehicle.

**Version**

Available since version 3.

**II.1.8.3 IRobotVehicleData Methods**

The methods of the IRobotVehicleData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	LoadFromDBase (see page 342)	Function reads vehicle parameters from the database.
💡	StoreToDBase (see page 343)	Function saves vehicle parameters to the database.

**II.1.8.3.1 LoadFromDBase****C++**

```
HRESULT LoadFromDBase(BSTR _vehicle_name, BSTR _db_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromDBase(String _vehicle_name, String _db_name);
```

**Visual Basic**

```
Public Function LoadFromDBase(_vehicle_name As String, _db_name As String) As Boolean
```

**Description**

Function reads vehicle parameters from the database.

**Version**

Available since version 3.

### II.1.8.8.3.2 StoreToDBase

#### C++

```
HRESULT StoreToDBase(BSTR _vehicle_name, BSTR _db_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool StoreToDBase(String _vehicle_name, String _db_name);
```

#### Visual Basic

```
Public Function StoreToDBase(_vehicle_name As String, _db_name As String) As Boolean
```

#### Description

Function saves vehicle parameters to the database.

#### Version

Available since version 3.

### II.1.8.9 IRobotVehicleLoadMngr

#### Class Hierarchy

#### C++

```
interface IRobotVehicleLoadMngr : IDispatch;
```

#### C#

```
public interface IRobotVehicleLoadMngr;
```

#### Visual Basic

```
Public Interface IRobotVehicleLoadMngr
```

#### Description

List of loads that define a vehicle.

#### Version

Available since version 3.

### II.1.8.9.1 IRobotVehicleLoadMngr Members

The following tables list the members exposed by IRobotVehicleLoadMngr.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 344</a> )	Number of records on the list.

#### Public Methods

	Name	Description
◆	Delete ( <a href="#">see page 344</a> )	Function removes a record from a list.
◆	Get ( <a href="#">see page 344</a> )	Function takes a record of the indicated index.
◆	New ( <a href="#">see page 345</a> )	Function adds a new load record at the end of the list.

### II.1.8.9.2 IRobotVehicleLoadMngr Fields

The fields of the IRobotVehicleLoadMngr class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 344)	Number of records on the list.

### II.1.8.9.2.1 Count

#### C++

```
HRESULT get_Count( long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of records on the list.

#### Version

Available since version 3.

### II.1.8.9.3 IRobotVehicleLoadMngr Methods

The methods of the IRobotVehicleLoadMngr class are listed here.

#### Public Methods

	Name	Description
◆	Delete (see page 344)	Function removes a record from a list.
◆	Get (see page 344)	Function takes a record of the indicated index.
◆	New (see page 345)	Function adds a new load record at the end of the list.

### II.1.8.9.3.1 Delete

#### C++

```
HRESULT Delete(long _idx);
```

#### C#

```
public void Delete(long _idx);
```

#### Visual Basic

```
Public Sub Delete(_idx As long)
```

#### Description

Function removes a record from a list.

#### Version

Available since version 3.

### II.1.8.9.3.2 Get

#### C++

```
HRESULT Get(long _idx, IRobotVehicleLoad** ret);
```

#### C#

```
public IRobotVehicleLoad Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotVehicleLoad
```

**Description**

Function takes a record of the indicated index.

**Version**

Available since version 3.

**II.1.8.9.3.3 New****C++**

```
HRESULT New(IRobotVehicleLoad** ret);
```

**C#**

```
public IRobotVehicleLoad New();
```

**Visual Basic**

```
Public Function New() As IRobotVehicleLoad
```

**Description**

Function adds a new load record at the end of the list.

**Version**

Available since version 3.

**II.1.8.10 IRobotVehicleLoad****Class Hierarchy****C++**

```
interface IRobotVehicleLoad : IDispatch;
```

**C#**

```
public interface IRobotVehicleLoad;
```

**Visual Basic**

```
Public Interface IRobotVehicleLoad
```

**Description**

Record describing a single load associated with the vehicle. .

**Version**

Available since version 3.

**II.1.8.10.1 IRobotVehicleLoad Members**

The following tables list the members exposed by IRobotVehicleLoad.

**Public Fields**

	Name	Description
◆	DX (see page 346)	For a linear load, it is the length of a segment on which the load is acting (along the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (along the vehicle axis); for concentrated force loads, the parameter is inactive.

◆	DY ( <a href="#">see page 347</a> )	For a linear load, it is the length of a segment on which the load is acting (perpendicular to the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (perpendicular to the vehicle axis); for concentrated force loads, the parameter is inactive .
◆	F ( <a href="#">see page 347</a> )	Depending on a load type, it is: value of a concentrated force, value of a linear load or value of a planar load .
◆	S ( <a href="#">see page 347</a> )	Depending on a load type, it is: value of force spacing, value of linear load spacing (only in the direction of Y axis) or value of planar load spacing (only in the direction of Y axis).
◆	Type ( <a href="#">see page 348</a> )	Load type.
◆	X ( <a href="#">see page 348</a> )	Coordinate value for the line along which the force is applied along the vehicle axis .

## II.1.8.10.2 IRobotVehicleLoad Fields

The fields of the IRobotVehicleLoad class are listed here.

### Public Fields

	Name	Description
◆	DX ( <a href="#">see page 346</a> )	For a linear load, it is the length of a segment on which the load is acting (along the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (along the vehicle axis); for concentrated force loads, the parameter is inactive.
◆	DY ( <a href="#">see page 347</a> )	For a linear load, it is the length of a segment on which the load is acting (perpendicular to the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (perpendicular to the vehicle axis); for concentrated force loads, the parameter is inactive .
◆	F ( <a href="#">see page 347</a> )	Depending on a load type, it is: value of a concentrated force, value of a linear load or value of a planar load .
◆	S ( <a href="#">see page 347</a> )	Depending on a load type, it is: value of force spacing, value of linear load spacing (only in the direction of Y axis) or value of planar load spacing (only in the direction of Y axis).
◆	Type ( <a href="#">see page 348</a> )	Load type.
◆	X ( <a href="#">see page 348</a> )	Coordinate value for the line along which the force is applied along the vehicle axis .

## II.1.8.10.2.1 DX

### C++

```
HRESULT get_DX( double* );
HRESULT put_DX( double );
```

### C#

```
public double DX { get; set; }
```

### Visual Basic

```
Public DX As Double
```

### Description

For a linear load, it is the length of a segment on which the load is acting (along the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (along the vehicle axis); for concentrated force loads, the parameter is inactive.

### Version

Available since version 3.

### II.1.8.10.2.2 DY

#### C++

```
HRESULT get_DY( double* );
HRESULT put_DY( double );
```

#### C#

```
public double DY { get; set; }
```

#### Visual Basic

```
Public DY As Double
```

#### Description

For a linear load, it is the length of a segment on which the load is acting (perpendicular to the vehicle axis); for a planar load, it is the length of a rectangle side on which the load is acting (perpendicular to the vehicle axis); for concentrated force loads, the parameter is inactive .

#### Version

Available since version 3.

### II.1.8.10.2.3 F

#### C++

```
HRESULT get_F( double* );
HRESULT put_F( double );
```

#### C#

```
public double F { get; set; }
```

#### Visual Basic

```
Public F As Double
```

#### Description

Depending on a load type, it is: value of a concentrated force, value of a linear load or value of a planar load .

#### Version

Available since version 3.

### II.1.8.10.2.4 S

#### C++

```
HRESULT get_S( double* );
HRESULT put_S( double );
```

#### C#

```
public double S { get; set; }
```

#### Visual Basic

```
Public S As Double
```

#### Description

Depending on a load type, it is: value of force spacing, value of linear load spacing (only in the direction of Y axis) or value of planar load spacing (only in the direction of Y axis).

#### Version

Available since version 3.

### II.1.8.10.2.5 Type

#### C++

```
HRESULT get_Type(IRobotVehicleLoadType* );
HRESULT put_Type(IRobotVehicleLoadType);
```

#### C#

```
public IRobotVehicleLoadType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotVehicleLoadType
```

#### Description

Load type.

#### Version

Available since version 3.

### II.1.8.10.2.6 X

#### C++

```
HRESULT get_X( double* );
HRESULT put_X( double );
```

#### C#

```
public double x { get; set; }
```

#### Visual Basic

```
Public x As double
```

#### Description

Coordinate value for the line along which the force is applied along the vehicle axis .

#### Version

Available since version 3.

### II.1.8.11 IRobotVehicleLoadType

#### C++

```
enum IRobotVehicleLoadType;
```

#### C#

```
public enum IRobotVehicleLoadType;
```

#### Visual Basic

```
Public Enum IRobotVehicleLoadType
```

#### Members

Members	Description
I_VLT_CONCENTRATED = 1	Concentrated force. Available since version 3.
I_VLT_LINEAR = 2	Linear load. Available since version 3.
I_VLT_SURFACE = 3	Surface load. Available since version 3.

**Description**

Load types.

**Version**

Available since version 3.

**II.1.8.12 IRobotVehicleDatabaseList****Class Hierarchy****C++**

```
interface IRobotVehicleDatabaseList : IDispatch;
```

**C#**

```
public interface IRobotVehicleDatabaseList;
```

**Visual Basic**

```
Public Interface IRobotVehicleDatabaseList
```

**Description**

Interface describing the list of vehicle databases.

**Version**

Available since version 3.

**II.1.8.12.1 IRobotVehicleDatabaseList Members**

The following tables list the members exposed by IRobotVehicleDatabaseList.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 350</a> )	Number of vehicle databases on the list.
◆	Default ( <a href="#">see page 350</a> )	Index of default vehicle database.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add ( <a href="#">see page 351</a> )	Function adds vehicle database of the indicated name at the end of the list. Status of the action is returned.
◆	AddFromFile ( <a href="#">see page 351</a> )	Function adds the vehicle database saved in the indicated file at the end of the list. Status of the operation execution is returned.
◆	Create ( <a href="#">see page 351</a> )	Function creates a new vehicle database.
◆	Find ( <a href="#">see page 352</a> )	Function searches the list seeking for the database of the indicated name. If there is no specified database on the list, zero value is returned, if there is, function returns the database index from the list. .
◆	Get ( <a href="#">see page 352</a> )	Function returns the name of the vehicle database located at the indicated position on the list.
◆	GetDatabase ( <a href="#">see page 352</a> )	Function returns the interface for the vehicle database with the index indicated on the list.
◆	Remove ( <a href="#">see page 353</a> )	Function deletes the database of the indicated index from the list. .

**II.1.8.12.2 IRobotVehicleDatabaseList Fields**

The fields of the IRobotVehicleDatabaseList class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 350)	Number of vehicle databases on the list.
◆	Default (see page 350)	Index of default vehicle database.

### II.1.8.12.2.1 Count

#### C++

```
HRESULT get_Count(int *);
```

#### C#

```
public int Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As int
```

#### Description

Number of vehicle databases on the list.

#### Version

Available since version 3.

### II.1.8.12.2.2 Default

#### C++

```
HRESULT get_Default(int *);  
HRESULT put_Default(int);
```

#### C#

```
public int Default { get; set; }
```

#### Visual Basic

```
Public Default As int
```

#### Description

Index of default vehicle database.

#### Version

Available since version 3.

### II.1.8.12.3 IRobotVehicleDatabaseList Methods

The methods of the IRobotVehicleDatabaseList class are listed here.

#### Public Methods

	Name	Description
◆	Add (see page 351)	Function adds vehicle database of the indicated name at the end of the list. Status of the action is returned.
◆	AddFromFile (see page 351)	Function adds the vehicle database saved in the indicated file at the end of the list. Status of the operation execution is returned.
◆	Create (see page 351)	Function creates a new vehicle database.
◆	Find (see page 352)	Function searches the list seeking for the database of the indicated name. If there is no specified database on the list, zero value is returned, if there is, function returns the database index from the list. .
◆	Get (see page 352)	Function returns the name of the vehicle database located at the indicated position on the list.

	GetDatabase (see page 352)	Function returns the interface for the vehicle database with the index indicated on the list.
	Remove (see page 353)	Function deletes the database of the indicated index from the list. .

### II.1.8.12.3.1 Add

**C++**

```
HRESULT Add(BSTR _db_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Add(String _db_name);
```

**Visual Basic**

```
Public Function Add(_db_name As String) As Boolean
```

**Description**

Function adds vehicle database of the indicated name at the end of the list. Status of the action is returned.

**Version**

Available since version 3.

### II.1.8.12.3.2 AddFromFile

**C++**

```
HRESULT AddFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool AddFromFile(String _file_path);
```

**Visual Basic**

```
Public Function AddFromFile(_file_path As String) As Boolean
```

**Description**

Function adds the vehicle database saved in the indicated file at the end of the list. Status of the operation execution is returned.

**Version**

Available since version 3.

### II.1.8.12.3.3 Create

**C++**

```
HRESULT Create(BSTR _db_name, BSTR _long_db_name, BSTR _description);
```

**C#**

```
public void Create(String _db_name, String _long_db_name, String _description);
```

**Visual Basic**

```
Public Sub Create(_db_name As String, _long_db_name As String, _description As String)
```

**Description**

Function creates a new vehicle database.

**Version**

Available since version 3.

#### II.1.8.12.3.4 Find

**C++**

```
HRESULT Find(BSTR _db_name, int* ret);
```

**C#**

```
public int Find(String _db_name);
```

**Visual Basic**

```
Public Function Find(_db_name As String) As int
```

**Description**

Function searches the list seeking for the database of the indicated name. If there is no specified database on the list, zero value is returned, if there is, function returns the database index from the list. .

**Version**

Available since version 3.

#### II.1.8.12.3.5 Get

**C++**

```
HRESULT Get(int _idx, String* ret);
```

**C#**

```
public String Get(int _idx);
```

**Visual Basic**

```
Public Function Get(_idx As int) As String
```

**Description**

Function returns the name of the vehicle database located at the indicated position on the list.

**Version**

Available since version 3.

#### II.1.8.12.3.6 GetDatabase

**C++**

```
HRESULT GetDatabase(int _idx, IRobotVehicleDatabase** ret);
```

**C#**

```
public IRobotVehicleDatabase GetDatabase(int _idx);
```

**Visual Basic**

```
Public Function GetDatabase(_idx As int) As IRobotVehicleDatabase
```

**Description**

Function returns the interface for the vehicle database with the index indicated on the list.

**Version**

Available since version 3.

### II.1.8.12.3.7 Remove

**C++**

```
HRESULT Remove(int _idx);
```

**C#**

```
public void Remove(int _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As int)
```

**Description**

Function deletes the database of the indicated index from the list. .

**Version**

Available since version 3.

### II.1.8.13 IRobotVehicleDatabase

**Class Hierarchy**

**C++**

```
interface IRobotVehicleDatabase : IDispatch;
```

**C#**

```
public interface IRobotVehicleDatabase;
```

**Visual Basic**

```
Public Interface IRobotVehicleDatabase
```

**Description**

Interface for vehicle database support.

**Version**

Available since version 3.

#### II.1.8.13.1 IRobotVehicleDatabase Members

The following tables list the members exposed by IRobotVehicleDatabase.

**Public Fields**

	Name	Description
◆	Description (↗ see page 354)	
◆	LongName (↗ see page 354)	Additional description of a vehicle database.
◆	Name (↗ see page 354)	Name identifying the vehicle database.

**Public Methods**

	Name	Description
♫	GetAll (↗ see page 355)	Function returns a table containing names of all the vehicles from the database. .
♫	Load (↗ see page 355)	Function reads vehicles from the vehicle database of the indicated name. If the specified database does not exist or an error occurs during read-out, zero value is returned. .

	LoadFromFile ( <a href="#">see page 356</a> )	Function reads vehicles from the database saved in the indicated file. If the specified path is incorrect or other reading error occurs, the function returns zero value. .
--	-----------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## II.1.8.13.2 IRobotVehicleDatabase Fields

The fields of the IRobotVehicleDatabase class are listed here.

### Public Fields

	Name	Description
	Description ( <a href="#">see page 354</a> )	
	LongName ( <a href="#">see page 354</a> )	Additional description of a vehicle database.
	Name ( <a href="#">see page 354</a> )	Name identifying the vehicle database.

### II.1.8.13.2.1 Description

#### C++

```
HRESULT get_Description(BSTR* );
```

#### C#

```
public String Description { get; }
```

#### Visual Basic

```
Public ReadOnly Description As String
```

#### Version

Available since version 3.

### II.1.8.13.2.2 LongName

#### C++

```
HRESULT get_LongName(BSTR* );
```

#### C#

```
public String LongName { get; }
```

#### Visual Basic

```
Public ReadOnly LongName As String
```

#### Description

Additional description of a vehicle database.

#### Version

Available since version 3.

### II.1.8.13.2.3 Name

#### C++

```
HRESULT get_Name(BSTR* );
```

#### C#

```
public String Name { get; }
```

#### Visual Basic

```
Public ReadOnly Name As String
```

**Description**

Name identifying the vehicle database.

**Version**

Available since version 3.

**II.1.8.13.3 IRobotVehicleDatabase Methods**

The methods of the IRobotVehicleDatabase class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetAll ( [ see page 355)	Function returns a table containing names of all the vehicles from the database. .
💡	Load ( [ see page 355)	Function reads vehicles from the vehicle database of the indicated name. If the specified database does not exist or an error occurs during read-out, zero value is returned. .
💡	LoadFromFile ( [ see page 356)	Function reads vehicles from the database saved in the indicated file. If the specified path is incorrect or other reading error occurs, the function returns zero value. .

**II.1.8.13.3.1 GetAll****C++**

```
HRESULT GetAll(IRobotNamesArray* ret);
```

**C#**

```
public IRobotNamesArray GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotNamesArray
```

**Description**

Function returns a table containing names of all the vehicles from the database. .

**Version**

Available since version 3.

**II.1.8.13.3.2 Load****C++**

```
HRESULT Load(BSTR _db_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Load(String _db_name);
```

**Visual Basic**

```
Public Function Load(_db_name As String) As Boolean
```

**Description**

Function reads vehicles from the vehicle database of the indicated name. If the specified database does not exist or an error occurs during read-out, zero value is returned. .

**Version**

Available since version 3.

### II.1.8.13.3.3 LoadFromFile

#### C++

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromFile(String _file_path);
```

#### Visual Basic

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

#### Description

Function reads vehicles from the database saved in the indicated file. If the specified path is incorrect or other reading error occurs, the function returns zero value..

#### Version

Available since version 3.

## II.1.9 Buckling analysis parameters

Available since version 3.

#### Enumerations

	Name	Description
	IRobotBucklingAnalysisMethod (see page 360)	Methods of eigenproblem solution for buckling analysis.

#### Interfaces

	Name	Description
	IRobotBucklingAnalysisParams (see page 356)	Buckling analysis parameters.

### II.1.9.1 IRobotBucklingAnalysisParams

#### Class Hierarchy

#### C++

```
interface IRobotBucklingAnalysisParams : IDispatch;
```

#### C#

```
public interface IRobotBucklingAnalysisParams;
```

#### Visual Basic

```
Public Interface IRobotBucklingAnalysisParams
```

#### Description

Buckling analysis parameters.

#### Version

Available since version 3.

### II.1.9.1.1 IRobotBucklingAnalysisParams Members

The following tables list the members exposed by IRobotBucklingAnalysisParams.

## Public Fields

	Name	Description
◆	Increment ( <a href="#">see page 357</a> )	Number of load increments.
◆	IsNonlinear ( <a href="#">see page 358</a> )	
◆	IterationsCount ( <a href="#">see page 358</a> )	Maximum number of iterations.
◆	Method ( <a href="#">see page 358</a> )	Method of eigenproblem solution.
◆	ModesCount ( <a href="#">see page 358</a> )	Number of buckling modes.
◆	NonlinearParams ( <a href="#">see page 359</a> )	Parameters of non-linear analysis (available if IsNonlinear ( <a href="#">see page 358</a> ) flag is set).
◆	Shift ( <a href="#">see page 359</a> )	Shift value.
◆	SturmVerification ( <a href="#">see page 359</a> )	Flag switching on/off the algorithm that enables searching disregarded structure modes.
◆	Tolerance ( <a href="#">see page 360</a> )	Value of the error tolerance that is to be obtained during structure iteration analysis.

### II.1.9.1.2 IRobotBucklingAnalysisParams Fields

The fields of the IRobotBucklingAnalysisParams class are listed here.

## Public Fields

	Name	Description
◆	Increment ( <a href="#">see page 357</a> )	Number of load increments.
◆	IsNonlinear ( <a href="#">see page 358</a> )	
◆	IterationsCount ( <a href="#">see page 358</a> )	Maximum number of iterations.
◆	Method ( <a href="#">see page 358</a> )	Method of eigenproblem solution.
◆	ModesCount ( <a href="#">see page 358</a> )	Number of buckling modes.
◆	NonlinearParams ( <a href="#">see page 359</a> )	Parameters of non-linear analysis (available if IsNonlinear ( <a href="#">see page 358</a> ) flag is set).
◆	Shift ( <a href="#">see page 359</a> )	Shift value.
◆	SturmVerification ( <a href="#">see page 359</a> )	Flag switching on/off the algorithm that enables searching disregarded structure modes.
◆	Tolerance ( <a href="#">see page 360</a> )	Value of the error tolerance that is to be obtained during structure iteration analysis.

### II.1.9.1.2.1 Increment

#### C++

```
HRESULT get_Increment(long* );
HRESULT put_Increment(long);
```

#### C#

```
public long Increment { get; set; }
```

#### Visual Basic

```
Public Increment As long
```

#### Description

Number of load increments.

#### Version

Available since version 3.

### II.1.9.1.2.2 IsNonlinear

#### C++

```
HRESULT get_IsNonlinear(VARIANT_BOOL*);
```

#### C#

```
public bool IsNonlinear { get; }
```

#### Visual Basic

```
Public ReadOnly IsNonlinear As Boolean
```

#### Version

Available since version 3.

### II.1.9.1.2.3 IterationsCount

#### C++

```
HRESULT get_IterationsCount(long*);  
HRESULT put_IterationsCount(long);
```

#### C#

```
public long IterationsCount { get; set; }
```

#### Visual Basic

```
Public IterationsCount As Long
```

#### Description

Maximum number of iterations.

#### Version

Available since version 3.

### II.1.9.1.2.4 Method

#### C++

```
HRESULT get_Method(IRobotBucklingAnalysisMethod*);  
HRESULT put_Method(IRobotBucklingAnalysisMethod);
```

#### C#

```
public IRobotBucklingAnalysisMethod Method { get; set; }
```

#### Visual Basic

```
Public Method As IRobotBucklingAnalysisMethod
```

#### Description

Method of eigenproblem solution.

#### Version

Available since version 3.

### II.1.9.1.2.5 ModesCount

#### C++

```
HRESULT get_ModesCount(long*);  
HRESULT put_ModesCount(long);
```

**C#**

```
public long ModesCount { get; set; }
```

**Visual Basic**

```
Public ModesCount As Long
```

**Description**

Number of buckling modes.

**Version**

Available since version 3.

**II.1.9.1.2.6 NonlinearParams****C++**

```
HRESULT get_NonlinearParams(IRobotNonlinearAnalysisParams**);
```

**C#**

```
public IRobotNonlinearAnalysisParams NonlinearParams { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearParams As IRobotNonlinearAnalysisParams
```

**Description**

Parameters of non-linear analysis (available if IsNonlinear (see page 358) flag is set).

**Version**

Available since version 3.

**II.1.9.1.2.7 Shift****C++**

```
HRESULT get_Shift(double*);  
HRESULT put_Shift(double);
```

**C#**

```
public double Shift { get; set; }
```

**Visual Basic**

```
Public Shift As Double
```

**Description**

Shift value.

**Version**

Available since version 3.

**II.1.9.1.2.8 SturmVerification****C++**

```
HRESULT get_SturmVerification(VARIANT_BOOL*);  
HRESULT put_SturmVerification(VARIANT_BOOL);
```

**C#**

```
public bool SturmVerification { get; set; }
```

**Visual Basic**

```
Public SturmVerification As Boolean
```

**Description**

Flag switching on/off the algorithm that enables searching disregarded structure modes.

**Version**

Available since version 3.

**II.1.9.1.2.9 Tolerance****C++**

```
HRESULT get_Tolerance(double* );
HRESULT put_Tolerance(double);
```

**C#**

```
public double Tolerance { get; set; }
```

**Visual Basic**

```
Public Tolerance As Double
```

**Description**

Value of the error tolerance that is to be obtained during structure iteration analysis.

**Version**

Available since version 3.

**II.1.9.2 IRobotBucklingAnalysisMethod****C++**

```
enum IRobotBucklingAnalysisMethod;
```

**C#**

```
public enum IRobotBucklingAnalysisMethod;
```

**Visual Basic**

```
Public Enum IRobotBucklingAnalysisMethod
```

**Members**

Members	Description
I_BAM_SUBSPACE_ITERATION = 0	Subspace iteration. Available since version 3.
I_BAM_BLOCK_SUBSPACE_ITERATION = 1	Block subspace iteration. Available since version 3.

**Description**

Methods of eigenproblem solution for buckling analysis.

**Version**

Available since version 3.

**II.1.10 Time history analysis parameters**

Available since version 3.

## Enumerations

	Name	Description
	IRobotTimeHistoryAnalysisMethod ( <a href="#">see page 366</a> )	Available methods of time history analysis. .

## Interfaces

	Name	Description
	IRobotTimeHistoryAnalysisParams ( <a href="#">see page 361</a> )	Parameters of time history analysis.
	IRobotTimeHistoryNewmarkParams ( <a href="#">see page 366</a> )	Parameters of Newmark method for time history analysis. .
	IRobotTimeHistoryModalDecompositionParams ( <a href="#">see page 368</a> )	Parameters of modal decomposition method. It is a list of pairs [mode number, damping value]. For each mode, maximally one such a pair may exist. Additionally, the pairs are indexed with numbers from 1 to Count ( <a href="#">see page 369</a> ). .
	IRobotTimeHistoryPointsCollection ( <a href="#">see page 371</a> )	Collection of points defining a time function for the time history analysis. .
	IRobotTimeHistoryFunctionList ( <a href="#">see page 375</a> )	List of defined time functions for time history analysis. .
	IRobotTimeHistoryNonlinearParams ( <a href="#">see page 380</a> )	Non-linear parameters of time history analysis.
	IRobotTimeHistoryHHTParams ( <a href="#">see page 382</a> )	Parameters of HHT (Hilber-Hughes-Taylor) method for time history analysis.
	IRobotTimeHistoryNewmarkAccelParams ( <a href="#">see page 385</a> )	Parameters of Newmark method (acceleration) for time history analysis.

### II.1.10.1 IRobotTimeHistoryAnalysisParams

#### Class Hierarchy

#### C++

```
interface IRobotTimeHistoryAnalysisParams : IDispatch;
```

#### C#

```
public interface IRobotTimeHistoryAnalysisParams;
```

#### Visual Basic

```
Public Interface IRobotTimeHistoryAnalysisParams
```

#### Description

Parameters of time history analysis.

#### Version

Available since version 3.

### II.1.10.1.1 IRobotTimeHistoryAnalysisParams Members

The following tables list the members exposed by IRobotTimeHistoryAnalysisParams.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 362</a> )	Number of defined time history analyses.
	Division ( <a href="#">see page 362</a> )	Number of time division for the result time step.
	End ( <a href="#">see page 363</a> )	Final value of the time variable for which the analysis is performed .
	InitialCase ( <a href="#">see page 363</a> )	Number of the load case for which results are to be taken as initial.

❖	Method ( <a href="#">see page 363</a> )	Time history analysis method.
❖	MethodParams ( <a href="#">see page 364</a> )	Parameters of selected method.
❖	TimeStep ( <a href="#">see page 364</a> )	Step of the time variable for which results are being saved.

**Public Methods**

	Name	Description
❖	Delete ( <a href="#">see page 365</a> )	Function deletes time history analysis definition of the index specified on the list.
❖	Find ( <a href="#">see page 365</a> )	Function returns index of the time history analysis definition for the specified static case (or mass). If for the indicated case the time history analysis has not been defined, function returns zero value. .
❖	Get ( <a href="#">see page 365</a> )	Function returns the time history analysis definition of the specified index. .
❖	Set ( <a href="#">see page 366</a> )	Function defines the time history analysis for the specified static case or mass.

**II.1.10.1.2 IRobotTimeHistoryAnalysisParams Fields**

The fields of the IRobotTimeHistoryAnalysisParams class are listed here.

**Public Fields**

	Name	Description
❖	Count ( <a href="#">see page 362</a> )	Number of defined time history analyses.
❖	Division ( <a href="#">see page 362</a> )	Number of time division for the result time step.
❖	End ( <a href="#">see page 363</a> )	Final value of the time variable for which the analysis is performed .
❖	InitialCase ( <a href="#">see page 363</a> )	Number of the load case for which results are to be taken as initial.
❖	Method ( <a href="#">see page 363</a> )	Time history analysis method.
❖	MethodParams ( <a href="#">see page 364</a> )	Parameters of selected method.
❖	TimeStep ( <a href="#">see page 364</a> )	Step of the time variable for which results are being saved.

**II.1.10.1.2.1 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of defined time history analyses.

**Version**

Available since version 3.

**II.1.10.1.2.2 Division****C++**

```
HRESULT get_Division(long*);  
HRESULT put_Division(long);
```

**C#**

```
public long Division { get; set; }
```

**Visual Basic**

```
Public Division As long
```

**Description**

Number of time division for the result time step.

**Version**

Available since version 3.

**II.1.10.1.2.3 End****C++**

```
HRESULT get_End(double* );
HRESULT put_End(double);
```

**C#**

```
public double End { get; set; }
```

**Visual Basic**

```
Public End As double
```

**Description**

Final value of the time variable for which the analysis is performed .

**Version**

Available since version 3.

**II.1.10.1.2.4 InitialCase****C++**

```
HRESULT get_InitialCase(long* );
HRESULT put_InitialCase(long);
```

**C#**

```
public long InitialCase { get; set; }
```

**Visual Basic**

```
Public InitialCase As long
```

**Description**

Number of the load case for which results are to be taken as initial.

**Version**

Available since version 11.

**II.1.10.1.2.5 Method****C++**

```
HRESULT get_Method(IRobotTimeHistoryAnalysisMethod* );
HRESULT put_Method(IRobotTimeHistoryAnalysisMethod);
```

**C#**

```
public IRobotTimeHistoryAnalysisMethod Method { get; set; }
```

**Visual Basic**

```
Public Method As IRobotTimeHistoryAnalysisMethod
```

**Description**

Time history analysis method.

**Version**

Available since version 3.

**II.1.10.1.2.6 MethodParams****C++**

```
HRESULT get_MethodParams(IDispatch*);
```

**C#**

```
public IDispatch MethodParams { get; }
```

**Visual Basic**

```
Public ReadOnly MethodParams As IDispatch
```

**Description**

Parameters of selected method.

**Version**

Available since version 3.

**II.1.10.1.2.7 TimeStep****C++**

```
HRESULT get_TimeStep(double*);  
HRESULT put_TimeStep(double);
```

**C#**

```
public double TimeStep { get; set; }
```

**Visual Basic**

```
Public TimeStep As Double
```

**Description**

Step of the time variable for which results are being saved.

**Version**

Available since version 3.

**II.1.10.1.3 IRobotTimeHistoryAnalysisParams Methods**

The methods of the IRobotTimeHistoryAnalysisParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Delete ( see page 365)	Function deletes time history analysis definition of the index specified on the list.
≡	Find ( see page 365)	Function returns index of the time history analysis definition for the specified static case (or mass). If for the indicated case the time history analysis has not been defined, function returns zero value. .
≡	Get ( see page 365)	Function returns the time history analysis definition of the specified index. .
≡	Set ( see page 366)	Function defines the time history analysis for the specified static case or mass.

### II.1.10.1.3.1 Delete

#### C++

```
HRESULT Delete(long _idx);
```

#### C#

```
public void Delete(long _idx);
```

#### Visual Basic

```
Public Sub Delete(_idx As long)
```

#### Description

Function deletes time history analysis definition of the index specified on the list.

#### Version

Available since version 3.

### II.1.10.1.3.2 Find

#### C++

```
HRESULT Find(long _case_num, long* ret);
```

#### C#

```
public long Find(long _case_num);
```

#### Visual Basic

```
Public Function Find(_case_num As long) As long
```

#### Description

Function returns index of the time history analysis definition for the specified static case (or mass). If for the indicated case the time history analysis has not been defined, function returns zero value. .

#### Version

Available since version 3.

### II.1.10.1.3.3 Get

#### C++

```
HRESULT Get(long _idx, long* _case_num, BSTR _time_fun, double* _coeff, double* _shift);
```

#### C#

```
public void Get(long _idx, long* _case_num, String _time_fun, double* _coeff, double* _shift);
```

#### Visual Basic

```
Public Sub Get(_idx As long, ByRef _case_num As long*, ByRef _time_fun As String, ByRef _coeff As double*, ByRef _shift As double*)
```

#### Description

Function returns the time history analysis definition of the specified index. .

#### Version

Available since version 3.

#### II.1.10.1.3.4 Set

##### C++

```
HRESULT Set(long _case_num, BSTR _time_fun, double _coeff, double _shift);
```

##### C#

```
public void Set(long _case_num, String _time_fun, double _coeff, double _shift);
```

##### Visual Basic

```
Public Sub Set(_case_num As long, _time_fun As String, _coeff As double, _shift As double)
```

##### Description

Function defines the time history analysis for the specified static case or mass.

##### Version

Available since version 3.

#### II.1.10.2 IRobotTimeHistoryAnalysisMethod

##### C++

```
enum IRobotTimeHistoryAnalysisMethod;
```

##### C#

```
public enum IRobotTimeHistoryAnalysisMethod;
```

##### Visual Basic

```
Public Enum IRobotTimeHistoryAnalysisMethod
```

##### Members

Members	Description
I_THAM_NEWMARK = 0	Available since version 3.
I_THAM_MODAL_DECOMPOSITION = 1	Available since version 3.
I_THAM_HHT = 3	HHT (Hilber-Hughes-Taylor) method. Available since version 11.
I_THAM_NEWMARK_ACCELERATION = 4	Newmark method (acceleration). Available since version 11.

##### Description

Available methods of time history analysis. .

##### Version

Available since version 3.

#### II.1.10.3 IRobotTimeHistoryNewmarkParams

##### Class Hierarchy

##### C++

```
interface IRobotTimeHistoryNewmarkParams : IDispatch;
```

##### C#

```
public interface IRobotTimeHistoryNewmarkParams;
```

## Visual Basic

```
Public Interface IRobotTimeHistoryNewmarkParams
```

### Description

Parameters of Newmark method for time history analysis. .

### Version

Available since version 3.

## II.1.10.3.1 IRobotTimeHistoryNewmarkParams Members

The following tables list the members exposed by IRobotTimeHistoryNewmarkParams.

### Public Fields

	Name	Description
◆	Alpha (see page 367)	Damping parameter.
◆	Beta (see page 367)	Damping parameter.
◆	MassMatrixType (see page 368)	Stiffness matrix type.

## II.1.10.3.2 IRobotTimeHistoryNewmarkParams Fields

The fields of the IRobotTimeHistoryNewmarkParams class are listed here.

### Public Fields

	Name	Description
◆	Alpha (see page 367)	Damping parameter.
◆	Beta (see page 367)	Damping parameter.
◆	MassMatrixType (see page 368)	Stiffness matrix type.

## II.1.10.3.2.1 Alpha

### C++

```
HRESULT get_Alpha(double*);  
HRESULT put_Alpha(double);
```

### C#

```
public double Alpha { get; set; }
```

### Visual Basic

```
Public Alpha As Double
```

### Description

Damping parameter.

### Version

Available since version 3.

## II.1.10.3.2.2 Beta

### C++

```
HRESULT get_Beta(double*);  
HRESULT put_Beta(double);
```

### C#

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As double
```

**Description**

Damping parameter.

**Version**

Available since version 3.

**II.1.10.3.2.3 MassMatrixType****C++**

```
HRESULT get_MassMatrixType(IRobotModalAnalysisMassMatrixType* );
HRESULT put_MassMatrixType(IRobotModalAnalysisMassMatrixType);
```

**C#**

```
public IRobotModalAnalysisMassMatrixType MassMatrixType { get; set; }
```

**Visual Basic**

```
Public MassMatrixType As IRobotModalAnalysisMassMatrixType
```

**Description**

Stiffness matrix type.

**Version**

Available since version 3.

**II.1.10.4 IRobotTimeHistoryModalDecompositionParams****Class Hierarchy****C++**

```
interface IRobotTimeHistoryModalDecompositionParams : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryModalDecompositionParams;
```

**Visual Basic**

```
Public Interface IRobotTimeHistoryModalDecompositionParams
```

**Description**

Parameters of modal decomposition method. It is a list of pairs [mode number, damping value]. For each mode, maximally one such a pair may exist. Additionally, the pairs are indexed with numbers from 1 to Count ([see page 369](#)). .

**Version**

Available since version 3.

**II.1.10.4.1 IRobotTimeHistoryModalDecompositionParams Members**

The following tables list the members exposed by IRobotTimeHistoryModalDecompositionParams.

**Public Fields**

	Name	Description
	Count ( <a href="#">see page 369</a> )	Number of defined mode pairs [mode number, damping value].

## Public Methods

	Name	Description
💡	Delete (🔗 see page 369)	Function removes the selected list element.
💡	DeleteMode (🔗 see page 370)	Function deletes the pair containing the selected mode from the list.
💡	Get (🔗 see page 370)	Function returns the next pair from the list.
💡	GetDamping (🔗 see page 370)	Function returns a damping value for a selected mode.
💡	IsDefined (🔗 see page 371)	Function returns non-zero value (True), if for the specified mode damping has been defined. .
💡	SetDamping (🔗 see page 371)	Function sets a damping value for a given mode.

### II.1.10.4.2 IRobotTimeHistoryModalDecompositionParams Fields

The fields of the IRobotTimeHistoryModalDecompositionParams class are listed here.

#### Public Fields

	Name	Description
💡	Count (🔗 see page 369)	Number of defined mode pairs [mode number, damping value].

#### II.1.10.4.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of defined mode pairs [mode number, damping value].

#### Version

Available since version 3.

### II.1.10.4.3 IRobotTimeHistoryModalDecompositionParams Methods

The methods of the IRobotTimeHistoryModalDecompositionParams class are listed here.

#### Public Methods

	Name	Description
💡	Delete (🔗 see page 369)	Function removes the selected list element.
💡	DeleteMode (🔗 see page 370)	Function deletes the pair containing the selected mode from the list.
💡	Get (🔗 see page 370)	Function returns the next pair from the list.
💡	GetDamping (🔗 see page 370)	Function returns a damping value for a selected mode.
💡	IsDefined (🔗 see page 371)	Function returns non-zero value (True), if for the specified mode damping has been defined. .
💡	SetDamping (🔗 see page 371)	Function sets a damping value for a given mode.

#### II.1.10.4.3.1 Delete

##### C++

```
HRESULT Delete(long _idx);
```

**C#**

```
public void Delete(long _idx);
```

**Visual Basic**

```
Public Sub Delete(_idx As Long)
```

**Description**

Function removes the selected list element.

**Version**

Available since version 3.

**II.1.10.4.3.2 DeleteMode****C++**

```
HRESULT DeleteMode(long _mode);
```

**C#**

```
public void DeleteMode(long _mode);
```

**Visual Basic**

```
Public Sub DeleteMode(_mode As Long)
```

**Description**

Function deletes the pair containing the selected mode from the list.

**Version**

Available since version 3.

**II.1.10.4.3.3 Get****C++**

```
HRESULT Get(long _idx, long* _mode, double* _damp);
```

**C#**

```
public void Get(long _idx, long* _mode, double* _damp);
```

**Visual Basic**

```
Public Sub Get(_idx As Long, ByRef _mode As Long*, ByRef _damp As Double*)
```

**Description**

Function returns the next pair from the list.

**Version**

Available since version 3.

**II.1.10.4.3.4 GetDamping****C++**

```
HRESULT GetDamping(long _mode, double* ret);
```

**C#**

```
public double GetDamping(long _mode);
```

**Visual Basic**

```
Public Function GetDamping(_mode As long) As double
```

**Description**

Function returns a damping value for a selected mode.

**Version**

Available since version 3.

**II.1.10.4.3.5 IsDefined****C++**

```
HRESULT IsDefined(long _mode, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsDefined(long _mode);
```

**Visual Basic**

```
Public Function IsDefined(_mode As long) As Boolean
```

**Description**

Function returns non-zero value (True), if for the specified mode damping has been defined. .

**Version**

Available since version 3.

**II.1.10.4.3.6 SetDamping****C++**

```
HRESULT SetDamping(long _mode, double _damp);
```

**C#**

```
public void SetDamping(long _mode, double _damp);
```

**Visual Basic**

```
Public Sub SetDamping(_mode As long, _damp As double)
```

**Description**

Function sets a damping value for a given mode.

**Version**

Available since version 3.

**II.1.10.5 IRobotTimeHistoryPointsCollection****Class Hierarchy****C++**

```
interface IRobotTimeHistoryPointsCollection : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryPointsCollection;
```

**Visual Basic**

```
Public Interface IRobotTimeHistoryPointsCollection
```

**Description**

Collection of points defining a time function for the time history analysis. .

**Version**

Available since version 3.

**II.1.10.5.1 IRobotTimeHistoryPointsCollection Members**

The following tables list the members exposed by IRobotTimeHistoryPointsCollection.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 372</a> )	Number of points in a collection.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add ( <a href="#">see page 373</a> )	Function adds a new point of the (X, Y) coordinates. If a point of X coordinate has been defined earlier, then its value will be replaced by a new Y value. .
◆	Clear ( <a href="#">see page 373</a> )	Function removes all collection points.
◆	Delete ( <a href="#">see page 374</a> )	Function removes selected collection point.
◆	Find ( <a href="#">see page 374</a> )	Function returns index of the point defined for the X coordinate. If for the indicated X coordinate, Y value has not been defined, then function returns zero value (0). .
◆	Get ( <a href="#">see page 374</a> )	Function returns selected collection point.
◆	LoadFromFile ( <a href="#">see page 375</a> )	Function reads in the collection of points from the file and adds it to the points currently defined. .
◆	SaveToFile ( <a href="#">see page 375</a> )	Function saves the collection of points to a file. .

**II.1.10.5.2 IRobotTimeHistoryPointsCollection Fields**

The fields of the IRobotTimeHistoryPointsCollection class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 372</a> )	Number of points in a collection.

**II.1.10.5.2.1 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of points in a collection.

**Version**

Available since version 3.

### II.1.10.5.3 IRobotTimeHistoryPointsCollection Methods

The methods of the IRobotTimeHistoryPointsCollection class are listed here.

#### Public Methods

	Name	Description
⊕	Add ( <a href="#">see page 373</a> )	Function adds a new point of the (X, Y) coordinates. If a point of X coordinate has been defined earlier, then its value will be replaced by a new Y value. .
⊕	Clear ( <a href="#">see page 373</a> )	Function removes all collection points.
⊕	Delete ( <a href="#">see page 374</a> )	Function removes selected collection point.
⊕	Find ( <a href="#">see page 374</a> )	Function returns index of the point defined for the X coordinate. If for the indicated X coordinate, Y value has not been defined, then function returns zero value (0). .
⊕	Get ( <a href="#">see page 374</a> )	Function returns selected collection point.
⊕	LoadFromFile ( <a href="#">see page 375</a> )	Function reads in the collection of points from the file and adds it to the points currently defined. .
⊕	SaveToFile ( <a href="#">see page 375</a> )	Function saves the collection of points to a file. .

#### II.1.10.5.3.1 Add

##### C++

```
HRESULT Add(double _x, double _y);
```

##### C#

```
public void Add(double _x, double _y);
```

##### Visual Basic

```
Public Sub Add(_x As double, _y As double)
```

##### Description

Function adds a new point of the (X, Y) coordinates. If a point of X coordinate has been defined earlier, then its value will be replaced by a new Y value. .

##### Version

Available since version 3.

#### II.1.10.5.3.2 Clear

##### C++

```
HRESULT Clear();
```

##### C#

```
public void Clear();
```

##### Visual Basic

```
Public Sub Clear()
```

##### Description

Function removes all collection points.

##### Version

Available since version 3.

### II.1.10.5.3.3 Delete

#### C++

```
HRESULT Delete(long _idx);
```

#### C#

```
public void Delete(long _idx);
```

#### Visual Basic

```
Public Sub Delete(_idx As long)
```

#### Description

Function removes selected collection point.

#### Version

Available since version 3.

### II.1.10.5.3.4 Find

#### C++

```
HRESULT Find(double _x, long* ret);
```

#### C#

```
public long Find(double _x);
```

#### Visual Basic

```
Public Function Find(_x As double) As long
```

#### Description

Function returns index of the point defined for the X coordinate. If for the indicated X coordinate, Y value has not been defined, then function returns zero value (0).

#### Version

Available since version 3.

### II.1.10.5.3.5 Get

#### C++

```
HRESULT Get(long _idx, double* _x, double* _y);
```

#### C#

```
public void Get(long _idx, double* _x, double* _y);
```

#### Visual Basic

```
Public Sub Get(_idx As long, ByRef _x As double*, ByRef _y As double*)
```

#### Description

Function returns selected collection point.

#### Version

Available since version 3.

### II.1.10.5.3.6 LoadFromFile

**C++**

```
HRESULT LoadFromFile(BSTR _file_path);
```

**C#**

```
public void LoadFromFile(String _file_path);
```

**Visual Basic**

```
Public Sub LoadFromFile(_file_path As String)
```

**Description**

Function reads in the collection of points from the file and adds it to the points currently defined. .

**Version**

Available since version 3.

### II.1.10.5.3.7 SaveToFile

**C++**

```
HRESULT SaveToFile(BSTR _file_path);
```

**C#**

```
public void SaveToFile(String _file_path);
```

**Visual Basic**

```
Public Sub SaveToFile(_file_path As String)
```

**Description**

Function saves the collection of points to a file. .

**Version**

Available since version 3.

## II.1.10.6 IRobotTimeHistoryFunctionList

**Class Hierarchy**

**C++**

```
interface IRobotTimeHistoryFunctionList : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryFunctionList;
```

**Visual Basic**

```
Public Interface IRobotTimeHistoryFunctionList
```

**Description**

List of defined time functions for time history analysis. .

**Version**

Available since version 3.

### II.1.10.6.1 IRobotTimeHistoryFunctionList Members

The following tables list the members exposed by IRobotTimeHistoryFunctionList.

#### Public Fields

	Name	Description
◆	Count (see page 376)	Number of defined (available) time functions.

#### Public Methods

	Name	Description
◆	AddFromFile (see page 377)	Function reads in the list of time functions from the indicated file. .
◆	Create (see page 377)	Function creates and returns a new (empty) collection of points defining a time function.
◆	CreateSum (see page 378)	Function creates and returns the collection of points as a sum of the indicated time functions.
◆	Delete (see page 378)	Function deletes the definition of time function of the specified index. .
◆	Find (see page 378)	Function returns index of the time function definition of the indicated name.
◆	Get (see page 379)	Function returns the collection of points defining a time function of the indicated index. .
◆	GetName (see page 379)	Function returns a name of the time function of the indicated index.
◆	SaveToFile (see page 379)	Function saves the list of time functions to the indicated file. .
◆	Store (see page 380)	Function saves the indicated collection of points as a definition of time function of the specified name. Index of the function definition from the list is returned.

### II.1.10.6.2 IRobotTimeHistoryFunctionList Fields

The fields of the IRobotTimeHistoryFunctionList class are listed here.

#### Public Fields

	Name	Description
◆	Count (see page 376)	Number of defined (available) time functions.

### II.1.10.6.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of defined (available) time functions.

#### Version

Available since version 3.

### II.1.10.6.3 IRobotTimeHistoryFunctionList Methods

The methods of the IRobotTimeHistoryFunctionList class are listed here.

## Public Methods

	Name	Description
≡	AddFromFile ( [ see page 377)	Function reads in the list of time functions from the indicated file. .
≡	Create ( [ see page 377)	Function creates and returns a new (empty) collection of points defining a time function.
≡	CreateSum ( [ see page 378)	Function creates and returns the collection of points as a sum of the indicated time functions.
≡	Delete ( [ see page 378)	Function deletes the definition of time function of the specified index. .
≡	Find ( [ see page 378)	Function returns index of the time function definition of the indicated name.
≡	Get ( [ see page 379)	Function returns the collection of points defining a time function of the indicated index. .
≡	GetName ( [ see page 379)	Function returns a name of the time function of the indicated index.
≡	SaveToFile ( [ see page 379)	Function saves the list of time functions to the indicated file. .
≡	Store ( [ see page 380)	Function saves the indicated collection of points as a definition of time function of the specified name. Index of the function definition from the list is returned.

### II.1.10.6.3.1 AddFromFile

#### C++

```
HRESULT AddFromFile(BSTR _file_path);
```

#### C#

```
public void AddFromFile(String _file_path);
```

#### Visual Basic

```
Public Sub AddFromFile(_file_path As String)
```

#### Description

Function reads in the list of time functions from the indicated file. .

#### Version

Available since version 3.

### II.1.10.6.3.2 Create

#### C++

```
HRESULT Create(IRobotTimeHistoryPointsCollection** ret);
```

#### C#

```
public IRobotTimeHistoryPointsCollection Create();
```

#### Visual Basic

```
Public Function Create() As IRobotTimeHistoryPointsCollection
```

#### Description

Function creates and returns a new (empty) collection of points defining a time function.

#### Version

Available since version 3.

### II.1.10.6.3.3 CreateSum

#### C++

```
HRESULT CreateSum(long _fun1_idx, double _fun1_coeff, long _fun2_idx, double _fun2_coeff,
IRobotTimeHistoryPointsCollection** ret);
```

#### C#

```
public IRobotTimeHistoryPointsCollection CreateSum(long _fun1_idx, double _fun1_coeff, long
_fun2_idx, double _fun2_coeff);
```

#### Visual Basic

```
Public Function CreateSum(_fun1_idx As long, _fun1_coeff As double, _fun2_idx As long,
_fun2_coeff As double) As IRobotTimeHistoryPointsCollection
```

#### Description

Function creates and returns the collection of points as a sum of the indicated time functions.

#### Version

Available since version 3.

### II.1.10.6.3.4 Delete

#### C++

```
HRESULT Delete(long _idx);
```

#### C#

```
public void Delete(long _idx);
```

#### Visual Basic

```
Public Sub Delete(_idx As long)
```

#### Description

Function deletes the definition of time function of the specified index. .

#### Version

Available since version 3.

### II.1.10.6.3.5 Find

#### C++

```
HRESULT Find(BSTR _function_name, long* ret);
```

#### C#

```
public long Find(String _function_name);
```

#### Visual Basic

```
Public Function Find(_function_name As String) As long
```

#### Description

Function returns index of the time function definition of the indicated name.

#### Version

Available since version 3.

### II.10.6.3.6 Get

#### C++

```
HRESULT Get(long _idx, IRobotTimeHistoryPointsCollection** ret);
```

#### C#

```
public IRobotTimeHistoryPointsCollection Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As IRobotTimeHistoryPointsCollection
```

#### Description

Function returns the collection of points defining a time function of the indicated index. .

#### Version

Available since version 3.

### II.10.6.3.7 GetName

#### C++

```
HRESULT GetName(long _idx, BSTR* ret);
```

#### C#

```
public String GetName(long _idx);
```

#### Visual Basic

```
Public Function GetName(_idx As long) As String
```

#### Description

Function returns a name of the time function of the indicated index.

#### Version

Available since version 3.

### II.10.6.3.8 SaveToFile

#### C++

```
HRESULT SaveToFile(BSTR _file_path);
```

#### C#

```
public void SaveToFile(String _file_path);
```

#### Visual Basic

```
Public Sub SaveToFile(_file_path As String)
```

#### Description

Function saves the list of time functions to the indicated file. .

#### Version

Available since version 3.

### II.1.10.6.3.9 Store

#### C++

```
HRESULT Store(BSTR _function_name, IRobotTimeHistoryPointsCollection* _points, long* ret);
```

#### C#

```
public long Store(String _function_name, IRobotTimeHistoryPointsCollection _points);
```

#### Visual Basic

```
Public Function Store(_function_name As String, ByRef _points As IRobotTimeHistoryPointsCollection) As long
```

#### Description

Function saves the indicated collection of points as a definition of time function of the specified name. Index of the function definition from the list is returned.

#### Version

Available since version 3.

## II.1.10.7 IRobotTimeHistoryNonlinearParams

#### Class Hierarchy

#### C++

```
interface IRobotTimeHistoryNonlinearParams : IDispatch;
```

#### C#

```
public interface IRobotTimeHistoryNonlinearParams;
```

#### Visual Basic

```
Public Interface IRobotTimeHistoryNonlinearParams
```

#### Description

Non-linear parameters of time history analysis.

#### Version

Available since version 3.

### II.1.10.7.1 IRobotTimeHistoryNonlinearParams Members

The following tables list the members exposed by IRobotTimeHistoryNonlinearParams.

#### Public Fields

	Name	Description
◆	MatrixUpdateAfterEachIteration ( [ see page 381 )	Matrix update after each iteration.
◆	MatrixUpdateAfterEachSubdivision ( [ see page 381 )	Matrices are updated after each subdivision .
◆	MaximumIterationNumberForOneIncrement ( [ see page 381 )	Maximum number of iterations for an increment.
◆	PDelta ( [ see page 382 )	Definition of P-Delta analysis.
◆	ResidualForcesRelativeCodeTolerance ( [ see page 382 )	Relative norm tolerance for residual forces.

## II.10.7.2 IRobotTimeHistoryNonlinearParams Fields

The fields of the IRobotTimeHistoryNonlinearParams class are listed here.

### Public Fields

	Name	Description
◆	MatrixUpdateAfterEachIteration (see page 381)	Matrix update after each iteration.
◆	MatrixUpdateAfterEachSubdivision (see page 381)	Matrices are updated after each subdivision .
◆	MaximumIterationNumberForOneIncrement (see page 381)	Maximum number of iterations for an increment.
◆	PDelta (see page 382)	Definition of P-Delta analysis.
◆	ResidualForcesRelativeCodeTolerance (see page 382)	Relative norm tolerance for residual forces.

### II.10.7.2.1 MatrixUpdateAfterEachIteration

#### C++

```
HRESULT get_MatrixUpdateAfterEachIteration(VARIANT_BOOL* );
HRESULT put_MatrixUpdateAfterEachIteration(VARIANT_BOOL);
```

#### C#

```
public bool MatrixUpdateAfterEachIteration { get; set; }
```

#### Visual Basic

```
Public MatrixUpdateAfterEachIteration As Boolean
```

#### Description

Matrix update after each iteration.

#### Version

Available since version 3.

### II.10.7.2.2 MatrixUpdateAfterEachSubdivision

#### C++

```
HRESULT get_MatrixUpdateAfterEachSubdivision(VARIANT_BOOL* );
HRESULT put_MatrixUpdateAfterEachSubdivision(VARIANT_BOOL);
```

#### C#

```
public bool MatrixUpdateAfterEachSubdivision { get; set; }
```

#### Visual Basic

```
Public MatrixUpdateAfterEachSubdivision As Boolean
```

#### Description

Matrices are updated after each subdivision .

#### Version

Available since version 3.

### II.10.7.2.3 MaximumIterationNumberForOneIncrement

#### C++

```
HRESULT get_MaximumIterationNumberForOneIncrement(long* );
HRESULT put_MaximumIterationNumberForOneIncrement(long);
```

**C#**

```
public long MaximumIterationNumberForOneIncrement { get; set; }
```

**Visual Basic**

```
Public MaximumIterationNumberForOneIncrement As Long
```

**Description**

Maximum number of iterations for an increment.

**Version**

Available since version 3.

**II.1.10.7.2.4 PDelta****C++**

```
HRESULT get_PDelta(VARIANT_BOOL* );
HRESULT put_PDelta(VARIANT_BOOL);
```

**C#**

```
public bool PDelta { get; set; }
```

**Visual Basic**

```
Public PDelta As Boolean
```

**Description**

Definition of P-Delta analysis.

**Version**

Available since version 3.

**II.1.10.7.2.5 ResidualForcesRelativeCodeTolerance****C++**

```
HRESULT get_ResidualForcesRelativeCodeTolerance(double* );
HRESULT put_ResidualForcesRelativeCodeTolerance(double);
```

**C#**

```
public double ResidualForcesRelativeCodeTolerance { get; set; }
```

**Visual Basic**

```
Public ResidualForcesRelativeCodeTolerance As Double
```

**Description**

Relative norm tolerance for residual forces.

**Version**

Available since version 3.

**II.1.10.8 IRobotTimeHistoryHHTParams****Class Hierarchy****C++**

```
interface IRobotTimeHistoryHHTParams : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryHHTParams;
```

## Visual Basic

```
Public Interface IRobotTimeHistoryHHTParams
```

### Description

Parameters of HHT (Hilber-Hughes-Taylor) method for time history analysis.

### Version

Available since version 11.

#### II.1.10.8.1 IRobotTimeHistoryHHTParams Members

The following tables list the members exposed by IRobotTimeHistoryHHTParams.

### Public Fields

	Name	Description
◆	Alpha (↗ see page 383)	Alpha parameter of Rayleigh damping.
◆	Beta (↗ see page 383)	Rayleigh damping beta parameter.
◆	CoeffAlpha (↗ see page 384)	Numerical damping.
◆	Nonlinearity (↗ see page 384)	Flag which decides on considering the non-linear analysis.
◆	NonlinearParams (↗ see page 384)	Non-linear parameters.

#### II.1.10.8.2 IRobotTimeHistoryHHTParams Fields

The fields of the IRobotTimeHistoryHHTParams class are listed here.

### Public Fields

	Name	Description
◆	Alpha (↗ see page 383)	Alpha parameter of Rayleigh damping.
◆	Beta (↗ see page 383)	Rayleigh damping beta parameter.
◆	CoeffAlpha (↗ see page 384)	Numerical damping.
◆	Nonlinearity (↗ see page 384)	Flag which decides on considering the non-linear analysis.
◆	NonlinearParams (↗ see page 384)	Non-linear parameters.

#### II.1.10.8.2.1 Alpha

##### C++

```
HRESULT get_Alpha(double* );
HRESULT put_Alpha(double);
```

##### C#

```
public double Alpha { get; set; }
```

## Visual Basic

```
Public Alpha As Double
```

### Description

Alpha parameter of Rayleigh damping.

### Version

Available since version 11.

#### II.1.10.8.2.2 Beta

##### C++

```
HRESULT get_Beta(double* );
HRESULT put_Beta(double);
```

**C#**

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As Double
```

**Description**

Rayleigh damping beta parameter.

**Version**

Available since version 11.

**II.1.10.8.2.3 CoeffAlpha****C++**

```
HRESULT get_CoeffAlpha(double* );
HRESULT put_CoeffAlpha(double);
```

**C#**

```
public double CoeffAlpha { get; set; }
```

**Visual Basic**

```
Public CoeffAlpha As Double
```

**Description**

Numerical damping.

**Version**

Available since version 11.

**II.1.10.8.2.4 Nonlinearity****C++**

```
HRESULT get_Nonlinearity(VARIANT_BOOL* );
HRESULT put_Nonlinearity(VARIANT_BOOL);
```

**C#**

```
public bool Nonlinearity { get; set; }
```

**Visual Basic**

```
Public Nonlinearity As Boolean
```

**Description**

Flag which decides on considering the non-linear analysis.

**Version**

Available since version 11.

**II.1.10.8.2.5 NonlinearParams****C++**

```
HRESULT get_NonlinearParams(IRobotTimeHistoryNonlinearParams** );
```

**C#**

```
public IRobotTimeHistoryNonlinearParams NonlinearParams { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearParams As IRobotTimeHistoryNonlinearParams
```

**Description**

Non-linear parameters.

**Version**

Available since version 11.

**II.1.10.9 IRobotTimeHistoryNewmarkAccelParams****Class Hierarchy****C++**

```
interface IRobotTimeHistoryNewmarkAccelParams : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryNewmarkAccelParams;
```

**Visual Basic**

```
Public Interface IRobotTimeHistoryNewmarkAccelParams
```

**Description**

Parameters of Newmark method (acceleration) for time history analysis.

**Version**

Available since version 11.

**II.1.10.9.1 IRobotTimeHistoryNewmarkAccelParams Members**

The following tables list the members exposed by IRobotTimeHistoryNewmarkAccelParams.

**Public Fields**

	Name	Description
❖	Alpha (↗ see page 385)	Alpha parameter of Rayleigh damping.
❖	Beta (↗ see page 386)	Beta parameter of Rayleigh damping.
❖	Nonlinearity (↗ see page 386)	Flag which decides on considering the non-linear analysis.
❖	NonlinearParams (↗ see page 386)	Parameters of non-linear analysis.

**II.1.10.9.2 IRobotTimeHistoryNewmarkAccelParams Fields**

The fields of the IRobotTimeHistoryNewmarkAccelParams class are listed here.

**Public Fields**

	Name	Description
❖	Alpha (↗ see page 385)	Alpha parameter of Rayleigh damping.
❖	Beta (↗ see page 386)	Beta parameter of Rayleigh damping.
❖	Nonlinearity (↗ see page 386)	Flag which decides on considering the non-linear analysis.
❖	NonlinearParams (↗ see page 386)	Parameters of non-linear analysis.

**II.1.10.9.2.1 Alpha****C++**

```
HRESULT get_Alpha(double* );
HRESULT put_Alpha(double);
```

**C#**

```
public double Alpha { get; set; }
```

**Visual Basic**

```
Public Alpha As Double
```

**Description**

Alpha parameter of Rayleigh damping.

**Version**

Available since version 11.

**II.1.10.9.2.2 Beta****C++**

```
HRESULT get_Beta(double*);  
HRESULT put_Beta(double);
```

**C#**

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As Double
```

**Description**

Beta parameter of Rayleigh damping.

**Version**

Available since version 11.

**II.1.10.9.2.3 Nonlinearity****C++**

```
HRESULT get_Nonlinearity(VARIANT_BOOL*);  
HRESULT put_Nonlinearity(VARIANT_BOOL);
```

**C#**

```
public bool Nonlinearity { get; set; }
```

**Visual Basic**

```
Public Nonlinearity As Boolean
```

**Description**

Flag which decides on considering the non-linear analysis.

**Version**

Available since version 11.

**II.1.10.9.2.4 NonlinearParams****C++**

```
HRESULT get_NonlinearParams(IRobotTimeHistoryNonlinearParams**);
```

**C#**

```
public IRobotTimeHistoryNonlinearParams NonlinearParams { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearParams As IRobotTimeHistoryNonlinearParams
```

**Description**

Parameters of non-linear analysis.

**Version**

Available since version 11.

**II.1.11 Harmonic analysis parameters**

Available since version 3.

**Interfaces**

	Name	Description
»	IRobotHarmonicAnalysisParams ( <a href="#">see page 387</a> )	

**II.1.11.1 IRobotHarmonicAnalysisParams****Class Hierarchy****C++**

```
interface IRobotHarmonicAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotHarmonicAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotHarmonicAnalysisParams
```

**Version**

Available since version 3.

**II.1.11.1.1 IRobotHarmonicAnalysisParams Members**

The following tables list the members exposed by IRobotHarmonicAnalysisParams.

**Public Fields**

	Name	Description
◆	Excitation ( <a href="#">see page 387</a> )	Excitation.
◆	MassMatrix ( <a href="#">see page 388</a> )	Stiffness matrix type.

**II.1.11.1.2 IRobotHarmonicAnalysisParams Fields**

The fields of the IRobotHarmonicAnalysisParams class are listed here.

**Public Fields**

	Name	Description
◆	Excitation ( <a href="#">see page 387</a> )	Excitation.
◆	MassMatrix ( <a href="#">see page 388</a> )	Stiffness matrix type.

**II.1.11.1.2.1 Excitation****C++**

```
HRESULT get_Excitation(IRobotModalAnalysisLimits**);
```

**C#**

```
public IRobotModalAnalysisLimits Excitation { get; }
```

**Visual Basic**

```
Public ReadOnly Excitation As IRobotModalAnalysisLimits
```

**Description**

Excitation.

**Version**

Available since version 3.

**II.1.11.1.2.2 MassMatrix****C++**

```
HRESULT get_MassMatrix(IRobotModalAnalysisMassMatrixType* );
HRESULT put_MassMatrix(IRobotModalAnalysisMassMatrixType);
```

**C#**

```
public IRobotModalAnalysisMassMatrixType MassMatrix { get; set; }
```

**Visual Basic**

```
Public MassMatrix As IRobotModalAnalysisMassMatrixType
```

**Description**

Stiffness matrix type.

**Version**

Available since version 3.

**II.1.12 Push-over analysis parameters**

Available since version 3.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotPushOverLoadDefinitionMethod ( <a href="#">see page 391</a> )	Available methods of load definition for push-over analysis.
	IRobotPushOverDirection ( <a href="#">see page 392</a> )	Available directions for Push-over analysis.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotPushOverAnalysisParams ( <a href="#">see page 388</a> )	Parameters of Push-over analysis.

**II.1.12.1 IRobotPushOverAnalysisParams****Class Hierarchy****C++**

```
interface IRobotPushOverAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotPushOverAnalysisParams;
```

## Visual Basic

```
Public Interface IRobotPushOverAnalysisParams
```

### Description

Parameters of Push-over analysis.

### Version

Available since version 3.

#### II.1.12.1.1 IRobotPushOverAnalysisParams Members

The following tables list the members exposed by IRobotPushOverAnalysisParams.

### Public Fields

	Name	Description
❖	Direction ( [ see page 389 )	Direction.
❖	LoadDefinition ( [ see page 390 )	Method of load definition.
❖	MaxDisplacement ( [ see page 390 )	Maximum displacement.
❖	Node ( [ see page 390 )	Node number.
❖	Nonlinear ( [ see page 391 )	Flag switching on/off a non-linear analysis.
❖	NonlinearParams ( [ see page 391 )	Non-linear analysis parameters (considered only if the Nonlinear ( [ see page 391 ) flag is set).

#### II.1.12.1.2 IRobotPushOverAnalysisParams Fields

The fields of the IRobotPushOverAnalysisParams class are listed here.

### Public Fields

	Name	Description
❖	Direction ( [ see page 389 )	Direction.
❖	LoadDefinition ( [ see page 390 )	Method of load definition.
❖	MaxDisplacement ( [ see page 390 )	Maximum displacement.
❖	Node ( [ see page 390 )	Node number.
❖	Nonlinear ( [ see page 391 )	Flag switching on/off a non-linear analysis.
❖	NonlinearParams ( [ see page 391 )	Non-linear analysis parameters (considered only if the Nonlinear ( [ see page 391 ) flag is set).

#### II.1.12.1.2.1 Direction

### C++

```
HRESULT get_Direction( IRobotPushOverDirection* );
HRESULT put_Direction( IRobotPushOverDirection );
```

### C#

```
public IRobotPushOverDirection Direction { get; set; }
```

## Visual Basic

```
Public Direction As IRobotPushOverDirection
```

### Description

Direction.

### Version

Available since version 3.

### II.1.12.1.2.2 LoadDefinition

#### C++

```
HRESULT get_LoadDefinition(IRobotPushOverLoadDefinitionMethod* );
HRESULT put_LoadDefinition(IRobotPushOverLoadDefinitionMethod);
```

#### C#

```
public IRobotPushOverLoadDefinitionMethod LoadDefinition { get; set; }
```

#### Visual Basic

```
Public LoadDefinition As IRobotPushOverLoadDefinitionMethod
```

#### Description

Method of load definition.

#### Version

Available since version 3.

### II.1.12.1.2.3 MaxDisplacement

#### C++

```
HRESULT get_MaxDisplacement(double* );
HRESULT put_MaxDisplacement(double);
```

#### C#

```
public double MaxDisplacement { get; set; }
```

#### Visual Basic

```
Public MaxDisplacement As Double
```

#### Description

Maximum displacement.

#### Version

Available since version 3.

### II.1.12.1.2.4 Node

#### C++

```
HRESULT get_Node(long* );
HRESULT put_Node(long);
```

#### C#

```
public long Node { get; set; }
```

#### Visual Basic

```
Public Node As Long
```

#### Description

Node number.

#### Version

Available since version 3.

### II.1.12.1.2.5 Nonlinear

#### C++

```
HRESULT get_Nonlinear(VARIANT_BOOL* );
HRESULT put_Nonlinear(VARIANT_BOOL);
```

#### C#

```
public bool Nonlinear { get; set; }
```

#### Visual Basic

```
Public Nonlinear As Boolean
```

#### Description

Flag switching on/off a non-linear analysis.

#### Version

Available since version 3.

### II.1.12.1.2.6 NonlinearParams

#### C++

```
HRESULT get_NonlinearParams(IRobotNonlinearAnalysisParams** );
```

#### C#

```
public IRobotNonlinearAnalysisParams NonlinearParams { get; }
```

#### Visual Basic

```
Public ReadOnly NonlinearParams As IRobotNonlinearAnalysisParams
```

#### Description

Non-linear analysis parameters (considered only if the Nonlinear (see page 391) flag is set).

#### Version

Available since version 3.

### II.1.12.2 IRobotPushOverLoadDefinitionMethod

#### C++

```
enum IRobotPushOverLoadDefinitionMethod;
```

#### C#

```
public enum IRobotPushOverLoadDefinitionMethod;
```

#### Visual Basic

```
Public Enum IRobotPushOverLoadDefinitionMethod
```

#### Members

Members	Description
I_POLDM_ACCELERATION = 0	Load definition based on the acceleration in the indicated direction. Available since version 3.
I_POLDM_USER_DEFINED = 2	Available since version 3.

#### Description

Available methods of load definition for push-over analysis.

**Version**

Available since version 3.

**II.1.12.3 IRobotPushOverDirection****C++**

```
enum IRobotPushOverDirection;
```

**C#**

```
public enum IRobotPushOverDirection;
```

**Visual Basic**

```
Public Enum IRobotPushOverDirection
```

**Members**

Members	Description
I_POD_UX_PLUS = 0	Available since version 3.
I_POD_UX_MINUS = 10	Available since version 3.
I_POD_UY_PLUS = 1	Available since version 3.
I_POD_UY_MINUS = 11	Available since version 3.
I_POD_UZ_PLUS = 2	Available since version 3.
I_POD_UZ_MINUS = 12	Available since version 3.

**Description**

Available directions for Push-over analysis.

**Version**

Available since version 3.

**II.1.13 FRF analysis parameters**

Available since version 9.7.

**Interfaces**

	Name	Description
→	IRobotFRFAnalysisParams (see page 392)	Parameters of FRF analysis.

**II.1.13.1 IRobotFRFAnalysisParams****Class Hierarchy****C++**

```
interface IRobotFRFAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotFRFAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotFRFAnalysisParams
```

**Description**

Parameters of FRF analysis.

**Version**

Available since version 9.7.

**II.1.13.1.1 IRobotFRFAnalysisParams Members**

The following tables list the members exposed by IRobotFRFAnalysisParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Damping (see page 393)	Damping parameters.
❖	FinalFrequency (see page 393)	Final frequency.
❖	FrequencyDivision (see page 394)	Division for frequency .
❖	IncludeEigenfrequencies (see page 394)	Flag indicating if eigenfrequencies should be included in calculations.
❖	InitialFrequency (see page 394)	Initial frequency.

**II.1.13.1.2 IRobotFRFAnalysisParams Fields**

The fields of the IRobotFRFAnalysisParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Damping (see page 393)	Damping parameters.
❖	FinalFrequency (see page 393)	Final frequency.
❖	FrequencyDivision (see page 394)	Division for frequency .
❖	IncludeEigenfrequencies (see page 394)	Flag indicating if eigenfrequencies should be included in calculations.
❖	InitialFrequency (see page 394)	Initial frequency.

**II.1.13.1.2.1 Damping****C++**

```
HRESULT get_Damping(IRobotDynamicAnalysisDamping**);
```

**C#**

```
public IRobotDynamicAnalysisDamping Damping { get; }
```

**Visual Basic**

```
Public ReadOnly Damping As IRobotDynamicAnalysisDamping
```

**Description**

Damping parameters.

**Version**

Available since version 9.7.

**II.1.13.1.2.2 FinalFrequency****C++**

```
HRESULT get_FinalFrequency(double*);  
HRESULT put_FinalFrequency(double);
```

**C#**

```
public double FinalFrequency { get; set; }
```

**Visual Basic**

```
Public FinalFrequency As double
```

**Description**

Final frequency.

**Version**

Available since version 9.7.

**II.1.13.1.2.3 FrequencyDivision****C++**

```
HRESULT get_FrequencyDivision(long*);  
HRESULT put_FrequencyDivision(long);
```

**C#**

```
public long FrequencyDivision { get; set; }
```

**Visual Basic**

```
Public FrequencyDivision As long
```

**Description**

Division for frequency .

**Version**

Available since version 9.7.

**II.1.13.1.2.4 IncludeEigenfrequencies****C++**

```
HRESULT get_IncludeEigenfrequencies(VARIANT_BOOL*);  
HRESULT put_IncludeEigenfrequencies(VARIANT_BOOL);
```

**C#**

```
public bool IncludeEigenfrequencies { get; set; }
```

**Visual Basic**

```
Public IncludeEigenfrequencies As Boolean
```

**Description**

Flag indicating if eigenfrequencies should be included in calculations.

**Version**

Available since version 9.7.

**II.1.13.1.2.5 InitialFrequency****C++**

```
HRESULT get_InitialFrequency(double*);  
HRESULT put_InitialFrequency(double);
```

**C#**

```
public double InitialFrequency { get; set; }
```

**Visual Basic**

```
Public InitialFrequency As double
```

**Description**

Initial frequency.

**Version**

Available since version 9.7.

**II.1.14 Footfall analysis parameters**

Available since version 9.7.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotFootfallAnalysisExcitationMethod ( <a href="#">see page 395</a> )	
	IRobotFootfallAnalysisExcitationForces ( <a href="#">see page 395</a> )	
	IRobotFootfallAnalysisNodeSelectionType ( <a href="#">see page 397</a> )	

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotFootfallAnalysisNodeSelection ( <a href="#">see page 396</a> )	
	IRobotFootfallAnalysisModalParams ( <a href="#">see page 398</a> )	Modal analysis parameters for the footfall analysis.
	IRobotFootfallAnalysisParams ( <a href="#">see page 400</a> )	Parameters of Footfall analysis.

**II.1.14.1 IRobotFootfallAnalysisExcitationMethod****C++**

```
enum IRobotFootfallAnalysisExcitationMethod;
```

**C#**

```
public enum IRobotFootfallAnalysisExcitationMethod;
```

**Visual Basic**

```
Public Enum IRobotFootfallAnalysisExcitationMethod
```

**Members**

<b>Members</b>	<b>Description</b>
I_FAEM_SELF_EXCITATION = 1	Available since version 9.7.
I_FAEM_FULL_EXCITATION = 2	Available since version 9.7.

**Version**

Available since version 9.7.

**II.1.14.2 IRobotFootfallAnalysisExcitationForces****C++**

```
enum IRobotFootfallAnalysisExcitationForces;
```

**C#**

```
public enum IRobotFootfallAnalysisExcitationForces;
```

## Visual Basic

```
Public Enum IRobotFootfallAnalysisExcitationForces
```

### Members

Members	Description
I_FAEF_CONCRETE_CENTRE = 1	Available since version 9.7.
I_FAEF_SCI_P354 = 2	Available since version 9.7.

### Version

Available since version 9.7.

## II.1.14.3 IRobotFootfallAnalysisNodeSelection

### Class Hierarchy

#### C++

```
interface IRobotFootfallAnalysisNodeSelection : IDispatch;
```

#### C#

```
public interface IRobotFootfallAnalysisNodeSelection;
```

## Visual Basic

```
Public Interface IRobotFootfallAnalysisNodeSelection
```

### Version

Available since version 9.7.

## II.1.14.3.1 IRobotFootfallAnalysisNodeSelection Members

The following tables list the members exposed by IRobotFootfallAnalysisNodeSelection.

### Public Fields

	Name	Description
❖	SelectedNodes ( <a href="#">see page 396</a> )	
❖	SelectedPanels ( <a href="#">see page 397</a> )	
❖	Type ( <a href="#">see page 397</a> )	

## II.1.14.3.2 IRobotFootfallAnalysisNodeSelection Fields

The fields of the IRobotFootfallAnalysisNodeSelection class are listed here.

### Public Fields

	Name	Description
❖	SelectedNodes ( <a href="#">see page 396</a> )	
❖	SelectedPanels ( <a href="#">see page 397</a> )	
❖	Type ( <a href="#">see page 397</a> )	

## II.1.14.3.2.1 SelectedNodes

#### C++

```
HRESULT get_SelectedNodes(IRobotSelection**);
```

#### C#

```
public IRobotSelection SelectedNodes { get; }
```

**Visual Basic**

```
Public ReadOnly SelectedNodes As IRobotSelection
```

**Version**

Available since version 9.7.

**II.1.14.3.2.2 SelectedPanels****C++**

```
HRESULT get_SelectedPanels(IRobotSelection**);
```

**C#**

```
public IRobotSelection SelectedPanels { get; }
```

**Visual Basic**

```
Public ReadOnly SelectedPanels As IRobotSelection
```

**Version**

Available since version 9.7.

**II.1.14.3.2.3 Type****C++**

```
HRESULT get_Type(IRobotFootfallAnalysisNodeSelectionType*);  
HRESULT put_Type(IRobotFootfallAnalysisNodeSelectionType);
```

**C#**

```
public IRobotFootfallAnalysisNodeSelectionType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotFootfallAnalysisNodeSelectionType
```

**Version**

Available since version 9.7.

**II.1.14.4 IRobotFootfallAnalysisNodeSelectionType****C++**

```
enum IRobotFootfallAnalysisNodeSelectionType;
```

**C#**

```
public enum IRobotFootfallAnalysisNodeSelectionType;
```

**Visual Basic**

```
Public Enum IRobotFootfallAnalysisNodeSelectionType
```

**Members**

Members	Description
I_FANST_ALL_NODES = 1	All nodes. Available since version 9.7.
I_FANST_SELECTED_NODES = 2	Selected nodes. Available since version 9.7.
I_FANST_NODES_BELONGING_TO_SELECTED_PANELS = 3	Nodes belonging to selected panels. Available since version 9.7.

**Version**

Available since version 9.7.

**II.1.14.5 IRobotFootfallAnalysisModalParams****Class Hierarchy****C++**

```
interface IRobotFootfallAnalysisModalParams : IDispatch;
```

**C#**

```
public interface IRobotFootfallAnalysisModalParams;
```

**Visual Basic**

```
Public Interface IRobotFootfallAnalysisModalParams
```

**Description**

Modal analysis parameters for the footfall analysis.

**Version**

Available since version 9.7.

**II.1.14.5.1 IRobotFootfallAnalysisModalParams Members**

The following tables list the members exposed by IRobotFootfallAnalysisModalParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	FrequencyLimit ( <a href="#">see page 398</a> )	Frequency limit.
❖	IgnoreDensity ( <a href="#">see page 399</a> )	Flag indicating if density should be ignored.
❖	IncludeMassForDirX ( <a href="#">see page 399</a> )	Flag indicating if masses should be considered on the X direction .
❖	IncludeMassForDirY ( <a href="#">see page 399</a> )	Flag indicating if masses should be considered on the Y direction.
❖	IncludeMassForDirZ ( <a href="#">see page 400</a> )	Flag indicating if masses should be considered on the Z direction.

**II.1.14.5.2 IRobotFootfallAnalysisModalParams Fields**

The fields of the IRobotFootfallAnalysisModalParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	FrequencyLimit ( <a href="#">see page 398</a> )	Frequency limit.
❖	IgnoreDensity ( <a href="#">see page 399</a> )	Flag indicating if density should be ignored.
❖	IncludeMassForDirX ( <a href="#">see page 399</a> )	Flag indicating if masses should be considered on the X direction .
❖	IncludeMassForDirY ( <a href="#">see page 399</a> )	Flag indicating if masses should be considered on the Y direction.
❖	IncludeMassForDirZ ( <a href="#">see page 400</a> )	Flag indicating if masses should be considered on the Z direction.

**II.1.14.5.2.1 FrequencyLimit****C++**

```
HRESULT get_FrequencyLimit(double*);
```

```

HRESULT put_FrequencyLimit(double);

C#
public double FrequencyLimit { get; set; }

Visual Basic
Public FrequencyLimit As Double

```

**Description**

Frequency limit.

**Version**

Available since version 9.7.

**II.1.14.5.2.2 IgnoreDensity****C++**

```

HRESULT get_IgnoreDensity(VARIANT_BOOL* );
HRESULT put_IgnoreDensity(VARIANT_BOOL);

```

**C#**

```

public bool IgnoreDensity { get; set; }

```

**Visual Basic**

```

Public IgnoreDensity As Boolean

```

**Description**

Flag indicating if density should be ignored.

**Version**

Available since version 9.7.

**II.1.14.5.2.3 IncludeMassForDirX****C++**

```

HRESULT get_IncludeMassForDirX(VARIANT_BOOL* );
HRESULT put_IncludeMassForDirX(VARIANT_BOOL);

```

**C#**

```

public bool IncludeMassForDirX { get; set; }

```

**Visual Basic**

```

Public IncludeMassForDirX As Boolean

```

**Description**

Flag indicating if masses should be considered on the X direction .

**Version**

Available since version 9.7.

**II.1.14.5.2.4 IncludeMassForDirY****C++**

```

HRESULT get_IncludeMassForDirY(VARIANT_BOOL* );
HRESULT put_IncludeMassForDirY(VARIANT_BOOL);

```

**C#**

```

public bool IncludeMassForDirY { get; set; }

```

**Visual Basic**

```
Public IncludeMassForDirY As Boolean
```

**Description**

Flag indicating if masses should be considered on the Y direction.

**Version**

Available since version 9.7.

**II.1.14.5.2.5 IncludeMassForDirZ****C++**

```
HRESULT get_IncludeMassForDirZ(VARIANT_BOOL* );
HRESULT put_IncludeMassForDirZ(VARIANT_BOOL);
```

**C#**

```
public bool IncludeMassForDirZ { get; set; }
```

**Visual Basic**

```
Public IncludeMassForDirZ As Boolean
```

**Description**

Flag indicating if masses should be considered on the Z direction.

**Version**

Available since version 9.7.

**II.1.14.6 IRobotFootfallAnalysisParams****Class Hierarchy****C++**

```
interface IRobotFootfallAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotFootfallAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotFootfallAnalysisParams
```

**Description**

Parameters of Footfall analysis.

**Version**

Available since version 9.7.

**II.1.14.6.1 IRobotFootfallAnalysisParams Members**

The following tables list the members exposed by IRobotFootfallAnalysisParams.

**Public Fields**

	Name	Description
◆	Damping (↗ see page 401)	Damping.
◆	ExcitationForces (↗ see page 401)	
◆	ExcitationMethod (↗ see page 402)	
◆	ExcitationNodes (↗ see page 402)	Selection of excitation nodes.

◆	FootstepsNumber (see page 402)	Number of footsteps.
◆	MaxWalkingFrequency (see page 402)	Maximum walking frequency.
◆	MinWalkingFrequency (see page 403)	Minimum walking frequency.
◆	ModalParams (see page 403)	Parameters of modal analysis.
◆	ResponseNodes (see page 403)	Selection of response nodes.
◆	WalkersWeight (see page 404)	Walker's weight.

### II.1.14.6.2 IRobotFootfallAnalysisParams Fields

The fields of the IRobotFootfallAnalysisParams class are listed here.

#### Public Fields

	Name	Description
◆	Damping (see page 401)	Damping.
◆	ExcitationForces (see page 401)	
◆	ExcitationMethod (see page 402)	
◆	ExcitationNodes (see page 402)	Selection of excitation nodes.
◆	FootstepsNumber (see page 402)	Number of footsteps.
◆	MaxWalkingFrequency (see page 402)	Maximum walking frequency.
◆	MinWalkingFrequency (see page 403)	Minimum walking frequency.
◆	ModalParams (see page 403)	Parameters of modal analysis.
◆	ResponseNodes (see page 403)	Selection of response nodes.
◆	WalkersWeight (see page 404)	Walker's weight.

#### II.1.14.6.2.1 Damping

##### C++

```
HRESULT get_Damping(IRobotDynamicAnalysisDamping**);
```

##### C#

```
public IRobotDynamicAnalysisDamping Damping { get; }
```

##### Visual Basic

```
Public ReadOnly Damping As IRobotDynamicAnalysisDamping
```

##### Description

Damping.

##### Version

Available since version 9.7.

#### II.1.14.6.2.2 ExcitationForces

##### C++

```
HRESULT get_ExcitationForces(IRobotFootfallAnalysisExcitationForces*);  
HRESULT put_ExcitationForces(IRobotFootfallAnalysisExcitationForces);
```

##### C#

```
public IRobotFootfallAnalysisExcitationForces ExcitationForces { get; set; }
```

##### Visual Basic

```
Public ExcitationForces As IRobotFootfallAnalysisExcitationForces
```

**Version**

Available since version 9.7.

**II.1.14.6.2.3 ExcitationMethod****C++**

```
HRESULT get_ExcitationMethod(IRobotFootfallAnalysisExcitationMethod*);  
HRESULT put_ExcitationMethod(IRobotFootfallAnalysisExcitationMethod);
```

**C#**

```
public IRobotFootfallAnalysisExcitationMethod ExcitationMethod { get; set; }
```

**Visual Basic**

```
Public ExcitationMethod As IRobotFootfallAnalysisExcitationMethod
```

**Version**

Available since version 9.7.

**II.1.14.6.2.4 ExcitationNodes****C++**

```
HRESULT get_ExcitationNodes(IRobotFootfallAnalysisNodeSelection**);
```

**C#**

```
public IRobotFootfallAnalysisNodeSelection ExcitationNodes { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationNodes As IRobotFootfallAnalysisNodeSelection
```

**Description**

Selection of excitation nodes.

**Version**

Available since version 9.7.

**II.1.14.6.2.5 FootstepsNumber****C++**

```
HRESULT get_FootstepsNumber(long*);  
HRESULT put_FootstepsNumber(long);
```

**C#**

```
public long FootstepsNumber { get; set; }
```

**Visual Basic**

```
Public FootstepsNumber As Long
```

**Description**

Number of footsteps.

**Version**

Available since version 9.7.

**II.1.14.6.2.6 MaxWalkingFrequency****C++**

```
HRESULT get_MaxWalkingFrequency(double*);
```

```
HRESULT put_MaxWalkingFrequency(double);
```

**C#**

```
public double MaxWalkingFrequency { get; set; }
```

**Visual Basic**

```
Public MaxWalkingFrequency As Double
```

**Description**

Maximum walking frequency.

**Version**

Available since version 9.7.

## II.1.14.6.2.7 MinWalkingFrequency

**C++**

```
HRESULT get_MinWalkingFrequency(double*);  
HRESULT put_MinWalkingFrequency(double);
```

**C#**

```
public double MinWalkingFrequency { get; set; }
```

**Visual Basic**

```
Public MinWalkingFrequency As Double
```

**Description**

Minimum walking frequency.

**Version**

Available since version 9.7.

## II.1.14.6.2.8 ModalParams

**C++**

```
HRESULT get_ModalParams(IRobotFootfallAnalysisModalParams**);
```

**C#**

```
public IRobotFootfallAnalysisModalParams ModalParams { get; }
```

**Visual Basic**

```
Public ReadOnly ModalParams As IRobotFootfallAnalysisModalParams
```

**Description**

Parameters of modal analysis.

**Version**

Available since version 9.7.

## II.1.14.6.2.9 ResponseNodes

**C++**

```
HRESULT get_ResponseNodes(IRobotFootfallAnalysisNodeSelection**);
```

**C#**

```
public IRobotFootfallAnalysisNodeSelection ResponseNodes { get; }
```

**Visual Basic**

```
Public ReadOnly ResponseNodes As IRobotFootfallAnalysisNodeSelection
```

**Description**

Selection of response nodes.

**Version**

Available since version 9.7.

**II.1.14.6.2.10 WalkersWeight****C++**

```
HRESULT get_WalkersWeight(double* );
HRESULT put_WalkersWeight(double);
```

**C#**

```
public double WalkersWeight { get; set; }
```

**Visual Basic**

```
Public WalkersWeight As Double
```

**Description**

Walker's weight.

**Version**

Available since version 9.7.

**II.1.15 IRobotCase****Class Hierarchy****C++**

```
interface IRobotCase : IDispatch;
```

**C#**

```
public interface IRobotCase;
```

**Visual Basic**

```
Public Interface IRobotCase
```

**Description**

The part of the load case interface that is shared by simple cases and combinations has been defined using RobotCase. .

**II.1.15.1 IRobotCase Members**

The following tables list the members exposed by IRobotCase.

**Public Fields**

	Name	Description
◆	AnalyzeType (see page 405)	Analysis type.
◆	Name (see page 405)	User-defined case name .
◆	Nature (see page 405)	Case nature .
◆	Number (see page 406)	User-defined number linked with the load case .
◆	Type (see page 406)	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .

## II.1.15.2 IRobotCase Fields

The fields of the IRobotCase class are listed here.

### Public Fields

	Name	Description
◆	AnalyzeType (see page 405)	Analysis type.
◆	Name (see page 405)	User-defined case name .
◆	Nature (see page 405)	Case nature .
◆	Number (see page 406)	User-defined number linked with the load case .
◆	Type (see page 406)	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .

### II.1.15.2.1 AnalyzeType

#### C++

```
HRESULT get_AnalyzeType(IRobotCaseAnalyzeType* );
HRESULT put_AnalyzeType(IRobotCaseAnalyzeType);
```

#### C#

```
public IRobotCaseAnalyzeType AnalyzeType { get; set; }
```

#### Visual Basic

```
Public AnalyzeType As IRobotCaseAnalyzeType
```

#### Description

Analysis type.

### II.1.15.2.2 Name

#### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Description

User-defined case name .

### II.1.15.2.3 Nature

#### C++

```
HRESULT get_Nature(IRobotCaseNature* );
HRESULT put_Nature(IRobotCaseNature);
```

#### C#

```
public IRobotCaseNature Nature { get; set; }
```

#### Visual Basic

```
Public Nature As IRobotCaseNature
```

#### Description

Case nature .

#### II.1.15.2.4 Number

##### C++

```
HRESULT get_Number( long* );
HRESULT put_Number( long );
```

##### C#

```
public long Number { get; set; }
```

##### Visual Basic

```
Public Number As long
```

##### Description

User-defined number linked with the load case .

#### II.1.15.2.5 Type

##### C++

```
HRESULT get_Type( IRobotCaseType* );
```

##### C#

```
public IRobotCaseType Type { get; }
```

##### Visual Basic

```
Public ReadOnly Type As IRobotCaseType
```

##### Description

Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .

#### II.1.16 IRobotCaseNature

##### C++

```
enum IRobotCaseNature;
```

##### C#

```
public enum IRobotCaseNature;
```

##### Visual Basic

```
Public Enum IRobotCaseNature
```

##### Members

Members	Description
I_CN_PERMANENT = 0	Dead .
I_CN_TEMPERATURE = 4	Temperature.
I_CN_EXPLOATATION = 1	Live .
I_CN_WIND = 2	Wind.
I_CN_SNOW = 3	Snow.
I_CN_SEISMIC = 6	Seismic.
I_CN_ACCIDENTAL = 5	Available since version 1.7.

##### Description

The set of case natures accepted by Robot is determined. .

## II.1.17 IRobotCaseAnalizeType

**C++**

```
enum IRobotCaseAnalizeType;
```

**C#**

```
public enum IRobotCaseAnalizeType;
```

**Visual Basic**

```
Public Enum IRobotCaseAnalizeType
```

**Members**

Members	Description
I_CAT_COMB_BUCKLING = -4	Buckling combination.
I_CAT_COMB_NONLINEAR_BUCKLING = -3	Non-linear buckling combination.
I_CAT_COMB_NONLINEAR = -1	Non-linear combination.
I_CAT_COMB = 0	Combination.
I_CAT_STATIC_LINEAR = 1	Static case.
I_CAT_STATIC_NONLINEAR = 2	Non-linear static case.
I_CAT_STATIC_LINEAR_AUXILIARY = 5	Auxiliary case.
I_CAT_DYNAMIC_MODAL = 11	Modal analysis.
I_CAT_DYNAMIC_SPECTRAL = 12	Spectral analysis.
I_CAT_DYNAMIC_SEISMIC = 13	Seismic analysis.
I_CAT_DYNAMIC_HARMONIC = 14	Harmonic analysis.
I_CAT_MOBILE_MAIN = 30	Moving load analysis - main case.
I_CAT_COMB_CODE = -12	Code combination Available since version 1.7.
I_CAT_STATIC_BUCKLING = 4	Buckling analysis case Available since version 2.0.
I_CAT_STATIC_NONLINEAR_BUCKLING = 7	Non-linear buckling analysis case Available since version 2.0.
I_CAT_DYNAMIC_NONLINEAR_MODAL_WITH_STATIC_FORCE = 8	Non-linear modal analysis considering static forces Available since version 2.0.
I_CAT_TIME_HISTORY = 20	Time history analysis case. Available since version 3.
I_CAT_PUSH_OVER = 25	Push-over analysis. Available since version 3.
I_CAT_DYNAMIC_FRF = 15	Available since version 9.7.
I_CAT_DYNAMIC_FOOTFALL = 16	Available since version 9.7.
I_CAT_STATIC_ELF_SEISMIC = 9	Available since version 12.5.

**Description**

Robot Object Model defines a set of analysis types for load cases accepted by Robot. .

## II.1.18 IRobotCaseServer

**Class Hierarchy**

**C++**

```
interface IRobotCaseServer : IDispatch;
```

**C#**

```
public interface IRobotCaseServer;
```

## Visual Basic

```
Public Interface IRobotCaseServer
```

### Description

Server of load cases .

#### II.1.18.1 IRobotCaseServer Members

The following tables list the members exposed by IRobotCaseServer.

##### Public Fields

	Name	Description
◆	CodeCmbEngine ( [ see page 409)	Generator of code-dependent loads Available since version 1.7.
◆	FreeNumber ( [ see page 409)	First available (free) user's number for a load case Available since version 1.7.
◆	QCmbTau ( [ see page 410)	
◆	SELFSeismicEngine ( [ see page 410)	Generator of loads for the simplified seismic analysis.
◆	SnowWindEngine ( [ see page 410)	Generator of snow/wind loads Available since version 1.7.
◆	TimeHistoryFunctions ( [ see page 410)	List of available time functions for time history analysis.
◆	WindLoadsSimulationEngine ( [ see page 411)	

##### Public Methods

	Name	Description
≡◆	BeginMultiOperation ( [ see page 412)	Function enables speed-up of generation or modification of a whole group of load cases through appropriate optimization of some operations. After completing generation of load cases, EndMultiOperation ( [ see page 414)() function should be called up to save changes in a structure.
≡◆	CreateCombination ( [ see page 412)	The function creates and returns combinations with the defined parameters. .
≡◆	CreateMobile ( [ see page 412)	Function creates moving load case.
≡◆	CreateSimple ( [ see page 413)	The function creates and returns a simple load case with the defined parameters. .
≡◆	Delete ( [ see page 413)	The function deletes from the structure model the load case with the indicated number .
≡◆	DeleteMany ( [ see page 413)	The function deletes all the load cases that meet the selection criteria introduced as argument. .
≡◆	EndMultiOperation ( [ see page 414)	Function updates definitions of load cases and combinations in a structure. It should be called up only if BeginMultiOperation ( [ see page 412)() function has been called up earlier.
≡◆	Exist ( [ see page 414)	The function checks if there exists a load case with the user-defined number.
≡◆	FindWithId ( [ see page 414)	Function returns a load case number with the specified unique identifier. If the load case is not found, then the function returns zero value (0). Available since version 1.7.
≡◆	Get ( [ see page 414)	The function returns a load case with the user-defined number. The type of the returned object depends on the real case type with the indicated number. If the indicated number defines a combination, the function returns an object of the RobotCaseCombination object type; otherwise, it will return RobotSimpleCase object type. If the case with the indicated number does not exist, running this function leads to critical error. .
≡◆	GetAll ( [ see page 415)	The function returns a collection of all load cases defined for a given structure.. .

	GetCmpntCount ( <a href="#">see page 415</a> )	Function returns a number of components of the specified load case. What a component means depends on a load case and analysis type; for example, for an FRF analysis case, components are successive steps.
	GetMany ( <a href="#">see page 415</a> )	The function returns a collection of load cases that meet the criterion of selection introduced as argument.
	GetRelatedValue ( <a href="#">see page 416</a> )	Function returns a specified value type that is associated with the specified load case.
	GetUniqueld ( <a href="#">see page 416</a> )	Function returns a unique identifier for a load case with the user-specified number.
	IsAuxiliary ( <a href="#">see page 416</a> )	
	SetAuxiliary ( <a href="#">see page 417</a> )	

## II.1.18.2 IRobotCaseServer Fields

The fields of the IRobotCaseServer class are listed here.

### Public Fields

	Name	Description
	CodeCmbEngine ( <a href="#">see page 409</a> )	Generator of code-dependent loads Available since version 1.7.
	FreeNumber ( <a href="#">see page 409</a> )	First available (free) user's number for a load case Available since version 1.7.
	QCmbTau ( <a href="#">see page 410</a> )	
	SELFSeismicEngine ( <a href="#">see page 410</a> )	Generator of loads for the simplified seismic analysis.
	SnowWindEngine ( <a href="#">see page 410</a> )	Generator of snow/wind loads Available since version 1.7.
	TimeHistoryFunctions ( <a href="#">see page 410</a> )	List of available time functions for time history analysis.
	WindLoadsSimulationEngine ( <a href="#">see page 411</a> )	

### II.1.18.2.1 CodeCmbEngine

#### C++

```
HRESULT get_CodeCmbEngine(IRobotCodeCombinationEngine**);
```

#### C#

```
public IRobotCodeCombinationEngine CodeCmbEngine { get; }
```

#### Visual Basic

```
Public ReadOnly CodeCmbEngine As IRobotCodeCombinationEngine
```

#### Description

Generator of code-dependent loads Available since version 1.7.

### II.1.18.2.2 FreeNumber

#### C++

```
HRESULT get_FreeNumber(long*);
```

#### C#

```
public long FreeNumber { get; }
```

#### Visual Basic

```
Public ReadOnly FreeNumber As long
```

#### Description

First available (free) user's number for a load case Available since version 1.7.

### II.1.18.2.3 QCmbTau

**C++**

```
HRESULT get_QCmbTau(double*);  
HRESULT put_QCmbTau(double);
```

**C#**

```
public double QCmbTau { get; set; }
```

**Visual Basic**

```
Public QCmbTau As Double
```

### II.1.18.2.4 SELFSeismicEngine

**C++**

```
HRESULT get_SELFSeismicEngine(IRobotSELFSeismicEngine**);
```

**C#**

```
public IRobotSELFSeismicEngine SELFSeismicEngine { get; }
```

**Visual Basic**

```
Public ReadOnly SELFSeismicEngine As IRobotSELFSeismicEngine
```

**Description**

Generator of loads for the simplified seismic analysis.

**Version**

Available since version 12.5.

### II.1.18.2.5 SnowWindEngine

**C++**

```
HRESULT get_SnowWindEngine(IRobotSnowWindEngine**);
```

**C#**

```
public IRobotSnowWindEngine SnowWindEngine { get; }
```

**Visual Basic**

```
Public ReadOnly SnowWindEngine As IRobotSnowWindEngine
```

**Description**

Generator of snow/wind loads Available since version 1.7.

### II.1.18.2.6 TimeHistoryFunctions

**C++**

```
HRESULT get_TimeHistoryFunctions(IRobotTimeHistoryFunctionList**);
```

**C#**

```
public IRobotTimeHistoryFunctionList TimeHistoryFunctions { get; }
```

**Visual Basic**

```
Public ReadOnly TimeHistoryFunctions As IRobotTimeHistoryFunctionList
```

**Description**

List of available time functions for time history analysis.

**Version**

Available since version 3.

**II.1.18.2.7 WindLoadsSimulationEngine****C++**

```
HRESULT get_WindLoadsSimulationEngine( IRobotWindLoadsSimulationEngine** );
```

**C#**

```
public IRobotWindLoadsSimulationEngine WindLoadsSimulationEngine { get; }
```

**Visual Basic**

```
Public ReadOnly WindLoadsSimulationEngine As IRobotWindLoadsSimulationEngine
```

**Version**

Available since version 14.5.

**II.1.18.3 IRobotCaseServer Methods**

The methods of the IRobotCaseServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳️	BeginMultiOperation (see page 412)	Function enables speed-up of generation or modification of a whole group of load cases through appropriate optimization of some operations. After completing generation of load cases, EndMultiOperation (see page 414)() function should be called up to save changes in a structure.
✳️	CreateCombination (see page 412)	The function creates and returns combinations with the defined parameters. .
✳️	CreateMobile (see page 412)	Function creates moving load case.
✳️	CreateSimple (see page 413)	The function creates and returns a simple load case with the defined parameters. .
✳️	Delete (see page 413)	The function deletes from the structure model the load case with the indicated number .
✳️	DeleteMany (see page 413)	The function deletes all the load cases that meet the selection criteria introduced as argument. .
✳️	EndMultiOperation (see page 414)	Function updates definitions of load cases and combinations in a structure. It should be called up only if BeginMultiOperation (see page 412)() function has been called up earlier.
✳️	Exist (see page 414)	The function checks if there exists a load case with the user-defined number.
✳️	FindWithId (see page 414)	Function returns a load case number with the specified unique identifier. If the load case is not found, then the function returns zero value (0). Available since version 1.7.
✳️	Get (see page 414)	The function returns a load case with the user-defined number. The type of the returned object depends on the real case type with the indicated number. If the indicated number defines a combination, the function returns an object of the RobotCaseCombination object type; otherwise, it will return RobotSimpleCase object type. If the case with the indicated number does not exist, running this function leads to critical error. .
✳️	GetAll (see page 415)	The function returns a collection of all load cases defined for a given structure. .
✳️	GetCmpntCount (see page 415)	Function returns a number of components of the specified load case. What a component means depends on a load case and analysis type; for example, for an FRF analysis case, components are successive steps.
✳️	GetMany (see page 415)	The function returns a collection of load cases that meet the criterion of selection introduced as argument. .

	GetRelatedValue (see page 416)	Function returns a specified value type that is associated with the specified load case.
	GetUniqueld (see page 416)	Function returns a unique identifier for a load case with the user-specified number.
	IsAuxiliary (see page 416)	
	SetAuxiliary (see page 417)	

### II.1.18.3.1 BeginMultiOperation

#### C++

```
HRESULT BeginMultiOperation();
```

#### C#

```
public void BeginMultiOperation();
```

#### Visual Basic

```
Public Sub BeginMultiOperation()
```

#### Description

Function enables speed-up of generation or modification of a whole group of load cases through appropriate optimization of some operations. After completing generation of load cases, EndMultiOperation (see page 414)() function should be called up to save changes in a structure.

#### Version

Available since version 14.

### II.1.18.3.2 CreateCombination

#### C++

```
HRESULT CreateCombination(long _num, BSTR _name, IRobotCombinationType _type,
IRobotCaseNature _nature, IRobotCaseAnalizeType _analize_type, IRobotCaseCombination** ret);
```

#### C#

```
public IRobotCaseCombination CreateCombination(long _num, String _name,
IRobotCombinationType _type, IRobotCaseNature _nature, IRobotCaseAnalizeType _analize_type);
```

#### Visual Basic

```
Public Function CreateCombination(_num As long, _name As String, _type As
IRobotCombinationType, _nature As IRobotCaseNature, _analize_type As IRobotCaseAnalizeType)
As IRobotCaseCombination
```

#### Description

The function creates and returns combinations with the defined parameters. .

### II.1.18.3.3 CreateMobile

#### C++

```
HRESULT CreateMobile(long _number, BSTR _name, IRobotCaseNature _nature, IRobotMobileCase** ret);
```

#### C#

```
public IRobotMobileCase CreateMobile(long _number, String _name, IRobotCaseNature _nature);
```

#### Visual Basic

```
Public Function CreateMobile(_number As long, _name As String, _nature As IRobotCaseNature)
```

```
As IRobotMobileCase
```

#### Description

Function creates moving load case.

#### Version

Available since version 3.

### II.1.18.3.4 CreateSimple

#### C++

```
HRESULT CreateSimple(long _number, BSTR _name, IRobotCaseNature _nature,
IRobotCaseAnalizeType _analize_type, IRobotSimpleCase** ret);
```

#### C#

```
public IRobotSimpleCase CreateSimple(long _number, String _name, IRobotCaseNature _nature,
IRobotCaseAnalizeType _analize_type);
```

#### Visual Basic

```
Public Function CreateSimple(_number As long, _name As String, _nature As IRobotCaseNature,
_analize_type As IRobotCaseAnalizeType) As IRobotSimpleCase
```

#### Description

The function creates and returns a simple load case with the defined parameters. .

### II.1.18.3.5 Delete

#### C++

```
HRESULT Delete(long _number);
```

#### C#

```
public void Delete(long _number);
```

#### Visual Basic

```
Public Sub Delete(_number As long)
```

#### Description

The function deletes from the structure model the load case with the indicated number .

### II.1.18.3.6 DeleteMany

#### C++

```
HRESULT DeleteMany(IRobotSelection* _case_selection);
```

#### C#

```
public void DeleteMany(IRobotSelection _case_selection);
```

#### Visual Basic

```
Public Sub DeleteMany(ByRef _case_selection As IRobotSelection)
```

#### Description

The function deletes all the load cases that meet the selection criteria introduced as argument. .

### II.1.18.3.7 EndMultiOperation

**C++**

```
HRESULT EndMultiOperation();
```

**C#**

```
public void EndMultiOperation();
```

**Visual Basic**

```
Public Sub EndMultiOperation()
```

**Description**

Function updates definitions of load cases and combinations in a structure. It should be called up only if BeginMultiOperation (see page 412)() function has been called up earlier.

**Version**

Available since version 14.

### II.1.18.3.8 Exist

**C++**

```
HRESULT Exist(long _number, VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist(long _number);
```

**Visual Basic**

```
Public Function Exist(_number As long) As Boolean
```

**Description**

The function checks if there exists a load case with the user-defined number.

### II.1.18.3.9 FindWithId

**C++**

```
HRESULT FindWithId(long _unique_id, long* ret);
```

**C#**

```
public long FindWithId(long _unique_id);
```

**Visual Basic**

```
Public Function FindWithId(_unique_id As long) As long
```

**Description**

Function returns a load case number with the specified unique identifier. If the load case is not found, then the function returns zero value (0). Available since version 1.7.

### II.1.18.3.10 Get

**C++**

```
HRESULT Get(long _number, IRobotCase** ret);
```

**C#**

```
public IRobotCase Get(long _number);
```

**Visual Basic**

```
Public Function Get(_number As long) As IRobotCase
```

**Description**

The function returns a load case with the user-defined number. The type of the returned object depends on the real case type with the indicated number. If the indicated number defines a combination, the function returns an object of the RobotCaseCombination object type; otherwise, it will return RobotSimpleCase object type. If the case with the indicated number does not exist, running this function leads to critical error. .

**II.1.18.3.11 GetAll****C++**

```
HRESULT GetAll(IRobotCaseCollection** ret);
```

**C#**

```
public IRobotCaseCollection GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotCaseCollection
```

**Description**

The function returns a collection of all load cases defined for a given structure. .

**II.1.18.3.12 GetCmpntCount****C++**

```
HRESULT GetCmpntCount(long _case, long* ret);
```

**C#**

```
public long GetCmpntCount(long _case);
```

**Visual Basic**

```
Public Function GetCmpntCount(_case As long) As long
```

**Description**

Function returns a number of components of the specified load case. What a component means depends on a load case and analysis type; for example, for an FRF analysis case, components are successive steps.

**Version**

Available since version 9.7.

**II.1.18.3.13 GetMany****C++**

```
HRESULT GetMany(IRobotSelection* _case_selection, IRobotCaseCollection** ret);
```

**C#**

```
public IRobotCaseCollection GetMany(IRobotSelection _case_selection);
```

**Visual Basic**

```
Public Function GetMany(ByRef _case_selection As IRobotSelection) As IRobotCaseCollection
```

## Description

The function returns a collection of load cases that meet the criterion of selection introduced as argument. .

### II.1.18.3.14 GetRelatedValue

#### C++

```
HRESULT GetRelatedValue(IRobotCaseRelatedValueType _val_type, long _case, long _case_cmpnt = 1, VARIANT* ret);
```

#### C#

```
public VARIANT GetRelatedValue(IRobotCaseRelatedValueType _val_type, long _case, long _case_cmpnt = 1);
```

#### Visual Basic

```
Public Function GetRelatedValue(_val_type As IRobotCaseRelatedValueType, _case As long, Optional _case_cmpnt As long = 1) As VARIANT
```

#### Description

Function returns a specified value type that is associated with the specified load case.

#### Version

Available since version 9.7.

### II.1.18.3.15 GetUniqueId

#### C++

```
HRESULT GetUniqueId(long _case_num, long* ret);
```

#### C#

```
public long GetUniqueId(long _case_num);
```

#### Visual Basic

```
Public Function GetUniqueId(_case_num As long) As long
```

#### Description

Function returns a unique identifier for a load case with the user-specified number.

#### Version

Available since version 8.2.

### II.1.18.3.16 IsAuxiliary

#### C++

```
HRESULT IsAuxiliary(long _case_num, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsAuxiliary(long _case_num);
```

#### Visual Basic

```
Public Function IsAuxiliary(_case_num As long) As Boolean
```

#### Version

Available since version 8.1.

### II.1.18.3.17 SetAuxiliary

**C++**

```
HRESULT SetAuxiliary(long _case_num, VARIANT_BOOL _is_aux);
```

**C#**

```
public void SetAuxiliary(long _case_num, bool _is_aux);
```

**Visual Basic**

```
Public Sub SetAuxiliary(_case_num As long, _is_aux As Boolean)
```

**Version**

Available since version 8.1.

### II.1.19 IRobotSimpleCase

**Class Hierarchy**

**C++**

```
interface IRobotSimpleCase : IRobotCase;
```

**C#**

```
public interface IRobotSimpleCase : IRobotCase;
```

**Visual Basic**

```
Public Interface IRobotSimpleCase
```

**Description**

A simple load case is defined using the list of load records. .

#### II.1.19.1 IRobotSimpleCase Members

The following tables list the members exposed by IRobotSimpleCase.

**Public Fields**

	Name	Description
◆	AnalyzeType ( [ see page 405)	Analysis type.
◆	IsAuxiliary ( [ see page 418)	
◆	Label ( [ see page 418)	
◆	MainMode ( [ see page 419)	Number ( [ see page 406) of the main mode (0 denotes lack of the main mode) Available since version 1.7.
◆	ModesCount ( [ see page 419)	Number ( [ see page 406) of modes Available since version 1.7.
◆	Name ( [ see page 405)	User-defined case name .
◆	Nature ( [ see page 405)	Case nature .
◆	NatureName ( [ see page 419)	Name ( [ see page 405) of load case nature.
◆	Number ( [ see page 406)	User-defined number linked with the load case .
◆	Records ( [ see page 419)	List of load records associated with the case .
◆	TimeStepCount ( [ see page 420)	Number ( [ see page 406) of defined steps (it concerns only cases of time history analysis) .
◆	Type ( [ see page 406)	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .
◆	UniqueId ( [ see page 420)	Unique load case identifier Available since version 1.7.

## Public Methods

	Name	Description
≡	GetAnalysisParams ( [ see page 421) ]	Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned).
≡	GetSeismicCode ( [ see page 421) ]	Function returns the name of a seismic code for which a given load case has been defined. For cases whose analysis type is different from seismic analysis, the function returns an empty character string. .
≡	SetAnalysisParams ( [ see page 421) ]	Function sets analysis parameters for the load case. If the specified parameters cannot be set for this case (e.g. due to incompatibility between the type of case analysis and type of the object defining parameters), then zero value (False) is returned. .
≡	SetNatureExt ( [ see page 422) ]	Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .

### II.1.19.2 IRobotSimpleCase Fields

The fields of the IRobotSimpleCase class are listed here.

#### Public Fields

	Name	Description
◆	IsAuxiliary ( [ see page 418) ]	
◆	Label ( [ see page 418) ]	
◆	MainMode ( [ see page 419) ]	Number ( [ see page 406) of the main mode (0 denotes lack of the main mode) Available since version 1.7.
◆	ModesCount ( [ see page 419) ]	Number ( [ see page 406) of modes Available since version 1.7.
◆	NatureName ( [ see page 419) ]	Name ( [ see page 405) of load case nature.
◆	Records ( [ see page 419) ]	List of load records associated with the case .
◆	TimeStepCount ( [ see page 420) ]	Number ( [ see page 406) of defined steps (it concerns only cases of time history analysis) .
◆	UniqueId ( [ see page 420) ]	Unique load case identifier Available since version 1.7.

#### II.1.19.2.1 IsAuxiliary

##### C++

```
HRESULT get_IsAuxiliary(VARIANT_BOOL* );
HRESULT put_IsAuxiliary(VARIANT_BOOL );
```

##### C#

```
public bool IsAuxiliary { get; set; }
```

##### Visual Basic

```
Public IsAuxiliary As Boolean
```

##### Version

Available since version 8.1.

#### II.1.19.2.2 Label

##### C++

```
HRESULT get_Label(BSTR* );
HRESULT put_Label(BSTR );
```

##### C#

```
public String Label { get; set; }
```

**Visual Basic**

```
Public Label As String
```

**Version**

Available since version 8.

**II.1.19.2.3 MainMode****C++**

```
HRESULT get_MainMode(long* );
HRESULT put_MainMode(long);
```

**C#**

```
public long MainMode { get; set; }
```

**Visual Basic**

```
Public MainMode As long
```

**Description**

Number (see page 406) of the main mode (0 denotes lack of the main mode) Available since version 1.7.

**II.1.19.2.4 ModesCount****C++**

```
HRESULT get_ModesCount(long* );
HRESULT put_ModesCount(long);
```

**C#**

```
public long ModesCount { get; set; }
```

**Visual Basic**

```
Public ModesCount As long
```

**Description**

Number (see page 406) of modes Available since version 1.7.

**II.1.19.2.5 NatureName****C++**

```
HRESULT get_NatureName(BSTR* );
```

**C#**

```
public String NatureName { get; }
```

**Visual Basic**

```
Public ReadOnly NatureName As String
```

**Description**

Name (see page 405) of load case nature.

**Version**

Available since version 3.

**II.1.19.2.6 Records****C++**

```
HRESULT get_Records(IRobotLoadRecordMngr** );
```

**C#**

```
public IRobotLoadRecordMngr Records { get; }
```

**Visual Basic**

```
Public ReadOnly Records As IRobotLoadRecordMngr
```

**Description**

List of load records associated with the case .

**II.1.19.2.7 TimeStepCount****C++**

```
HRESULT get_TimeStepCount(long*);
```

**C#**

```
public long TimeStepCount { get; }
```

**Visual Basic**

```
Public ReadOnly TimeStepCount As long
```

**Description**

Number (see page 406) of defined steps (it concerns only cases of time history analysis) .

**Version**

Available since version 3.

**II.1.19.2.8 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique load case identifier Available since version 1.7.

**II.1.19.3 IRobotSimpleCase Methods**

The methods of the IRobotSimpleCase class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	GetAnalysisParams (see page 421)	Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned).
✳	GetSeismicCode (see page 421)	Function returns the name of a seismic code for which a given load case has been defined. For cases whose analysis type is different from seismic analysis, the function returns an empty character string. .
✳	SetAnalysisParams (see page 421)	Function sets analysis parameters for the load case. If the specified parameters cannot be set for this case (e.g. due to incompatibility between the type of case analysis and type of the object defining parameters), then zero value (False) is returned. .

	SetNatureExt (see page 422)	Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .
-----------------------------------------------------------------------------------	-----------------------------	--------------------------------------------------------------------------------------------------------------------------------------

### II.1.19.3.1 GetAnalysisParams

**C++**

```
HRESULT GetAnalysisParams(IDispatch* ret);
```

**C#**

```
public IDispatch GetAnalysisParams();
```

**Visual Basic**

```
Public Function GetAnalysisParams() As IDispatch
```

**Description**

Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned).

**Version**

Available since version 2.5.

### II.1.19.3.2 GetSeismicCode

**C++**

```
HRESULT GetSeismicCode(BSTR* ret);
```

**C#**

```
public String GetSeismicCode();
```

**Visual Basic**

```
Public Function GetSeismicCode() As String
```

**Description**

Function returns the name of a seismic code for which a given load case has been defined. For cases whose analysis type is different from seismic analysis, the function returns an empty character string. .

**Version**

Available since version 2.5.

### II.1.19.3.3 SetAnalysisParams

**C++**

```
HRESULT SetAnalysisParams(IDispatch _params, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetAnalysisParams(IDispatch _params);
```

**Visual Basic**

```
Public Function SetAnalysisParams(_params As IDispatch) As Boolean
```

**Description**

Function sets analysis parameters for the load case. If the specified parameters cannot be set for this case (e.g. due to incompatibility between the type of case analysis and type of the object defining parameters), then zero value (False) is returned. .

**Version**

Available since version 2.5.

**II.1.19.3.4 SetNatureExt****C++**

```
HRESULT SetNatureExt(long _rgl_action);
```

**C#**

```
public void SetNatureExt(long _rgl_action);
```

**Visual Basic**

```
Public Sub SetNatureExt(_rgl_action As long)
```

**Description**

Function assigns - to the case - the nature consistent with the indicated number of "action" described in the current regulations. .

**Version**

Available since version 3.

**II.1.20 IRobotCaseCombination****Class Hierarchy****C++**

```
interface IRobotCaseCombination : IRobotCase;
```

**C#**

```
public interface IRobotCaseCombination : IRobotCase;
```

**Visual Basic**

```
Public Interface IRobotCaseCombination
```

**Description**

A combination is defined by means of simple load cases with appropriate factors. .

**II.1.20.1 IRobotCaseCombination Members**

The following tables list the members exposed by IRobotCaseCombination.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AnalyzeType ( <a href="#">see page 405</a> )	Analysis type.
◆	CaseFactors ( <a href="#">see page 423</a> )	List of pairs (simple case, factor) defining a combination .
◆	CombinationType ( <a href="#">see page 423</a> )	Combination type .
◆	IsAuxiliary ( <a href="#">see page 424</a> )	
◆	Label ( <a href="#">see page 424</a> )	
◆	Name ( <a href="#">see page 405</a> )	User-defined case name .
◆	Nature ( <a href="#">see page 405</a> )	Case nature .
◆	NatureName ( <a href="#">see page 424</a> )	Name ( <a href="#">see page 405</a> ) of nature.
◆	Number ( <a href="#">see page 406</a> )	User-defined number linked with the load case .
◆	Quadratic ( <a href="#">see page 425</a> )	Indicator informing whether we are dealing with a quadratic combination .

❖	SeismicType (see page 425)	Seismic combination type .
❖	Signed (see page 425)	A flag indicating whether the combination is signed or not Available since version 1.7.
❖	Type (see page 406)	Load case type - allows one to differentiate between a simple case and combination at the level of the shared part of the interface .
❖	Uniqueld (see page 425)	Unique identifier of load case combinations Available since version 1.7.

## Public Methods

	Name	Description
❖	GetAnalysisParams (see page 426)	Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned). .
❖	SetAnalysisParams (see page 426)	Function sets analysis parameters for the load case. If the parameters specified cannot be set for this case (e.g. due to incompatibility between the case analysis type and the type of the object defining parameters), then zero value (False) is returned. .
❖	SetNatureExt (see page 427)	Function assigns - to the combination - the nature consistent with the indicated number of "action" described in the current regulations. .

### II.1.20.2 IRobotCaseCombination Fields

The fields of the IRobotCaseCombination class are listed here.

#### Public Fields

	Name	Description
❖	CaseFactors (see page 423)	List of pairs (simple case, factor) defining a combination .
❖	CombinationType (see page 423)	Combination type .
❖	IsAuxiliary (see page 424)	
❖	Label (see page 424)	
❖	NatureName (see page 424)	Name (see page 405) of nature.
❖	Quadratic (see page 425)	Indicator informing whether we are dealing with a quadratic combination .
❖	SeismicType (see page 425)	Seismic combination type .
❖	Signed (see page 425)	A flag indicating whether the combination is signed or not Available since version 1.7.
❖	Uniqueld (see page 425)	Unique identifier of load case combinations Available since version 1.7.

#### II.1.20.2.1 CaseFactors

##### C++

```
HRESULT get_CaseFactors( IRobotCaseFactorMngr** );
```

##### C#

```
public IRobotCaseFactorMngr CaseFactors { get; }
```

##### Visual Basic

```
Public ReadOnly CaseFactors As IRobotCaseFactorMngr
```

##### Description

List of pairs (simple case, factor) defining a combination .

### II.1.20.2.2 CombinationType

##### C++

```
HRESULT get_CombinationType( IRobotCombinationType* );
HRESULT put_CombinationType( IRobotCombinationType );
```

**C#**

```
public IRobotCombinationType CombinationType { get; set; }
```

**Visual Basic**

```
Public CombinationType As IRobotCombinationType
```

**Description**

Combination type.

### II.1.20.2.3 IsAuxiliary

**C++**

```
HRESULT get_IsAuxiliary(VARIANT_BOOL* );
HRESULT put_IsAuxiliary(VARIANT_BOOL);
```

**C#**

```
public bool IsAuxiliary { get; set; }
```

**Visual Basic**

```
Public IsAuxiliary As Boolean
```

**Version**

Available since version 8.1.

### II.1.20.2.4 Label

**C++**

```
HRESULT get_Label(BSTR* );
HRESULT put_Label(BSTR);
```

**C#**

```
public String Label { get; set; }
```

**Visual Basic**

```
Public Label As String
```

**Version**

Available since version 8.

### II.1.20.2.5 NatureName

**C++**

```
HRESULT get_NatureName(BSTR* );
```

**C#**

```
public String NatureName { get; }
```

**Visual Basic**

```
Public ReadOnly NatureName As String
```

**Description**

Name (see page 405) of nature.

**Version**

Available since version 3.

### II.1.20.2.6 Quadratic

#### C++

```
HRESULT get_Quadratic(VARIANT_BOOL* );
HRESULT put_Quadratic(VARIANT_BOOL);
```

#### C#

```
public bool Quadratic { get; set; }
```

#### Visual Basic

```
Public Quadratic As Boolean
```

#### Description

Indicator informing whether we are dealing with a quadratic combination .

### II.1.20.2.7 SeismicType

#### C++

```
HRESULT get_SeismicType(IRobotModeCombinationType* );
HRESULT put_SeismicType(IRobotModeCombinationType);
```

#### C#

```
public IRobotModeCombinationType SeismicType { get; set; }
```

#### Visual Basic

```
Public SeismicType As IRobotModeCombinationType
```

#### Description

Seismic combination type .

### II.1.20.2.8 Signed

#### C++

```
HRESULT get_Signed(VARIANT_BOOL* );
HRESULT put_Signed(VARIANT_BOOL);
```

#### C#

```
public bool Signed { get; set; }
```

#### Visual Basic

```
Public Signed As Boolean
```

#### Description

A flag indicating whether the combination is signed or not Available since version 1.7.

### II.1.20.2.9 UniqueId

#### C++

```
HRESULT get_UniqueId(long* );
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As Long
```

## Description

Unique identifier of load case combinations Available since version 1.7.

### II.1.20.3 IRobotCaseCombination Methods

The methods of the IRobotCaseCombination class are listed here.

#### Public Methods

	Name	Description
💡	GetAnalysisParams ( [ see page 426) ]	Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned). .
💡	SetAnalysisParams ( [ see page 426)	Function sets analysis parameters for the load case. If the parameters specified cannot be set for this case (e.g. due to incompatibility between the case analysis type and the type of the object defining parameters), then zero value (False) is returned. .
💡	SetNatureExt ( [ see page 427)	Function assigns - to the combination - the nature consistent with the indicated number of "action" described in the current regulations. .

#### II.1.20.3.1 GetAnalysisParams

##### C++

```
HRESULT GetAnalysisParams(IDispatch* ret);
```

##### C#

```
public IDispatch GetAnalysisParams();
```

##### Visual Basic

```
Public Function GetAnalysisParams() As IDispatch
```

#### Description

Function returns the object describing analysis parameters defined for the load case. The type of the returned object depends on the analysis type that has been assigned to this load case (e.g. for modal cases the object of the RobotModalAnalysisParams type is returned). .

#### Version

Available since version 2.5.

#### II.1.20.3.2 SetAnalysisParams

##### C++

```
HRESULT SetAnalysisParams(IDispatch _params, VARIANT_BOOL* ret);
```

##### C#

```
public bool SetAnalysisParams(IDispatch _params);
```

##### Visual Basic

```
Public Function SetAnalysisParams(_params As IDispatch) As Boolean
```

#### Description

Function sets analysis parameters for the load case. If the parameters specified cannot be set for this case (e.g. due to incompatibility between the case analysis type and the type of the object defining parameters), then zero value (False) is returned. .

**Version**

Available since version 2.5.

**II.1.20.3.3 SetNatureExt****C++**

```
HRESULT SetNatureExt(long _rgl_action);
```

**C#**

```
public void SetNatureExt(long _rgl_action);
```

**Visual Basic**

```
Public Sub SetNatureExt(_rgl_action As long)
```

**Description**

Function assigns - to the combination - the nature consistent with the indicated number of "action" described in the current regulations. .

**Version**

Available since version 3.

**II.1.21 IRobotCaseFactorMngr****Class Hierarchy****C++**

```
interface IRobotCaseFactorMngr : IDispatch;
```

**C#**

```
public interface IRobotCaseFactorMngr;
```

**Visual Basic**

```
Public Interface IRobotCaseFactorMngr
```

**Description**

Interface granting access to the list of pairs (load case, coefficient). All pairs from the list are numbered (indexed) with numbers from 1 to the number of items (Count ([see page 428](#))). .

**II.1.21.1 IRobotCaseFactorMngr Members**

The following tables list the members exposed by IRobotCaseFactorMngr.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 428</a> )	Number of managed items .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Delete ( <a href="#">see page 428</a> )	The function deletes items with the defined index. .
◆	Get ( <a href="#">see page 428</a> )	The function returns the item with the indicated index. .
◆	New ( <a href="#">see page 429</a> )	The function creates a new pair (load case, coefficient) and adds to the list of managed items. It returns the index of the created pair. .

## II.1.21.2 IRobotCaseFactorMngr Fields

The fields of the IRobotCaseFactorMngr class are listed here.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 428</a> )	Number of managed items .

### II.1.21.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of managed items .

## II.1.21.3 IRobotCaseFactorMngr Methods

The methods of the IRobotCaseFactorMngr class are listed here.

### Public Methods

	Name	Description
◆	Delete ( <a href="#">see page 428</a> )	The function deletes items with the defined index. .
◆	Get ( <a href="#">see page 428</a> )	The function returns the item with the indicated index. .
◆	New ( <a href="#">see page 429</a> )	The function creates a new pair (load case, coefficient) and adds to the list of managed items. It returns the index of the created pair. .

### II.1.21.3.1 Delete

#### C++

```
HRESULT Delete(long _index);
```

#### C#

```
public void Delete(long _index);
```

#### Visual Basic

```
Public Sub Delete(_index As long)
```

#### Description

The function deletes items with the defined index. .

### II.1.21.3.2 Get

#### C++

```
HRESULT Get(long _index, IRobotCaseFactor** ret);
```

#### C#

```
public IRobotCaseFactor Get(long _index);
```

**Visual Basic**

```
Public Function Get(_index As long) As IRobotCaseFactor
```

**Description**

The function returns the item with the indicated index. .

**II.1.21.3.3 New****C++**

```
HRESULT New(long _case_number, double _factor, long* ret);
```

**C#**

```
public long New(long _case_number, double _factor);
```

**Visual Basic**

```
Public Function New(_case_number As long, _factor As double) As long
```

**Description**

The function creates a new pair (load case, coefficient) and adds to the list of managed items. It returns the index of the created pair. .

**II.1.22 IRobotCombinationType****C++**

```
enum IRobotCombinationType;
```

**C#**

```
public enum IRobotCombinationType;
```

**Visual Basic**

```
Public Enum IRobotCombinationType
```

**Members**

Members	Description
I_CBT_SPC = 3	Special state.
I_CBT_ULS = 0	Ultimate limit state Available since version 2.0.
I_CBT_SLS = 1	Serviceability limit state Available since version 2.0.
I_CBT_ALS = 2	Accidental limit state Available since version 2.0.
I_CBT_SLS_EC_FRE = 4	Available since version 12.
I_CBT_SLS_EC_QPR = 5	Available since version 12.
I_CBT_SLS_EC_RAR = 6	Available since version 12.

**Description**

Type of load case combinations available in Robot. .

**II.1.23 IRobotCaseType****C++**

```
enum IRobotCaseType;
```

**C#**

```
public enum IRobotCaseType;
```

## Visual Basic

```
Public Enum IRobotCaseType
```

### Members

Members	Description
I_CT_SIMPLE = 0	Simple case .
I_CT_COMBINATION = 1	Combination .
I_CT_CODE_COMBINATION = 2	Code combination (defined according to a definition of the selected code) Available since version 1.7.
I_CT_MOBILE = 3	Moving load case. Available since version 2.5.

### Description

Among load cases, one distinguishes three basic types: a simple case, a combination and a code combination. .

## II.1.24 IRobotCaseFactor

### Class Hierarchy

#### C++

```
interface IRobotCaseFactor : IDispatch;
```

#### C#

```
public interface IRobotCaseFactor;
```

### Visual Basic

```
Public Interface IRobotCaseFactor
```

### Description

Interface describing a pair (load case, coefficient) used in the definition of load case combination. .

### II.1.24.1 IRobotCaseFactor Members

The following tables list the members exposed by IRobotCaseFactor.

#### Public Fields

	Name	Description
◆	CaseNumber (see page 430)	Load case number .
◆	Factor (see page 431)	(multiplication) factor.

### II.1.24.2 IRobotCaseFactor Fields

The fields of the IRobotCaseFactor class are listed here.

#### Public Fields

	Name	Description
◆	CaseNumber (see page 430)	Load case number .
◆	Factor (see page 431)	(multiplication) factor.

### II.1.24.2.1 CaseNumber

#### C++

```
HRESULT get_CaseNumber(long* );
HRESULT put_CaseNumber(long);
```

**C#**

```
public long CaseNumber { get; set; }
```

**Visual Basic**

```
Public CaseNumber As Long
```

**Description**

Load case number.

**II.1.24.2.2 Factor****C++**

```
HRESULT get_Factor(double*);  
HRESULT put_Factor(double);
```

**C#**

```
public double Factor { get; set; }
```

**Visual Basic**

```
Public Factor As Double
```

**Description**

(multiplication) factor.

**II.1.25 IRobotCaseCollection****Class Hierarchy****C++**

```
interface IRobotCaseCollection : IRobotCollection;
```

**C#**

```
public interface IRobotCaseCollection : IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotCaseCollection
```

**Description**

Collection of load cases.

**II.1.26 IRobotLimitState****C++**

```
enum IRobotLimitState;
```

**C#**

```
public enum IRobotLimitState;
```

**Visual Basic**

```
Public Enum IRobotLimitState
```

**Members**

Members	Description
I_LS_ULT = 0	Ultimate limit state.
I_LS_SLS = 1	Serviceability limit state.

I_LS_ALS = 2	Accidental state.
--------------	-------------------

**Description**

Limit state type.

**II.1.27 IRobotCaseAnalysisModesFilter****Class Hierarchy****C++**

```
interface IRobotCaseAnalysisModesFilter : IDispatch;
```

**C#**

```
public interface IRobotCaseAnalysisModesFilter;
```

**Visual Basic**

```
Public Interface IRobotCaseAnalysisModesFilter
```

**Description**

Definition of modes taken into account during structure dynamic analysis. .

**Version**

Available since version 2.5.

**II.1.27.1 IRobotCaseAnalysisModesFilter Members**

The following tables list the members exposed by IRobotCaseAnalysisModesFilter.

**Public Fields**

	Name	Description
❖	MassPercentage ( <a href="#">see page 432</a> )	
❖	Modes ( <a href="#">see page 433</a> )	List of modes taken into account during structure dynamic analysis. .
❖	Type ( <a href="#">see page 433</a> )	Filtering type.

**II.1.27.2 IRobotCaseAnalysisModesFilter Fields**

The fields of the IRobotCaseAnalysisModesFilter class are listed here.

**Public Fields**

	Name	Description
❖	MassPercentage ( <a href="#">see page 432</a> )	
❖	Modes ( <a href="#">see page 433</a> )	List of modes taken into account during structure dynamic analysis. .
❖	Type ( <a href="#">see page 433</a> )	Filtering type.

**II.1.27.2.1 MassPercentage****C++**

```
HRESULT get_MassPercentage(double* );
HRESULT put_MassPercentage(double);
```

**C#**

```
public double MassPercentage { get; set; }
```

**Visual Basic**

```
Public MassPercentage As Double
```

**Version**

Available since version 2.5.

**II.1.27.2.2 Modes****C++**

```
HRESULT get_Modes(BSTR* );
HRESULT put_Modes(BSTR);
```

**C#**

```
public BSTR Modes { get; set; }
```

**Visual Basic**

```
Public Modes As BSTR
```

**Description**

List of modes taken into account during structure dynamic analysis. .

**Version**

Available since version 2.5.

**II.1.27.2.3 Type****C++**

```
HRESULT get_Type(IRobotCaseAnalysisModesFilterType* );
HRESULT put_Type(IRobotCaseAnalysisModesFilterType);
```

**C#**

```
public IRobotCaseAnalysisModesFilterType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotCaseAnalysisModesFilterType
```

**Description**

Filtering type.

**Version**

Available since version 2.5.

**II.1.28 IRobotCaseAnalysisModesFilterType****C++**

```
enum IRobotCaseAnalysisModesFilterType;
```

**C#**

```
public enum IRobotCaseAnalysisModesFilterType;
```

**Visual Basic**

```
Public Enum IRobotCaseAnalysisModesFilterType
```

**Members**

Members	Description
I_CAMFT_NON_ACTIVE = 0	Inactive filtering. Available since version 2.5.

I_CAMFT_MASS_PERCENTAGE = 1	During structure analysis only these modes will be taken into account whose value of the mass participation factor is greater than the specified one . Available since version 2.5.
I_CAMFT_LIST_OF_MODES = 2	During structure analysis only the modes specified by the user will be taken into account. Available since version 2.5.

**Description**

Available methods of mode filtering.

**Version**

Available since version 2.5.

**II.1.29 IRobotDynamicAnalysisExcitationDirection****Class Hierarchy****C++**

```
interface IRobotDynamicAnalysisExcitationDirection : IDispatch;
```

**C#**

```
public interface IRobotDynamicAnalysisExcitationDirection;
```

**Visual Basic**

```
Public Interface IRobotDynamicAnalysisExcitationDirection
```

**Version**

Available since version 5.5.

**II.1.29.1 IRobotDynamicAnalysisExcitationDirection Members**

The following tables list the members exposed by IRobotDynamicAnalysisExcitationDirection.

**Public Fields**

	Name	Description
❖	CombType ( <a href="#">see page 435</a> )	
❖	Group1 ( <a href="#">see page 435</a> )	
❖	Group2 ( <a href="#">see page 436</a> )	
❖	Group3 ( <a href="#">see page 436</a> )	
❖	Lambda ( <a href="#">see page 436</a> )	
❖	Mi ( <a href="#">see page 436</a> )	
❖	NormalizedX ( <a href="#">see page 436</a> )	
❖	NormalizedY ( <a href="#">see page 437</a> )	
❖	NormalizedZ ( <a href="#">see page 437</a> )	
❖	QuadraticActive ( <a href="#">see page 437</a> )	
❖	QuadraticSigned ( <a href="#">see page 437</a> )	
❖	ResolutionActive ( <a href="#">see page 437</a> )	
❖	Rx ( <a href="#">see page 438</a> )	
❖	Ry ( <a href="#">see page 438</a> )	
❖	Rz ( <a href="#">see page 438</a> )	
❖	UseNormalized ( <a href="#">see page 438</a> )	
❖	X ( <a href="#">see page 438</a> )	
❖	Y ( <a href="#">see page 439</a> )	

	Z (see page 439)	
-----------------------------------------------------------------------------------	------------------	--

## II.1.29.2 IRobotDynamicAnalysisExcitationDirection Fields

The fields of the IRobotDynamicAnalysisExcitationDirection class are listed here.

### Public Fields

	Name	Description
◆	CombType (see page 435)	
◆	Group1 (see page 435)	
◆	Group2 (see page 436)	
◆	Group3 (see page 436)	
◆	Lambda (see page 436)	
◆	Mi (see page 436)	
◆	NormalizedX (see page 436)	
◆	NormalizedY (see page 437)	
◆	NormalizedZ (see page 437)	
◆	QuadraticActive (see page 437)	
◆	QuadraticSigned (see page 437)	
◆	ResolutionActive (see page 437)	
◆	Rx (see page 438)	
◆	Ry (see page 438)	
◆	Rz (see page 438)	
◆	UseNormalized (see page 438)	
◆	X (see page 438)	
◆	Y (see page 439)	
◆	Z (see page 439)	

### II.1.29.2.1 CombType

#### C++

```
HRESULT get_CombType(IRobotModeCombinationType* );
HRESULT put_CombType(IRobotModeCombinationType);
```

#### C#

```
public IRobotModeCombinationType CombType { get; set; }
```

#### Visual Basic

```
Public CombType As IRobotModeCombinationType
```

### II.1.29.2.2 Group1

#### C++

```
HRESULT get_Group1(VARIANT_BOOL* );
HRESULT put_Group1(VARIANT_BOOL);
```

#### C#

```
public bool Group1 { get; set; }
```

#### Visual Basic

```
Public Group1 As Boolean
```

**II.1.29.2.3 Group2****C++**

```
HRESULT get_Group2(VARIANT_BOOL* );
HRESULT put_Group2(VARIANT_BOOL);
```

**C#**

```
public bool Group2 { get; set; }
```

**Visual Basic**

```
Public Group2 As Boolean
```

**II.1.29.2.4 Group3****C++**

```
HRESULT get_Group3(VARIANT_BOOL* );
HRESULT put_Group3(VARIANT_BOOL);
```

**C#**

```
public bool Group3 { get; set; }
```

**Visual Basic**

```
Public Group3 As Boolean
```

**II.1.29.2.5 Lambda****C++**

```
HRESULT get_Lambda(double* );
HRESULT put_Lambda(double);
```

**C#**

```
public double Lambda { get; set; }
```

**Visual Basic**

```
Public Lambda As Double
```

**II.1.29.2.6 Mi****C++**

```
HRESULT get_Mi(double* );
HRESULT put_Mi(double);
```

**C#**

```
public double Mi { get; set; }
```

**Visual Basic**

```
Public Mi As Double
```

**II.1.29.2.7 NormalizedX****C++**

```
HRESULT get_NormalizedX(double* );
```

**C#**

```
public double NormalizedX { get; }
```

**Visual Basic**

```
Public ReadOnly NormalizedX As Double
```

### II.1.29.2.8 NormalizedY

**C++**

```
HRESULT get_NormalizedY(double*);
```

**C#**

```
public double NormalizedY { get; }
```

**Visual Basic**

```
Public ReadOnly NormalizedY As double
```

### II.1.29.2.9 NormalizedZ

**C++**

```
HRESULT get_NormalizedZ(double*);
```

**C#**

```
public double NormalizedZ { get; }
```

**Visual Basic**

```
Public ReadOnly NormalizedZ As double
```

### II.1.29.2.10 QuadraticActive

**C++**

```
HRESULT get_QuadraticActive(VARIANT_BOOL*);  
HRESULT put_QuadraticActive(VARIANT_BOOL);
```

**C#**

```
public bool QuadraticActive { get; set; }
```

**Visual Basic**

```
Public QuadraticActive As Boolean
```

### II.1.29.2.11 QuadraticSigned

**C++**

```
HRESULT get_QuadraticSigned(VARIANT_BOOL*);  
HRESULT put_QuadraticSigned(VARIANT_BOOL);
```

**C#**

```
public bool QuadraticSigned { get; set; }
```

**Visual Basic**

```
Public QuadraticSigned As Boolean
```

### II.1.29.2.12 ResolutionActive

**C++**

```
HRESULT get_ResolutionActive(VARIANT_BOOL*);  
HRESULT put_ResolutionActive(VARIANT_BOOL);
```

**C#**

```
public bool ResolutionActive { get; set; }
```

**Visual Basic**

```
Public ResolutionActive As Boolean
```

### II.1.29.2.13 Rx

**C++**

```
HRESULT get_Rx(double*);  
HRESULT put_Rx(double);
```

**C#**

```
public double Rx { get; set; }
```

**Visual Basic**

```
Public Rx As Double
```

### II.1.29.2.14 Ry

**C++**

```
HRESULT get_Ry(double*);  
HRESULT put_Ry(double);
```

**C#**

```
public double Ry { get; set; }
```

**Visual Basic**

```
Public Ry As Double
```

### II.1.29.2.15 Rz

**C++**

```
HRESULT get_Rz(double*);  
HRESULT put_Rz(double);
```

**C#**

```
public double Rz { get; set; }
```

**Visual Basic**

```
Public Rz As Double
```

### II.1.29.2.16 UseNormalized

**C++**

```
HRESULT get_UseNormalized(VARIANT_BOOL*);  
HRESULT put_UseNormalized(VARIANT_BOOL);
```

**C#**

```
public bool UseNormalized { get; set; }
```

**Visual Basic**

```
Public UseNormalized As Boolean
```

### II.1.29.2.17 X

**C++**

```
HRESULT get_X(double*);  
HRESULT put_X(double);
```

**C#**

```
public double X { get; set; }
```

**Visual Basic**

```
Public X As Double
```

## II.1.29.2.18 Y

**C++**

```
HRESULT get_Y(double* );
HRESULT put_Y(double);
```

**C#**

```
public double Y { get; set; }
```

**Visual Basic**

```
Public Y As Double
```

## II.1.29.2.19 Z

**C++**

```
HRESULT get_Z(double* );
HRESULT put_Z(double);
```

**C#**

```
public double Z { get; set; }
```

**Visual Basic**

```
Public Z As Double
```

## II.1.30 IRobotDynamicAnalysisDamping

**Class Hierarchy****C++**

```
interface IRobotDynamicAnalysisDamping : IDispatch;
```

**C#**

```
public interface IRobotDynamicAnalysisDamping;
```

**Visual Basic**

```
Public Interface IRobotDynamicAnalysisDamping
```

**Description**

Damping parameters for the dynamic analysis.

**Version**

Available since version 9.7.

### II.1.30.1 IRobotDynamicAnalysisDamping Members

The following tables list the members exposed by IRobotDynamicAnalysisDamping.

**Public Fields**

	Name	Description
◆	ConstValue (see page 440)	Constant damping value.
◆	Type (see page 440)	Damping type.

**Public Methods**

	Name	Description
◆	GetModeDamping (see page 441)	Function returns a damping value for the specified vibration mode.
◆	GetModes (see page 441)	Function returns a table with numbers of vibration modes with defined damping values.

	RemoveModeDamping ( <a href="#">see page 441</a> )	Function deletes a damping value for the specified vibration mode.
	SetModeDamping ( <a href="#">see page 442</a> )	Function sets a damping value for the specified vibration mode.

## II.1.30.2 IRobotDynamicAnalysisDamping Fields

The fields of the IRobotDynamicAnalysisDamping class are listed here.

### Public Fields

	Name	Description
	ConstValue ( <a href="#">see page 440</a> )	Constant damping value.
	Type ( <a href="#">see page 440</a> )	Damping type.

### II.1.30.2.1 ConstValue

#### C++

```
HRESULT get_ConstValue(double* );
HRESULT put_ConstValue(double);
```

#### C#

```
public double ConstValue { get; set; }
```

#### Visual Basic

```
Public ConstValue As Double
```

#### Description

Constant damping value.

#### Version

Available since version 9.7.

### II.1.30.2.2 Type

#### C++

```
HRESULT get_Type(IRobotDynamicAnalysisDampingType* );
HRESULT put_Type(IRobotDynamicAnalysisDampingType);
```

#### C#

```
public IRobotDynamicAnalysisDampingType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotDynamicAnalysisDampingType
```

#### Description

Damping type.

#### Version

Available since version 9.7.

## II.1.30.3 IRobotDynamicAnalysisDamping Methods

The methods of the IRobotDynamicAnalysisDamping class are listed here.

### Public Methods

	Name	Description
	GetModeDamping ( <a href="#">see page 441</a> )	Function returns a damping value for the specified vibration mode.

	GetModes ( <a href="#">see page 441</a> )	Function returns a table with numbers of vibration modes with defined damping values.
	RemoveModeDamping ( <a href="#">see page 441</a> )	Function deletes a damping value for the specified vibration mode.
	SetModeDamping ( <a href="#">see page 442</a> )	Function sets a damping value for the specified vibration mode.

### II.1.30.3.1 GetModeDamping

**C++**

```
HRESULT GetModeDamping( long _mode, double* ret );
```

**C#**

```
public double GetModeDamping( long _mode );
```

**Visual Basic**

```
Public Function GetModeDamping( _mode As long ) As double
```

**Description**

Function returns a damping value for the specified vibration mode.

**Version**

Available since version 9.7.

### II.1.30.3.2 GetModes

**C++**

```
HRESULT GetModes( SAFEARRAY** ret );
```

**C#**

```
public Array GetModes( );
```

**Visual Basic**

```
Public Function GetModes() As long()
```

**Description**

Function returns a table with numbers of vibration modes with defined damping values.

**Version**

Available since version 9.7.

### II.1.30.3.3 RemoveModeDamping

**C++**

```
HRESULT RemoveModeDamping( long _mode );
```

**C#**

```
public void RemoveModeDamping( long _mode );
```

**Visual Basic**

```
Public Sub RemoveModeDamping( _mode As long )
```

**Description**

Function deletes a damping value for the specified vibration mode.

**Version**

Available since version 9.7.

**II.1.30.3.4 SetModeDamping****C++**

```
HRESULT SetModeDamping(long _mode, double _damp_value);
```

**C#**

```
public void SetModeDamping(long _mode, double _damp_value);
```

**Visual Basic**

```
Public Sub SetModeDamping(_mode As long, _damp_value As double)
```

**Description**

Function sets a damping value for the specified vibration mode.

**Version**

Available since version 9.7.

**II.1.31 IRobotDynamicAnalysisDampingType****C++**

```
enum IRobotDynamicAnalysisDampingType;
```

**C#**

```
public enum IRobotDynamicAnalysisDampingType;
```

**Visual Basic**

```
Public Enum IRobotDynamicAnalysisDampingType
```

**Members**

Members	Description
I_DADT_NONE = 0	Without damping. Available since version 9.7.
I_DADT_CONSTANT = 1	Constant damping. Available since version 9.7.
I_DADT_VARIABLE = 3	Damping that varies for each mode. Available since version 9.7.
I_DADT_MODAL = 2	Modal damping. Available since version 9.7.

**Version**

Available since version 9.7.

**II.1.32 IRobotCaseRelatedValueType****C++**

```
enum IRobotCaseRelatedValueType;
```

**C#**

```
public enum IRobotCaseRelatedValueType;
```

## Visual Basic

```
Public Enum IRobotCaseRelatedValueType
```

### Members

Members	Description
I_CRVT_FRF_FREQUENCY = 1	Frequency for FRF analysis. Available since version 9.7.

### Description

A set of identifiers for types of values that can be associated with the load case.

### Version

Available since version 9.7.

## II.1.33 Parameters of simplified seismic analysis

Parameters of seismic analysis (method of static equivalent lateral force).

### Version

Available since version 12.5.

### Enumerations

	Name	Description
⊕	IRobotSELFSismicLevelDefinitionMethod ( <a href="#">see page 450</a> )	
⊕	IRobotSELFSismicTBaseMethod ( <a href="#">see page 451</a> )	
⊕	IRobotSELFSismic_ASCE_7_10_SiteClassType ( <a href="#">see page 454</a> )	
⊕	IRobotSELFSismic_ASCE_7_10_StructureType ( <a href="#">see page 458</a> )	
⊕	IRobotSELFSismic_EC_8_StructureType ( <a href="#">see page 462</a> )	
⊕	IRobotSELFSismic_EC_8_SiteClass ( <a href="#">see page 462</a> )	
⊕	IRobotSELFSismic_EC_8_SpectrumType ( <a href="#">see page 463</a> )	
⊕	IRobotSELFSismic_AS_1170_4_SoilCategoryType ( <a href="#">see page 466</a> )	
⊕	IRobotSELFSismic_AS_1170_4_StructureType ( <a href="#">see page 467</a> )	
⊕	IRobotSELFSismic_AS_1170_4_ProbabilityType ( <a href="#">see page 467</a> )	

### Interfaces

	Name	Description
⊕	IRobotSELFSismicEngine ( <a href="#">see page 444</a> )	Generator of load cases for the simplified seismic analysis.
⊕	IRobotSELFSismicGenerationParams ( <a href="#">see page 445</a> )	Parameters for generation of load cases.
⊕	IRobotSELFSismicStructureParams ( <a href="#">see page 448</a> )	
⊕	IRobotSELFSismicAnalysisParams ( <a href="#">see page 451</a> )	Parameters of the seismic analysis (method of static equivalent lateral force).
⊕	IRobotSELFSismic_ASCE_7_10_Params ( <a href="#">see page 454</a> )	Parameters of the ASCE 7-10 code for the simplified seismic analysis.

	IRobotSELFSeismic_EC_8_Params (see page 458)	Parameters of EC 8 for the simplified seismic analysis.
	IRobotSELFSeismic_AS_1170_4_Params (see page 463)	Parameters of the AS 1170-4 code for the simplified seismic analysis.

## II.1.33.1 IRobotSELFSeismicEngine

### Class Hierarchy

#### C++

```
interface IRobotSELFSeismicEngine : IDispatch;
```

#### C#

```
public interface IRobotSELFSeismicEngine;
```

### Visual Basic

```
Public Interface IRobotSELFSeismicEngine
```

### Description

Generator of load cases for the simplified seismic analysis.

### Version

Available since version 12.5.

## II.1.33.1.1 IRobotSELFSeismicEngine Members

The following tables list the members exposed by IRobotSELFSeismicEngine.

### Public Fields

	Name	Description
	GenerationParams (see page 444)	Parameters for generation of load cases for the simplified seismic analysis.

### Public Methods

	Name	Description
	GenerateLoadCases (see page 445)	The function generates new load cases for the simplified seismic analysis based on the parameters of generation set currently. The function returns the number of generated cases.

## II.1.33.1.2 IRobotSELFSeismicEngine Fields

The fields of the IRobotSELFSeismicEngine class are listed here.

### Public Fields

	Name	Description
	GenerationParams (see page 444)	Parameters for generation of load cases for the simplified seismic analysis.

## II.1.33.1.2.1 GenerationParams

#### C++

```
HRESULT get_GenerationParams(IRobotSELFSeismicGenerationParams**);
```

#### C#

```
public IRobotSELFSeismicGenerationParams GenerationParams { get; }
```

**Visual Basic**

```
Public ReadOnly GenerationParams As IRobotSELFSeismicGenerationParams
```

**Description**

Parameters for generation of load cases for the simplified seismic analysis.

**Version**

Available since version 12.5.

**II.1.33.1.3 IRobotSELFSeismicEngine Methods**

The methods of the IRobotSELFSeismicEngine class are listed here.

**Public Methods**

	Name	Description
💡	GenerateLoadCases (see page 445)	The function generates new load cases for the simplified seismic analysis based on the parameters of generation set currently. The function returns the number of generated cases.

**II.1.33.1.3.1 GenerateLoadCases****C++**

```
HRESULT GenerateLoadCases(long* ret);
```

**C#**

```
public long GenerateLoadCases();
```

**Visual Basic**

```
Public Function GenerateLoadCases() As long
```

**Description**

The function generates new load cases for the simplified seismic analysis based on the parameters of generation set currently. The function returns the number of generated cases.

**Version**

Available since version 12.5.

**II.1.33.2 IRobotSELFSeismicGenerationParams****Class Hierarchy****C++**

```
interface IRobotSELFSeismicGenerationParams : IDispatch;
```

**C#**

```
public interface IRobotSELFSeismicGenerationParams;
```

**Visual Basic**

```
Public Interface IRobotSELFSeismicGenerationParams
```

**Description**

Parameters for generation of load cases.

**Version**

Available since version 12.5.

### II.1.33.2.1 IRobotSELFSeismicGenerationParams Members

The following tables list the members exposed by IRobotSELFSeismicGenerationParams.

#### Public Fields

	Name	Description
◆	CodeName ( <a href="#">see page 446</a> )	
◆	CodeNumber ( <a href="#">see page 446</a> )	
◆	ExcitationDir ( <a href="#">see page 447</a> )	
◆	ModalCaseParams ( <a href="#">see page 447</a> )	
◆	SeismicParams ( <a href="#">see page 447</a> )	
◆	TBaseMethod ( <a href="#">see page 447</a> )	

### II.1.33.2.2 IRobotSELFSeismicGenerationParams Fields

The fields of the IRobotSELFSeismicGenerationParams class are listed here.

#### Public Fields

	Name	Description
◆	CodeName ( <a href="#">see page 446</a> )	
◆	CodeNumber ( <a href="#">see page 446</a> )	
◆	ExcitationDir ( <a href="#">see page 447</a> )	
◆	ModalCaseParams ( <a href="#">see page 447</a> )	
◆	SeismicParams ( <a href="#">see page 447</a> )	
◆	TBaseMethod ( <a href="#">see page 447</a> )	

### II.1.33.2.2.1 CodeName

#### C++

```
HRESULT get_CodeName(BSTR* );
HRESULT put_CodeName(BSTR);
```

#### C#

```
public String CodeName { get; set; }
```

#### Visual Basic

```
Public CodeName As String
```

#### Version

Available since version 12.5.

### II.1.33.2.2.2 CodeNumber

#### C++

```
HRESULT get_CodeNumber(long* );
HRESULT put_CodeNumber(long);
```

#### C#

```
public long CodeNumber { get; set; }
```

#### Visual Basic

```
Public CodeNumber As long
```

**Version**

Available since version 12.5.

**II.1.33.2.2.3 ExcitationDir****C++**

```
HRESULT get_ExcitationDir(IRobotDynamicAnalysisExcitationDirection**);
```

**C#**

```
public IRobotDynamicAnalysisExcitationDirection ExcitationDir { get; }
```

**Visual Basic**

```
Public ReadOnly ExcitationDir As IRobotDynamicAnalysisExcitationDirection
```

**Version**

Available since version 12.5.

**II.1.33.2.2.4 ModalCaseParams****C++**

```
HRESULT get_ModalCaseParams(IRobotModalAnalysisParams**);
```

**C#**

```
public IRobotModalAnalysisParams ModalCaseParams { get; }
```

**Visual Basic**

```
Public ReadOnly ModalCaseParams As IRobotModalAnalysisParams
```

**Version**

Available since version 12.5.

**II.1.33.2.2.5 SeismicParams****C++**

```
HRESULT get_SeismicParams(IRobotSELFSeismicAnalysisParams**);
```

**C#**

```
public IRobotSELFSeismicAnalysisParams SeismicParams { get; }
```

**Visual Basic**

```
Public ReadOnly SeismicParams As IRobotSELFSeismicAnalysisParams
```

**Version**

Available since version 12.5.

**II.1.33.2.2.6 TBaseMethod****C++**

```
HRESULT get_TBaseMethod(IRobotSELFSeismicTBaseMethod*);  
HRESULT put_TBaseMethod(IRobotSELFSeismicTBaseMethod*);
```

**C#**

```
public IRobotSELFSeismicTBaseMethod TBaseMethod { get; set; }
```

**Visual Basic**

```
Public TBaseMethod As IRobotSELFSeismicTBaseMethod
```

## Version

Available since version 12.5.

### II.1.33.3 IRobotSELFSeismicStructureParams

#### Class Hierarchy

##### C++

```
interface IRobotSELFSeismicStructureParams : IDispatch;
```

##### C#

```
public interface IRobotSELFSeismicStructureParams;
```

#### Visual Basic

```
Public Interface IRobotSELFSeismicStructureParams
```

#### Version

Available since version 12.5.

### II.1.33.3.1 IRobotSELFSeismicStructureParams Members

The following tables list the members exposed by IRobotSELFSeismicStructureParams.

#### Public Fields

	Name	Description
◆	BaseLevelCoordZ ( <a href="#">see page 448</a> )	
◆	BaseLevelDefMethod ( <a href="#">see page 449</a> )	
◆	BaseLevelStorey ( <a href="#">see page 449</a> )	Story no.
◆	TopLevelCoordZ ( <a href="#">see page 449</a> )	
◆	TopLevelDefMethod ( <a href="#">see page 450</a> )	
◆	TopLevelStorey ( <a href="#">see page 450</a> )	Story no.

### II.1.33.3.2 IRobotSELFSeismicStructureParams Fields

The fields of the IRobotSELFSeismicStructureParams class are listed here.

#### Public Fields

	Name	Description
◆	BaseLevelCoordZ ( <a href="#">see page 448</a> )	
◆	BaseLevelDefMethod ( <a href="#">see page 449</a> )	
◆	BaseLevelStorey ( <a href="#">see page 449</a> )	Story no.
◆	TopLevelCoordZ ( <a href="#">see page 449</a> )	
◆	TopLevelDefMethod ( <a href="#">see page 450</a> )	
◆	TopLevelStorey ( <a href="#">see page 450</a> )	Story no.

### II.1.33.3.2.1 BaseLevelCoordZ

##### C++

```
HRESULT get_BaseLevelCoordZ(double* );
HRESULT put_BaseLevelCoordZ(double);
```

**C#**

```
public double BaseLevelCoordZ { get; set; }
```

**Visual Basic**

```
Public BaseLevelCoordZ As Double
```

**Version**

Available since version 12.5.

### II.1.33.3.2.2 BaseLevelDefMethod

**C++**

```
HRESULT get_BaseLevelDefMethod(IRobotSELFS seismicLevelDefinitionMethod*);  
HRESULT put_BaseLevelDefMethod(IRobotSELFS seismicLevelDefinitionMethod);
```

**C#**

```
public IRobotSELFS seismicLevelDefinitionMethod BaseLevelDefMethod { get; set; }
```

**Visual Basic**

```
Public BaseLevelDefMethod As IRobotSELFS seismicLevelDefinitionMethod
```

**Version**

Available since version 12.5.

### II.1.33.3.2.3 BaseLevelStorey

**C++**

```
HRESULT get_BaseLevelStorey(long*);  
HRESULT put_BaseLevelStorey(long);
```

**C#**

```
public long BaseLevelStorey { get; set; }
```

**Visual Basic**

```
Public BaseLevelStorey As Long
```

**Description**

Story no.

**Version**

Available since version 12.5.

### II.1.33.3.2.4 TopLevelCoordZ

**C++**

```
HRESULT get_TopLevelCoordZ(double*);  
HRESULT put_TopLevelCoordZ(double);
```

**C#**

```
public double TopLevelCoordZ { get; set; }
```

**Visual Basic**

```
Public TopLevelCoordZ As Double
```

**Version**

Available since version 12.5.

### II.1.33.3.2.5 TopLevelDefMethod

#### C++

```
HRESULT get_TopLevelDefMethod(IRobotSELFS seismicLevelDefinitionMethod* );
HRESULT put_TopLevelDefMethod(IRobotSELFS seismicLevelDefinitionMethod);
```

#### C#

```
public IRobotSELFS seismicLevelDefinitionMethod TopLevelDefMethod { get; set; }
```

#### Visual Basic

```
Public TopLevelDefMethod As IRobotSELFS seismicLevelDefinitionMethod
```

#### Version

Available since version 12.5.

### II.1.33.3.2.6 TopLevelStorey

#### C++

```
HRESULT get_TopLevelStorey(long* );
HRESULT put_TopLevelStorey(long);
```

#### C#

```
public long TopLevelStorey { get; set; }
```

#### Visual Basic

```
Public TopLevelStorey As long
```

#### Description

Story no.

#### Version

Available since version 12.5.

### II.1.33.4 IRobotSELFS seismicLevelDefinitionMethod

#### C++

```
enum IRobotSELFS seismicLevelDefinitionMethod;
```

#### C#

```
public enum IRobotSELFS seismicLevelDefinitionMethod;
```

#### Visual Basic

```
Public Enum IRobotSELFS seismicLevelDefinitionMethod
```

#### Members

Members	Description
I_SSLLDM_AUTOMATIC = 0	Available since version 12.5.
I_SSLLDM_BY_Z_COORDINATE = 1	Available since version 12.5.
I_SSLLDM_BY_STOREY = 2	Available since version 12.5.

#### Version

Available since version 12.5.

## II.1.33.5 IRobotSELFSeismicTBaseMethod

### C++

```
enum IRobotSELFSeismicTBaseMethod;
```

### C#

```
public enum IRobotSELFSeismicTBaseMethod;
```

### Visual Basic

```
Public Enum IRobotSELFSeismicTBaseMethod
```

### Members

Members	Description
I_SSTBM_APPROXIMATE_BY_CODE = 0	Available since version 12.5.
I_SSTBM_USER_DEFINED = 1	Available since version 12.5.
I_SSTBM_PRECISE_MODAL = 2	Available since version 12.5.
I_SSTBM_PRECISE_MODAL_WITH_MASS = 3	Available since version 12.5.

### Version

Available since version 12.5.

## II.1.33.6 IRobotSELFSeismicAnalysisParams

### Class Hierarchy

### C++

```
interface IRobotSELFSeismicAnalysisParams : IDispatch;
```

### C#

```
public interface IRobotSELFSeismicAnalysisParams;
```

### Visual Basic

```
Public Interface IRobotSELFSeismicAnalysisParams
```

### Description

Parameters of the seismic analysis (method of static equivalent lateral force).

### Version

Available since version 12.5.

## II.1.33.6.1 IRobotSELFSeismicAnalysisParams Members

The following tables list the members exposed by IRobotSELFSeismicAnalysisParams.

### Public Fields

	Name	Description
◆	CodeName (↗ see page 452)	
◆	CodeNumber (↗ see page 452)	
◆	CodeParams (↗ see page 452)	Object defining code-specific parameters; the object type corresponds to a selected code, for example for the ASCE 7-10 code the object of RobotSELFSeismic_ASCE_7_10_Params class is available;.
◆	Eccentricities (↗ see page 453)	
◆	StructureParams (↗ see page 453)	
◆	TBaseMethod (↗ see page 453)	

## Public Methods

	Name	Description
💡	GetExcitationDir (see page 454)	

### II.1.33.6.2 IRobotSELFSeismicAnalysisParams Fields

The fields of the IRobotSELFSeismicAnalysisParams class are listed here.

#### Public Fields

	Name	Description
💡	CodeName (see page 452)	
💡	CodeNumber (see page 452)	
💡	CodeParams (see page 452)	Object defining code-specific parameters; the object type corresponds to a selected code, for example for the ASCE 7-10 code the object of RobotSELFSeismic_ASCE_7_10_Params class is available;.
💡	Eccentricities (see page 453)	
💡	StructureParams (see page 453)	
💡	TBaseMethod (see page 453)	

#### II.1.33.6.2.1 CodeName

##### C++

```
HRESULT get_CodeName(BSTR*);
```

##### C#

```
public String CodeName { get; }
```

##### Visual Basic

```
Public ReadOnly CodeName As String
```

##### Version

Available since version 12.5.

#### II.1.33.6.2.2 CodeNumber

##### C++

```
HRESULT get_CodeNumber(long*);
```

##### C#

```
public long CodeNumber { get; }
```

##### Visual Basic

```
Public ReadOnly CodeNumber As long
```

##### Version

Available since version 12.5.

#### II.1.33.6.2.3 CodeParams

##### C++

```
HRESULT get_CodeParams(IDispatch*);
```

##### C#

```
public IDispatch CodeParams { get; }
```

**Visual Basic**

```
Public ReadOnly CodeParams As IDispatch
```

**Description**

Object defining code-specific parameters; the object type corresponds to a selected code, for example for the ASCE 7-10 code the object of RobotSELFSeismic\_ASCE\_7\_10\_Params class is available;

**Version**

Available since version 12.5.

**II.1.33.6.2.4 Eccentricities****C++**

```
HRESULT get_Eccentricities(IRobotMassEccentricities**);
```

**C#**

```
public IRobotMassEccentricities Eccentricities { get; }
```

**Visual Basic**

```
Public ReadOnly Eccentricities As IRobotMassEccentricities
```

**Version**

Available since version 12.5.

**II.1.33.6.2.5 StructureParams****C++**

```
HRESULT get_StructureParams(IRobotSELFS seismicStructureParams**);
```

**C#**

```
public IRobotSELFS seismicStructureParams StructureParams { get; }
```

**Visual Basic**

```
Public ReadOnly StructureParams As IRobotSELFS seismicStructureParams
```

**Version**

Available since version 12.5.

**II.1.33.6.2.6 TBaseMethod****C++**

```
HRESULT get_TBaseMethod(IRobotSELFS seismicTBaseMethod*);
```

**C#**

```
public IRobotSELFS seismicTBaseMethod TBaseMethod { get; }
```

**Visual Basic**

```
Public ReadOnly TBaseMethod As IRobotSELFS seismicTBaseMethod
```

**Version**

Available since version 12.5.

**II.1.33.6.3 IRobotSELFS seismicAnalysisParams Methods**

The methods of the IRobotSELFS seismicAnalysisParams class are listed here.

## Public Methods

	Name	Description
💡	GetExcitationDir (see page 454)	

### II.1.33.6.3.1 GetExcitationDir

#### C++

```
HRESULT GetExcitationDir(IRobotDynamicAnalysisExcitationDirection** ret);
```

#### C#

```
public IRobotDynamicAnalysisExcitationDirection GetExcitationDir();
```

#### Visual Basic

```
Public Function GetExcitationDir() As IRobotDynamicAnalysisExcitationDirection
```

#### Version

Available since version 12.5.

### II.1.33.7 IRobotSELFS seismic\_ASCE\_7\_10\_SiteClassType

#### C++

```
enum IRobotSELFS seismic_ASCE_7_10_SiteClassType;
```

#### C#

```
public enum IRobotSELFS seismic_ASCE_7_10_SiteClassType;
```

#### Visual Basic

```
Public Enum IRobotSELFS seismic_ASCE_7_10_SiteClassType
```

#### Members

Members	Description
I_SSSCT_ASCE_7_10_A = 1	Available since version 12.5.
I_SSSCT_ASCE_7_10_B = 2	Available since version 12.5.
I_SSSCT_ASCE_7_10_C = 3	Available since version 12.5.
I_SSSCT_ASCE_7_10_D = 4	Available since version 12.5.
I_SSSCT_ASCE_7_10_E = 5	Available since version 12.5.
I_SSSCT_ASCE_7_10_F = 6	Available since version 12.5.

#### Version

Available since version 12.5.

### II.1.33.8 IRobotSELFS seismic\_ASCE\_7\_10\_Params

#### Class Hierarchy

#### C++

```
interface IRobotSELFS seismic_ASCE_7_10_Params : IDispatch;
```

#### C#

```
public interface IRobotSELFS seismic_ASCE_7_10_Params;
```

#### Visual Basic

```
Public Interface IRobotSELFS seismic_ASCE_7_10_Params
```

## Description

Parameters of the ASCE 7-10 code for the simplified seismic analysis.

## Version

Available since version 12.5.

### II.1.33.8.1 IRobotSELFSeismic\_ASCE\_7\_10\_Params Members

The following tables list the members exposed by IRobotSELFSeismic\_ASCE\_7\_10\_Params.

#### Public Fields

	Name	Description
◆	I ( <a href="#">see page 455</a> )	
◆	R ( <a href="#">see page 456</a> )	
◆	S1 ( <a href="#">see page 456</a> )	
◆	SiteClass ( <a href="#">see page 456</a> )	
◆	Ss ( <a href="#">see page 456</a> )	
◆	StructureTypeX ( <a href="#">see page 457</a> )	
◆	StructureTypeY ( <a href="#">see page 457</a> )	
◆	TBaseValueX ( <a href="#">see page 457</a> )	
◆	TBaseValueY ( <a href="#">see page 457</a> )	
◆	TL ( <a href="#">see page 458</a> )	

### II.1.33.8.2 IRobotSELFSeismic\_ASCE\_7\_10\_Params Fields

The fields of the IRobotSELFSeismic\_ASCE\_7\_10\_Params class are listed here.

#### Public Fields

	Name	Description
◆	I ( <a href="#">see page 455</a> )	
◆	R ( <a href="#">see page 456</a> )	
◆	S1 ( <a href="#">see page 456</a> )	
◆	SiteClass ( <a href="#">see page 456</a> )	
◆	Ss ( <a href="#">see page 456</a> )	
◆	StructureTypeX ( <a href="#">see page 457</a> )	
◆	StructureTypeY ( <a href="#">see page 457</a> )	
◆	TBaseValueX ( <a href="#">see page 457</a> )	
◆	TBaseValueY ( <a href="#">see page 457</a> )	
◆	TL ( <a href="#">see page 458</a> )	

### II.1.33.8.2.1 I

#### C++

```
HRESULT get_I(double* );
HRESULT put_I(double);
```

#### C#

```
public double I { get; set; }
```

#### Visual Basic

```
Public I As Double
```

## Version

Available since version 12.5.

### II.1.33.8.2.2 R

#### C++

```
HRESULT get_R(double*);  
HRESULT put_R(double);
```

#### C#

```
public double R { get; set; }
```

#### Visual Basic

```
Public R As Double
```

#### Version

Available since version 12.5.

### II.1.33.8.2.3 S1

#### C++

```
HRESULT get_S1(double*);  
HRESULT put_S1(double);
```

#### C#

```
public double S1 { get; set; }
```

#### Visual Basic

```
Public S1 As Double
```

#### Version

Available since version 12.5.

### II.1.33.8.2.4 SiteClass

#### C++

```
HRESULT get_SiteClass(IRobotSELFS seismic_ASCE_7_10_SiteClassType*);  
HRESULT put_SiteClass(IRobotSELFS seismic_ASCE_7_10_SiteClassType);
```

#### C#

```
public IRobotSELFS seismic_ASCE_7_10_SiteClassType SiteClass { get; set; }
```

#### Visual Basic

```
Public SiteClass As IRobotSELFS seismic_ASCE_7_10_SiteClassType
```

#### Version

Available since version 12.5.

### II.1.33.8.2.5 Ss

#### C++

```
HRESULT get_Ss(double*);  
HRESULT put_Ss(double);
```

#### C#

```
public double Ss { get; set; }
```

#### Visual Basic

```
Public Ss As Double
```

**Version**

Available since version 12.5.

**II.1.33.8.2.6 StructureTypeX****C++**

```
HRESULT get_StructureTypeX(IRobotSELFS seismicic_ASCE_7_10_StructureType* );
HRESULT put_StructureTypeX(IRobotSELFS seismicic_ASCE_7_10_StructureType);
```

**C#**

```
public IRobotSELFS seismicic_ASCE_7_10_StructureType StructureTypeX { get; set; }
```

**Visual Basic**

```
Public StructureTypeX As IRobotSELFS seismicic_ASCE_7_10_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.8.2.7 StructureTypeY****C++**

```
HRESULT get_StructureTypeY(IRobotSELFS seismicic_ASCE_7_10_StructureType* );
HRESULT put_StructureTypeY(IRobotSELFS seismicic_ASCE_7_10_StructureType);
```

**C#**

```
public IRobotSELFS seismicic_ASCE_7_10_StructureType StructureTypeY { get; set; }
```

**Visual Basic**

```
Public StructureTypeY As IRobotSELFS seismicic_ASCE_7_10_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.8.2.8 TBaseValueX****C++**

```
HRESULT get_TBaseValueX(double* );
HRESULT put_TBaseValueX(double);
```

**C#**

```
public double TBaseValueX { get; set; }
```

**Visual Basic**

```
Public TBaseValueX As Double
```

**Version**

Available since version 12.5.

**II.1.33.8.2.9 TBaseValueY****C++**

```
HRESULT get_TBaseValueY(double* );
HRESULT put_TBaseValueY(double);
```

**C#**

```
public double TBaseValueY { get; set; }
```

**Visual Basic**

```
Public TBaseValueY As double
```

**Version**

Available since version 12.5.

**II.1.33.8.2.10 TL****C++**

```
HRESULT get_TL(double* );
HRESULT put_TL(double);
```

**C#**

```
public double TL { get; set; }
```

**Visual Basic**

```
Public TL As double
```

**Version**

Available since version 12.5.

**II.1.33.9 IRobotSELFS seismic\_ASCE\_7\_10\_StructureType****C++**

```
enum IRobotSELFS seismic_ASCE_7_10_StructureType;
```

**C#**

```
public enum IRobotSELFS seismic_ASCE_7_10_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSELFS seismic_ASCE_7_10_StructureType
```

**Members**

Members	Description
I_SSST_ASCE_7_10_STEEL_FRAMES = 1	Available since version 12.5.
I_SSST_ASCE_7_10_CONCRETE_FRAMES = 2	Available since version 12.5.
I_SSST_ASCE_7_10_BRACED_STEEL_FRAMES = 3	Available since version 12.5.
I_SSST_ASCE_7_10_OTHER_STRUCTURE_SYSTEM = 4	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.10 IRobotSELFS seismic\_EC\_8\_Params****Class Hierarchy****C++**

```
interface IRobotSELFS seismic_EC_8_Params : IDispatch;
```

**C#**

```
public interface IRobotSELFS seismic_EC_8_Params;
```

**Visual Basic**

```
Public Interface IRobotSELFS seismic_EC_8_Params
```

## Description

Parameters of EC 8 for the simplified seismic analysis.

## Version

Available since version 12.5.

### II.1.33.10.1 IRobotSELFSeismic\_EC\_8\_Params Members

The following tables list the members exposed by IRobotSELFSeismic\_EC\_8\_Params.

#### Public Fields

	Name	Description
◆	Ag ( <a href="#">see page 459</a> )	
◆	Beta ( <a href="#">see page 460</a> )	
◆	Q ( <a href="#">see page 460</a> )	
◆	SiteClass ( <a href="#">see page 460</a> )	Ground type; to set the envelope for several ground types, you should set the sum of values to be included in the envelope.
◆	SpectrumType ( <a href="#">see page 460</a> )	
◆	StructureTypeX ( <a href="#">see page 461</a> )	
◆	StructureTypeY ( <a href="#">see page 461</a> )	

#### Public Methods

	Name	Description
≡◆	SiteClassEnvelopeCheck ( <a href="#">see page 461</a> )	
≡◆	SiteClassEnvelopesChecked ( <a href="#">see page 462</a> )	

### II.1.33.10.2 IRobotSELFSeismic\_EC\_8\_Params Fields

The fields of the IRobotSELFSeismic\_EC\_8\_Params class are listed here.

#### Public Fields

	Name	Description
◆	Ag ( <a href="#">see page 459</a> )	
◆	Beta ( <a href="#">see page 460</a> )	
◆	Q ( <a href="#">see page 460</a> )	
◆	SiteClass ( <a href="#">see page 460</a> )	Ground type; to set the envelope for several ground types, you should set the sum of values to be included in the envelope.
◆	SpectrumType ( <a href="#">see page 460</a> )	
◆	StructureTypeX ( <a href="#">see page 461</a> )	
◆	StructureTypeY ( <a href="#">see page 461</a> )	

### II.1.33.10.2.1 Ag

#### C++

```
HRESULT get_Ag(double* );
HRESULT put_Ag(double);
```

#### C#

```
public double Ag { get; set; }
```

#### Visual Basic

```
Public Ag As Double
```

**Version**

Available since version 12.5.

**II.1.33.10.2.2 Beta****C++**

```
HRESULT get_Beta(double*);  
HRESULT put_Beta(double);
```

**C#**

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As double
```

**Version**

Available since version 12.5.

**II.1.33.10.2.3 Q****C++**

```
HRESULT get_Q(double*);  
HRESULT put_Q(double);
```

**C#**

```
public double Q { get; set; }
```

**Visual Basic**

```
Public Q As double
```

**Version**

Available since version 12.5.

**II.1.33.10.2.4 SiteClass****C++**

```
HRESULT get_SiteClass(IRobotSELFSeismic_EC_8_SiteClass*);  
HRESULT put_SiteClass(IRobotSELFSeismic_EC_8_SiteClass);
```

**C#**

```
public IRobotSELFSeismic_EC_8_SiteClass SiteClass { get; set; }
```

**Visual Basic**

```
Public SiteClass As IRobotSELFSeismic_EC_8_SiteClass
```

**Description**

Ground type; to set the envelope for several ground types, you should set the sum of values to be included in the envelope.

**Version**

Available since version 12.5.

**II.1.33.10.2.5 SpectrumType****C++**

```
HRESULT get_SpectrumType(IRobotSELFSeismic_EC_8_SpectrumType*);  
HRESULT put_SpectrumType(IRobotSELFSeismic_EC_8_SpectrumType);
```

**C#**

```
public IRobotSELFS seismicic_EC_8_SpectrumType SpectrumType { get; set; }
```

**Visual Basic**

```
Public SpectrumType As IRobotSELFS seismicic_EC_8_SpectrumType
```

**Version**

Available since version 12.5.

**II.1.33.10.2.6 StructureTypeX****C++**

```
HRESULT get_StructureTypeX(IRobotSELFS seismicic_EC_8_StructureType* );
HRESULT put_StructureTypeX(IRobotSELFS seismicic_EC_8_StructureType);
```

**C#**

```
public IRobotSELFS seismicic_EC_8_StructureType StructureTypeX { get; set; }
```

**Visual Basic**

```
Public StructureTypeX As IRobotSELFS seismicic_EC_8_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.10.2.7 StructureTypeY****C++**

```
HRESULT get_StructureTypeY(IRobotSELFS seismicic_EC_8_StructureType* );
HRESULT put_StructureTypeY(IRobotSELFS seismicic_EC_8_StructureType);
```

**C#**

```
public IRobotSELFS seismicic_EC_8_StructureType StructureTypeY { get; set; }
```

**Visual Basic**

```
Public StructureTypeY As IRobotSELFS seismicic_EC_8_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.10.3 IRobotSELFS seismicic\_EC\_8\_Params Methods**

The methods of the IRobotSELFS seismicic\_EC\_8\_Params class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
⊕	SiteClassEnvelopeCheck ( <a href="#">see page 461</a> )	
⊕	SiteClassEnvelopeIsChecked ( <a href="#">see page 462</a> )	

**II.1.33.10.3.1 SiteClassEnvelopeCheck****C++**

```
HRESULT SiteClassEnvelopeCheck(IRobotSELFS seismicic_EC_8_SiteClass _siteClass, VARIANT_BOOL _check);
```

**C#**

```
public void SiteClassEnvelopeCheck(IRobotSELFS seismic_EC_8_SiteClass _siteClass, bool _check);
```

**Visual Basic**

```
Public Sub SiteClassEnvelopeCheck(_siteClass As IRobotSELFS seismic_EC_8_SiteClass, _check As Boolean)
```

**Version**

Available since version 12.5.

**II.1.33.10.3.2 SiteClassEnvelopeIsChecked****C++**

```
HRESULT SiteClassEnvelopeIsChecked(IRobotSELFS seismic_EC_8_SiteClass _siteClass, VARIANT_BOOL* ret);
```

**C#**

```
public bool SiteClassEnvelopeIsChecked(IRobotSELFS seismic_EC_8_SiteClass _siteClass);
```

**Visual Basic**

```
Public Function SiteClassEnvelopeIsChecked(_siteClass As IRobotSELFS seismic_EC_8_SiteClass) As Boolean
```

**Version**

Available since version 12.5.

**II.1.33.11 IRobotSELFS seismic\_EC\_8\_StructureType****C++**

```
enum IRobotSELFS seismic_EC_8_StructureType;
```

**C#**

```
public enum IRobotSELFS seismic_EC_8_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSELFS seismic_EC_8_StructureType
```

**Members**

Members	Description
I_SSST_EC_8_STEEL_FRAMES = 1	Available since version 12.5.
I_SSST_EC_8_CONCRETE_FRAMES = 2	Available since version 12.5.
I_SSST_EC_8_BRACED_STEEL_FRAMES = 3	Available since version 12.5.
I_SSST_EC_8_OTHER_STRUCTURE_SYSTEM = 4	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.12 IRobotSELFS seismic\_EC\_8\_SiteClass****C++**

```
enum IRobotSELFS seismic_EC_8_SiteClass;
```

**C#**

```
public enum IRobotSELFS seismic_EC_8_SiteClass;
```

**Visual Basic**

```
Public Enum IRobotSELFS seismic_EC_8_SiteClass
```

**Members**

Members	Description
I_SSSC_EC_8_A = 1	Available since version 12.5.
I_SSSC_EC_8_B = 2	Available since version 12.5.
I_SSSC_EC_8_C = 4	Available since version 12.5.
I_SSSC_EC_8_D = 8	Available since version 12.5.
I_SSSC_EC_8_E = 16	Available since version 12.5.
I_SSSC_EC_8_ENVELOPE = 255	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.13 IRobotSELFS seismic\_EC\_8\_SpectrumType****C++**

```
enum IRobotSELFS seismic_EC_8_SpectrumType;
```

**C#**

```
public enum IRobotSELFS seismic_EC_8_SpectrumType;
```

**Visual Basic**

```
Public Enum IRobotSELFS seismic_EC_8_SpectrumType
```

**Members**

Members	Description
I_SSST_EC_8_Type1 = 1	Available since version 12.5.
I_SSST_EC_8_Type2 = 2	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.14 IRobotSELFS seismic\_AS\_1170\_4\_Params****Class Hierarchy****C++**

```
interface IRobotSELFS seismic_AS_1170_4_Params : IDispatch;
```

**C#**

```
public interface IRobotSELFS seismic_AS_1170_4_Params;
```

**Visual Basic**

```
Public Interface IRobotSELFS seismic_AS_1170_4_Params
```

**Description**

Parameters of the AS 1170-4 code for the simplified seismic analysis.

## Version

Available since version 12.5.

### II.1.33.14.1 IRobotSELFSeismic\_AS\_1170\_4\_Params Members

The following tables list the members exposed by IRobotSELFSeismic\_AS\_1170\_4\_Params.

#### Public Fields

	Name	Description
◆	Kp ( <a href="#">see page 464</a> )	
◆	Mi ( <a href="#">see page 464</a> )	
◆	Probability ( <a href="#">see page 465</a> )	
◆	SoilCategory ( <a href="#">see page 465</a> )	
◆	Sp ( <a href="#">see page 465</a> )	
◆	StructureTypeX ( <a href="#">see page 465</a> )	
◆	StructureTypeY ( <a href="#">see page 466</a> )	
◆	Z ( <a href="#">see page 466</a> )	

### II.1.33.14.2 IRobotSELFSeismic\_AS\_1170\_4\_Params Fields

The fields of the IRobotSELFSeismic\_AS\_1170\_4\_Params class are listed here.

#### Public Fields

	Name	Description
◆	Kp ( <a href="#">see page 464</a> )	
◆	Mi ( <a href="#">see page 464</a> )	
◆	Probability ( <a href="#">see page 465</a> )	
◆	SoilCategory ( <a href="#">see page 465</a> )	
◆	Sp ( <a href="#">see page 465</a> )	
◆	StructureTypeX ( <a href="#">see page 465</a> )	
◆	StructureTypeY ( <a href="#">see page 466</a> )	
◆	Z ( <a href="#">see page 466</a> )	

### II.1.33.14.2.1 Kp

#### C++

```
HRESULT get_Kp( double* );
HRESULT put_Kp( double );
```

#### C#

```
public double Kp { get; set; }
```

#### Visual Basic

```
Public Kp As Double
```

#### Version

Available since version 12.5.

### II.1.33.14.2.2 Mi

#### C++

```
HRESULT get_Mi( double* );
HRESULT put_Mi( double );
```

**C#**

```
public double Mi { get; set; }
```

**Visual Basic**

```
Public Mi As Double
```

**Version**

Available since version 12.5.

### II.1.33.14.2.3 Probability

**C++**

```
HRESULT get_Probability(IRobotSELFSeismic_AS_1170_4_ProbabilityType*);  
HRESULT put_Probability(IRobotSELFSeismic_AS_1170_4_ProbabilityType);
```

**C#**

```
public IRobotSELFSeismic_AS_1170_4_ProbabilityType Probability { get; set; }
```

**Visual Basic**

```
Public Probability As IRobotSELFSeismic_AS_1170_4_ProbabilityType
```

**Version**

Available since version 12.5.

### II.1.33.14.2.4 SoilCategory

**C++**

```
HRESULT get_SoilCategory(IRobotSELFSeismic_AS_1170_4_SoilCategoryType*);  
HRESULT put_SoilCategory(IRobotSELFSeismic_AS_1170_4_SoilCategoryType);
```

**C#**

```
public IRobotSELFSeismic_AS_1170_4_SoilCategoryType SoilCategory { get; set; }
```

**Visual Basic**

```
Public SoilCategory As IRobotSELFSeismic_AS_1170_4_SoilCategoryType
```

**Version**

Available since version 12.5.

### II.1.33.14.2.5 Sp

**C++**

```
HRESULT get_Sp(double*);  
HRESULT put_Sp(double);
```

**C#**

```
public double Sp { get; set; }
```

**Visual Basic**

```
Public Sp As Double
```

**Version**

Available since version 12.5.

### II.1.33.14.2.6 StructureTypeX

**C++**

```
HRESULT get_StructureTypeX(IRobotSELFSeismic_AS_1170_4_StructureType*);
```

```
HRESULT put_StructureTypeX(IRobotSELFS seismic_AS_1170_4_StructureType);
```

**C#**

```
public IRobotSELFS seismic_AS_1170_4_StructureType StructureTypeX { get; set; }
```

**Visual Basic**

```
Public StructureTypeX As IRobotSELFS seismic_AS_1170_4_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.14.2.7 StructureTypeY****C++**

```
HRESULT get_StructureTypeY(IRobotSELFS seismic_AS_1170_4_StructureType*);  
HRESULT put_StructureTypeY(IRobotSELFS seismic_AS_1170_4_StructureType);
```

**C#**

```
public IRobotSELFS seismic_AS_1170_4_StructureType StructureTypeY { get; set; }
```

**Visual Basic**

```
Public StructureTypeY As IRobotSELFS seismic_AS_1170_4_StructureType
```

**Version**

Available since version 12.5.

**II.1.33.14.2.8 Z****C++**

```
HRESULT get_Z(double*);  
HRESULT put_Z(double);
```

**C#**

```
public double Z { get; set; }
```

**Visual Basic**

```
Public Z As Double
```

**Version**

Available since version 12.5.

**II.1.33.15 IRobotSELFS seismic\_AS\_1170\_4\_SoilCategoryType****C++**

```
enum IRobotSELFS seismic_AS_1170_4_SoilCategoryType;
```

**C#**

```
public enum IRobotSELFS seismic_AS_1170_4_SoilCategoryType;
```

**Visual Basic**

```
Public Enum IRobotSELFS seismic_AS_1170_4_SoilCategoryType
```

**Members**

Members	Description
I_SSSC_AS_1170_4_A = 1	Available since version 12.5.
I_SSSC_AS_1170_4_B = 2	Available since version 12.5.

I_SSSC_AS_1170_4_C = 3	Available since version 12.5.
I_SSSC_AS_1170_4_D = 4	Available since version 12.5.
I_SSSC_AS_1170_4_E = 5	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.16 IRobotSELFSeismic\_AS\_1170\_4\_StructureType****C++**

```
enum IRobotSELFSeismic_AS_1170_4_StructureType;
```

**C#**

```
public enum IRobotSELFSeismic_AS_1170_4_StructureType;
```

**Visual Basic**

```
Public Enum IRobotSELFSeismic_AS_1170_4_StructureType
```

**Members**

Members	Description
I_SSST_AS_1170_4_STEEL_FRAMES = 1	Available since version 12.5.
I_SSST_AS_1170_4_CONCRETE_FRAMES = 2	Available since version 12.5.
I_SSST_AS_1170_4_BRACED_STEEL_FRAMES = 3	Available since version 12.5.
I_SSST_AS_1170_4_OTHER_STRUCTURES = 4	Available since version 12.5.

**Version**

Available since version 12.5.

**II.1.33.17 IRobotSELFSeismic\_AS\_1170\_4\_ProbabilityType****C++**

```
enum IRobotSELFSeismic_AS_1170_4_ProbabilityType;
```

**C#**

```
public enum IRobotSELFSeismic_AS_1170_4_ProbabilityType;
```

**Visual Basic**

```
Public Enum IRobotSELFSeismic_AS_1170_4_ProbabilityType
```

**Members**

Members	Description
I_SSPT_AS_1170_4_1_2500 = 1	Available since version 12.5.
I_SSPT_AS_1170_4_1_2000 = 2	Available since version 12.5.
I_SSPT_AS_1170_4_1_1500 = 3	Available since version 12.5.
I_SSPT_AS_1170_4_1_1000 = 4	Available since version 12.5.
I_SSPT_AS_1170_4_1_800 = 5	Available since version 12.5.
I_SSPT_AS_1170_4_1_500 = 6	Available since version 12.5.
I_SSPT_AS_1170_4_1_250 = 7	Available since version 12.5.
I_SSPT_AS_1170_4_1_200 = 8	Available since version 12.5.
I_SSPT_AS_1170_4_1_100 = 9	Available since version 12.5.
I_SSPT_AS_1170_4_1_50 = 10	Available since version 12.5.
I_SSPT_AS_1170_4_1_25 = 11	Available since version 12.5.

I_SSPT_AS_1170_4_1_20 = 12	Available since version 12.5.
----------------------------	-------------------------------

## Version

Available since version 12.5.

## II.1.34 Wind loads simulation

Available since version 14.5.

### Interfaces

	Name	Description
	IRobotWindLoadsSimulationEngine ( <a href="#">see page 468</a> )	
	IRobotWindLoadsSimulationParams ( <a href="#">see page 469</a> )	

### II.1.34.1 IRobotWindLoadsSimulationEngine

#### Class Hierarchy

#### C++

```
interface IRobotWindLoadsSimulationEngine : IDispatch;
```

#### C#

```
public interface IRobotWindLoadsSimulationEngine;
```

#### Visual Basic

```
Public Interface IRobotWindLoadsSimulationEngine
```

#### Version

Available since version 14.5.

### II.1.34.1.1 IRobotWindLoadsSimulationEngine Members

The following tables list the members exposed by IRobotWindLoadsSimulationEngine.

#### Public Fields

	Name	Description
	Params ( <a href="#">see page 468</a> )	

#### Public Methods

	Name	Description
	Generate ( <a href="#">see page 469</a> )	

### II.1.34.1.2 IRobotWindLoadsSimulationEngine Fields

The fields of the IRobotWindLoadsSimulationEngine class are listed here.

#### Public Fields

	Name	Description
	Params ( <a href="#">see page 468</a> )	

### II.1.34.1.2.1 Params

#### C++

```
HRESULT get_Params(IRobotWindLoadsSimulationParams**);
```

**C#**

```
public IRobotWindLoadsSimulationParams* Params { get; }
```

**Visual Basic**

```
Public ReadOnly Params As IRobotWindLoadsSimulationParams*
```

**Version**

Available since version 14.5.

**II.1.34.1.3 IRobotWindLoadsSimulationEngine Methods**

The methods of the IRobotWindLoadsSimulationEngine class are listed here.

**Public Methods**

	Name	Description
💡	Generate (see page 469)	

**II.1.34.1.3.1 Generate****C++**

```
HRESULT Generate(VARIANT_BOOL* ret);
```

**C#**

```
public bool Generate();
```

**Visual Basic**

```
Public Function Generate() As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2 IRobotWindLoadsSimulationParams****Class Hierarchy****C++**

```
interface IRobotWindLoadsSimulationParams : IDispatch;
```

**C#**

```
public interface IRobotWindLoadsSimulationParams;
```

**Visual Basic**

```
Public Interface IRobotWindLoadsSimulationParams
```

**Version**

Available since version 14.5.

**II.1.34.2.1 IRobotWindLoadsSimulationParams Members**

The following tables list the members exposed by IRobotWindLoadsSimulationParams.

**Public Fields**

	Name	Description
💡	DeviationPercent (see page 470)	
💡	DirectionXNEabled (see page 471)	

◆	DirectionXNYNEEnabled (see page 471)	
◆	DirectionXNYPEnabled (see page 471)	
◆	DirectionXPEnabled (see page 471)	
◆	DirectionXPYNEEnabled (see page 472)	
◆	DirectionXPYPEnabled (see page 472)	
◆	DirectionYNEEnabled (see page 472)	
◆	DirectionYPEEnabled (see page 472)	
◆	Elements (see page 473)	
◆	OpeningsClosed (see page 473)	
◆	TerrainLevel (see page 473)	
◆	Velocity (see page 474)	

### II.1.34.2.2 IRobotWindLoadsSimulationParams Fields

The fields of the IRobotWindLoadsSimulationParams class are listed here.

#### Public Fields

	Name	Description
◆	DeviationPercent (see page 470)	
◆	DirectionXNEEnabled (see page 471)	
◆	DirectionXNYNEEnabled (see page 471)	
◆	DirectionXNYPEnabled (see page 471)	
◆	DirectionXPEnabled (see page 471)	
◆	DirectionXPYNEEnabled (see page 472)	
◆	DirectionXPYPEnabled (see page 472)	
◆	DirectionYNEEnabled (see page 472)	
◆	DirectionYPEEnabled (see page 472)	
◆	Elements (see page 473)	
◆	OpeningsClosed (see page 473)	
◆	TerrainLevel (see page 473)	
◆	Velocity (see page 474)	

#### II.1.34.2.2.1 DeviationPercent

##### C++

```
HRESULT get_DeviationPercent(double* );
HRESULT put_DeviationPercent(double);
```

##### C#

```
public double DeviationPercent { get; set; }
```

**Visual Basic**

```
Public DeviationPercent As double
```

**Version**

Available since version 14.5.

**II.1.34.2.2.2 DirectionXNEnabled****C++**

```
HRESULT get_DirectionXNEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXNEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXNEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXNEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.3 DirectionXNYNEnabled****C++**

```
HRESULT get_DirectionXNYNEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXNYNEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXNYNEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXNYNEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.4 DirectionXNYPEEnabled****C++**

```
HRESULT get_DirectionXNYPEEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXNYPEEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXNYPEEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXNYPEEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.5 DirectionXPEnabled****C++**

```
HRESULT get_DirectionXPEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXPEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXPEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXPEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.6 DirectionXPYNEnabled****C++**

```
HRESULT get_DirectionXPYNEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXPYNEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXPYNEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXPYNEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.7 DirectionXPYPEnabled****C++**

```
HRESULT get_DirectionXPYPEnabled(VARIANT_BOOL* );
HRESULT put_DirectionXPYPEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionXPYPEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionXPYPEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.8 DirectionYNEnabled****C++**

```
HRESULT get_DirectionYNEnabled(VARIANT_BOOL* );
HRESULT put_DirectionYNEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionYNEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionYNEnabled As Boolean
```

**Version**

Available since version 14.5.

**II.1.34.2.2.9 DirectionYPEnabled****C++**

```
HRESULT get_DirectionYPEnabled(VARIANT_BOOL* );
```

```
HRESULT put_DirectionYPEnabled(VARIANT_BOOL);
```

**C#**

```
public bool DirectionYPEnabled { get; set; }
```

**Visual Basic**

```
Public DirectionYPEnabled As Boolean
```

**Version**

Available since version 14.5.

### II.1.34.2.2.10 Elements

**C++**

```
HRESULT get_Elements(IRobotSelection **);
```

**C#**

```
public IRobotSelection * Elements { get; }
```

**Visual Basic**

```
Public ReadOnly Elements As IRobotSelection *
```

**Version**

Available since version 14.5.

### II.1.34.2.2.11 OpeningsClosed

**C++**

```
HRESULT get_OpeningsClosed(VARIANT_BOOL*);  
HRESULT put_OpeningsClosed(VARIANT_BOOL);
```

**C#**

```
public bool OpeningsClosed { get; set; }
```

**Visual Basic**

```
Public OpeningsClosed As Boolean
```

**Version**

Available since version 14.5.

### II.1.34.2.2.12 TerrainLevel

**C++**

```
HRESULT get_TerrainLevel(double*);  
HRESULT put_TerrainLevel(double);
```

**C#**

```
public double TerrainLevel { get; set; }
```

**Visual Basic**

```
Public TerrainLevel As double
```

**Version**

Available since version 14.5.

### II.1.34.2.2.13 Velocity

#### C++

```
HRESULT get_Velocity( double* );
HRESULT put_Velocity( double );
```

#### C#

```
public double Velocity { get; set; }
```

#### Visual Basic

```
Public Velocity As Double
```

#### Version

Available since version 14.5.

### II.1.35 IRobotCasePredefinedNumber

#### C++

```
enum IRobotCasePredefinedNumber;
```

#### C#

```
public enum IRobotCasePredefinedNumber;
```

#### Visual Basic

```
Public Enum IRobotCasePredefinedNumber
```

#### Members

Members	Description
I_CPN_MASS_CONVERSION_GLOBAL = -2	Use this value to obtain the load case with a list of mass activation records including masses in all dynamic cases and in the static cases for body force loads and angular acceleration forces. . Available since version 15.2.
I_CPN_MASS_CONVERSION_DYNAMIC = -1	Use this value to obtain the load case with a list of mass activation records including masses in all dynamic cases. Available since version 15.2.

#### Description

A list of predefined values which can be used when calling method Get of RobotCaseServer in order to obtain a specific RobotCase object.

#### Version

Available since version 15.2.

## II.2 Nodes

A structure node is described by means of the RobotNode type. All the nodes are managed by the node server of the RobotNodeServer type. .

#### Interfaces

	Name	Description
	IRobotNode ( <a href="#">see page 522</a> )	Each structure node is represented in the Robot Object Model by means of the RobotNode interface. .
	IRobotNodeServer ( <a href="#">see page 527</a> )	Server of nodes manages all the structure nodes .

## II.2.1 Complex attributes of a node

Robot Object Model provides access to structure node attributes by means of a uniform mechanism of labels. .

### Enumerations

	<b>Name</b>	<b>Description</b>
	IRobotNodeSupportFixingDirection ( <a href="#">see page 510</a> )	Identifiers of directions that may be blocked out or released in a support. .
	IRobotNodeSupportOneDirectionFixingType ( <a href="#">see page 510</a> )	Identifiers of the manner of unilateral blocking of the indicated direction for supports.
	IRobotAdvancedSupportType ( <a href="#">see page 513</a> )	
	IRobotSupportEquivalentElasticityType ( <a href="#">see page 518</a> )	
	IRobotSupportColumnFixingType ( <a href="#">see page 522</a> )	

### Interfaces

	<b>Name</b>	<b>Description</b>
	IRobotNodeSupportData ( <a href="#">see page 499</a> )	Support parameters are described in the Robot Object Model by means of a set of values defined as RobotNodeSupportData. An object of this type contains data with the labels of the ILT_NODE_SUPPORT type. .
	IRobotEmitter ( <a href="#">see page 511</a> )	Emiter definition.
	IRobotAdvancedSupportData ( <a href="#">see page 513</a> )	
	IRobotSupportEquivalentElasticity ( <a href="#">see page 516</a> )	Definition of equivalent elasticity of a support.
	IRobotSupportEquivalentWallElasticity ( <a href="#">see page 518</a> )	
	IRobotSupportEquivalentColumnElasticity ( <a href="#">see page 520</a> )	

## II.2.1.1 Rigid links

Available since version 2.5.

### Interfaces

	<b>Name</b>	<b>Description</b>
	IRobotNodeRigidLinkData ( <a href="#">see page 475</a> )	Rigid link parameters for nodes.
	IRobotNodeRigidLinkDef ( <a href="#">see page 478</a> )	Definition of a node rigid link.
	IRobotNodeRigidLinkServer ( <a href="#">see page 480</a> )	Server allowing definition of rigid links for nodes. Moreover, the server provides access to the list of all the rigid links defined. .

## II.2.1.1.1 IRobotNodeRigidLinkData

### Class Hierarchy

#### C++

```
interface IRobotNodeRigidLinkData : IDispatch;
```

#### C#

```
public interface IRobotNodeRigidLinkData;
```

**Visual Basic**

```
Public Interface IRobotNodeRigidLinkData
```

**Description**

Rigid link parameters for nodes.

**Version**

Available since version 2.5.

**II.2.1.1.1.1 IRobotNodeRigidLinkData Members**

The following tables list the members exposed by IRobotNodeRigidLinkData.

**Public Fields**

	Name	Description
❖	RX (see page 476)	Blocked displacement in the RX direction.
❖	RY (see page 477)	Blocked displacement in the RY direction.
❖	RZ (see page 477)	Blocked displacement in the RZ direction.
❖	UX (see page 477)	Blocked displacement in the UX direction.
❖	UY (see page 478)	Blocked displacement in the UY direction.
❖	UZ (see page 478)	Blocked displacement in the UZ direction.

**II.2.1.1.1.2 IRobotNodeRigidLinkData Fields**

The fields of the IRobotNodeRigidLinkData class are listed here.

**Public Fields**

	Name	Description
❖	RX (see page 476)	Blocked displacement in the RX direction.
❖	RY (see page 477)	Blocked displacement in the RY direction.
❖	RZ (see page 477)	Blocked displacement in the RZ direction.
❖	UX (see page 477)	Blocked displacement in the UX direction.
❖	UY (see page 478)	Blocked displacement in the UY direction.
❖	UZ (see page 478)	Blocked displacement in the UZ direction.

**II.2.1.1.1.2.1 RX****C++**

```
HRESULT get_RX(VARIANT_BOOL* );
HRESULT put_RX(VARIANT_BOOL);
```

**C#**

```
public bool RX { get; set; }
```

**Visual Basic**

```
Public RX As Boolean
```

**Description**

Blocked displacement in the RX direction.

**Version**

Available since version 2.5.

### II.2.1.1.2.2 RY

#### C++

```
HRESULT get_RY(VARIANT_BOOL* );
HRESULT put_RY(VARIANT_BOOL);
```

#### C#

```
public bool RY { get; set; }
```

#### Visual Basic

```
Public RY As Boolean
```

#### Description

Blocked displacement in the RY direction.

#### Version

Available since version 2.5.

### II.2.1.1.2.3 RZ

#### C++

```
HRESULT get_RZ(VARIANT_BOOL* );
HRESULT put_RZ(VARIANT_BOOL);
```

#### C#

```
public bool RZ { get; set; }
```

#### Visual Basic

```
Public RZ As Boolean
```

#### Description

Blocked displacement in the RZ direction.

#### Version

Available since version 2.5.

### II.2.1.1.2.4 UX

#### C++

```
HRESULT get_UX(VARIANT_BOOL* );
HRESULT put_UX(VARIANT_BOOL);
```

#### C#

```
public bool UX { get; set; }
```

#### Visual Basic

```
Public UX As Boolean
```

#### Description

Blocked displacement in the UX direction.

#### Version

Available since version 2.5.

### II.2.1.1.2.5 UY

#### C++

```
HRESULT get_UY(VARIANT_BOOL* );
HRESULT put_UY(VARIANT_BOOL);
```

#### C#

```
public bool UY { get; set; }
```

#### Visual Basic

```
Public UY As Boolean
```

#### Description

Blocked displacement in the UY direction.

#### Version

Available since version 2.5.

### II.2.1.1.2.6 UZ

#### C++

```
HRESULT get_UZ(VARIANT_BOOL* );
HRESULT put_UZ(VARIANT_BOOL);
```

#### C#

```
public bool UZ { get; set; }
```

#### Visual Basic

```
Public UZ As Boolean
```

#### Description

Blocked displacement in the UZ direction.

#### Version

Available since version 2.5.

### II.2.1.1.2 IRobotNodeRigidLinkDef

#### Class Hierarchy

#### C++

```
interface IRobotNodeRigidLinkDef : IDispatch;
```

#### C#

```
public interface IRobotNodeRigidLinkDef;
```

#### Visual Basic

```
Public Interface IRobotNodeRigidLinkDef
```

#### Description

Definition of a node rigid link.

#### Version

Available since version 2.5.

### II.2.1.1.2.1 IRobotNodeRigidLinkDef Members

The following tables list the members exposed by IRobotNodeRigidLinkDef.

#### Public Fields

	Name	Description
◆	LabelName (see page 479)	Name of the label storing data for the link .
◆	Master (see page 479)	Number of master node.
◆	Slaves (see page 480)	Selection of slave nodes.

### II.2.1.1.2.2 IRobotNodeRigidLinkDef Fields

The fields of the IRobotNodeRigidLinkDef class are listed here.

#### Public Fields

	Name	Description
◆	LabelName (see page 479)	Name of the label storing data for the link .
◆	Master (see page 479)	Number of master node.
◆	Slaves (see page 480)	Selection of slave nodes.

### II.2.1.1.2.2.1 LabelName

#### C++

```
HRESULT get_LabelName(BSTR* );
HRESULT put_LabelName(BSTR);
```

#### C#

```
public String LabelName { get; set; }
```

#### Visual Basic

```
Public LabelName As String
```

#### Description

Name of the label storing data for the link .

#### Version

Available since version 2.5.

### II.2.1.1.2.2.2 Master

#### C++

```
HRESULT get_Master(long* );
HRESULT put_Master(long);
```

#### C#

```
public long Master { get; set; }
```

#### Visual Basic

```
Public Master As long
```

#### Description

Number of master node.

#### Version

Available since version 2.5.

### II.2.1.1.2.2.3 Slaves

#### C++

```
HRESULT get_Slaves(BSTR* );
HRESULT put_Slaves(BSTR);
```

#### C#

```
public String Slaves { get; set; }
```

#### Visual Basic

```
Public Slaves As String
```

#### Description

Selection of slave nodes.

#### Version

Available since version 2.5.

### II.2.1.1.3 IRobotNodeRigidLinkServer

#### Class Hierarchy

#### C++

```
interface IRobotNodeRigidLinkServer : IDispatch;
```

#### C#

```
public interface IRobotNodeRigidLinkServer;
```

#### Visual Basic

```
Public Interface IRobotNodeRigidLinkServer
```

#### Description

Server allowing definition of rigid links for nodes. Moreover, the server provides access to the list of all the rigid links defined.

#### Version

Available since version 2.5.

### II.2.1.1.3.1 IRobotNodeRigidLinkServer Members

The following tables list the members exposed by IRobotNodeRigidLinkServer.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 481</a> )	Number of defined rigid links.

#### Public Methods

	Name	Description
◆	Find ( <a href="#">see page 482</a> )	Function locates the rigid link in which the specified node occurs (either as a master node or slave node) and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned. .
◆	FindLabel ( <a href="#">see page 482</a> )	Function locates the rigid link described by the label of the specified name and returns first index of the object describing this link beginning with the determined start index. .

	FindMaster ( <a href="#">see page 482</a> )	Function locates a rigid link defined for the specified master node and returns index of the object describing this link. If no connection has been defined for the indicated node, then zero value is returned. .
	FindSlave ( <a href="#">see page 483</a> )	Function locates the rigid link, in which the node specified occurs as a slave node and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned.
	Get ( <a href="#">see page 483</a> )	Function returns object defining the selected rigid link. By activating this function with successive indexes from 1 to Count ( <a href="#">see page 481</a> ), the user may obtain information about all the rigid links defined.. .
	GetLabel ( <a href="#">see page 483</a> )	Function returns label storing parameters of the selected rigid link.
	Remove ( <a href="#">see page 484</a> )	Function deletes the rigid link for which definition parameters are described by the object with the indicated index on the list of rigid links defined. .
	RemoveSlave ( <a href="#">see page 484</a> )	Function deletes the specified slave node from the definition of the link which includes the node.
	Set ( <a href="#">see page 484</a> )	Function defines a rigid link for the indicated nodes. .

### II.2.1.1.3.2 IRobotNodeRigidLinkServer Fields

The fields of the IRobotNodeRigidLinkServer class are listed here.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 481</a> )	Number of defined rigid links.

#### II.2.1.1.3.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As Long
```

##### Description

Number of defined rigid links.

##### Version

Available since version 2.5.

### II.2.1.1.3.3 IRobotNodeRigidLinkServer Methods

The methods of the IRobotNodeRigidLinkServer class are listed here.

#### Public Methods

	Name	Description
	Find ( <a href="#">see page 482</a> )	Function locates the rigid link in which the specified node occurs (either as a master node or slave node) and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned. .
	FindLabel ( <a href="#">see page 482</a> )	Function locates the rigid link described by the label of the specified name and returns first index of the object describing this link beginning with the determined start index. .
	FindMaster ( <a href="#">see page 482</a> )	Function locates a rigid link defined for the specified master node and returns index of the object describing this link. If no connection has been defined for the indicated node, then zero value is returned. .

	FindSlave (see page 483)	Function locates the rigid link, in which the node specified occurs as a slave node and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned.
	Get (see page 483)	Function returns object defining the selected rigid link. By activating this function with successive indexes from 1 to Count (see page 481), the user may obtain information about all the rigid links defined. .
	GetLabel (see page 483)	Function returns label storing parameters of the selected rigid link.
	Remove (see page 484)	Function deletes the rigid link for which definition parameters are described by the object with the indicated index on the list of rigid links defined. .
	RemoveSlave (see page 484)	Function deletes the specified slave node from the definition of the link which includes the node.
	Set (see page 484)	Function defines a rigid link for the indicated nodes. .

### II.2.1.1.3.3.1 Find

#### C++

```
HRESULT Find(long _node_num, long* ret);
```

#### C#

```
public long Find(long _node_num);
```

#### Visual Basic

```
Public Function Find(_node_num As long) As long
```

#### Description

Function locates the rigid link in which the specified node occurs (either as a master node or slave node) and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned. .

#### Version

Available since version 2.5.

### II.2.1.1.3.3.2 FindLabel

#### C++

```
HRESULT FindLabel(BSTR _label_name, long _start_def_idx = 1, long* ret);
```

#### C#

```
public long FindLabel(String _label_name, long _start_def_idx = 1);
```

#### Visual Basic

```
Public Function FindLabel(_label_name As String, Optional _start_def_idx As long = 1) As long
```

#### Description

Function locates the rigid link described by the label of the specified name and returns first index of the object describing this link beginning with the determined start index. .

#### Version

Available since version 2.5.

### II.2.1.1.3.3.3 FindMaster

#### C++

```
HRESULT FindMaster(long _master, long* ret);
```

**C#**

```
public long FindMaster(long _master);
```

**Visual Basic**

```
Public Function FindMaster(_master As long) As long
```

**Description**

Function locates a rigid link defined for the specified master node and returns index of the object describing this link. If no connection has been defined for the indicated node, then zero value is returned. .

**Version**

Available since version 2.5.

**II.2.1.3.3.4 FindSlave****C++**

```
HRESULT FindSlave(long _slave_node, long* ret);
```

**C#**

```
public long FindSlave(long _slave_node);
```

**Visual Basic**

```
Public Function FindSlave(_slave_node As long) As long
```

**Description**

Function locates the rigid link, in which the node specified occurs as a slave node and returns index of the object describing this link. If no link has been defined for the determined node, then zero value is returned.

**Version**

Available since version 2.5.

**II.2.1.3.3.5 Get****C++**

```
HRESULT Get(long _def_idx, IRobotNodeRigidLinkDef** ret);
```

**C#**

```
public IRobotNodeRigidLinkDef Get(long _def_idx);
```

**Visual Basic**

```
Public Function Get(_def_idx As long) As IRobotNodeRigidLinkDef
```

**Description**

Function returns object defining the selected rigid link. By activating this function with successive indexes from 1 to Count (see page 481), the user may obtain information about all the rigid links defined. .

**Version**

Available since version 2.5.

**II.2.1.3.3.6 GetLabel****C++**

```
HRESULT GetLabel(long _def_idx, IRobotLabel** ret);
```

**C#**

```
public IRobotLabel GetLabel(long _def_idx);
```

**Visual Basic**

```
Public Function GetLabel(_def_idx As long) As IRobotLabel
```

**Description**

Function returns label storing parameters of the selected rigid link.

**Version**

Available since version 2.5.

**II.2.1.3.3.7 Remove****C++**

```
HRESULT Remove(long _def_idx);
```

**C#**

```
public void Remove(long _def_idx);
```

**Visual Basic**

```
Public Sub Remove(_def_idx As long)
```

**Description**

Function deletes the rigid link for which definition parameters are described by the object with the indicated index on the list of rigid links defined. .

**Version**

Available since version 2.5.

**II.2.1.3.3.8 RemoveSlave****C++**

```
HRESULT RemoveSlave(long _slave);
```

**C#**

```
public void RemoveSlave(long _slave);
```

**Visual Basic**

```
Public Sub RemoveSlave(_slave As long)
```

**Description**

Function deletes the specified slave node from the definition of the link which includes the node.

**Version**

Available since version 2.5.

**II.2.1.3.3.9 Set****C++**

```
HRESULT Set(long _master, BSTR _slaves, BSTR _label_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Set(long _master, String _slaves, String _label_name);
```

**Visual Basic**

```
Public Function Set(_master As long, _slaves As String, _label_name As String) As Boolean
```

**Description**

Function defines a rigid link for the indicated nodes..

**Version**

Available since version 2.5.

**II.2.1.2 Compatible nodes**

Available since version 2.5.

**Interfaces**

	<b>Name</b>	<b>Description</b>
↳	IRobotNodeCompatibilityData (see page 485)	Compatible node.
↳	IRobotNodeCompatibilityDef (see page 493)	Definition of compatible node.
↳	IRobotNodeCompatibilityServer (see page 495)	Server allowing definition of compatible nodes.

**II.2.1.2.1 IRobotNodeCompatibilityData****Class Hierarchy****C++**

```
interface IRobotNodeCompatibilityData : IDispatch;
```

**C#**

```
public interface IRobotNodeCompatibilityData;
```

**Visual Basic**

```
Public Interface IRobotNodeCompatibilityData
```

**Description**

Compatible node.

**Version**

Available since version 2.5.

**II.2.1.2.1.1 IRobotNodeCompatibilityData Members**

The following tables list the members exposed by IRobotNodeCompatibilityData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Alpha (see page 486)	Value of the compatibility angle (Z axis).
◆	AX (see page 487)	Damping coefficient (X axis).
◆	AY (see page 487)	Damping coefficient (Y axis).
◆	AZ (see page 487)	Damping coefficient (Z axis).
◆	Beta (see page 488)	Value of the compatibility angle (Y axis).
◆	BX (see page 488)	
◆	BY (see page 488)	
◆	BZ (see page 489)	

◆	Gamma (see page 489)	Value of the compatibility angle (X axis).
◆	HX (see page 489)	
◆	HY (see page 489)	
◆	HZ (see page 490)	
◆	KX (see page 490)	Elastic coefficient.
◆	KY (see page 490)	Elastic coefficient.
◆	KZ (see page 491)	Elastic coefficient.
◆	NonlinearModel (see page 491)	Non-linearity support.
◆	RX (see page 491)	Blocked direction (rotation about X axis).
◆	RY (see page 491)	Blocked direction (rotation about Y axis).
◆	RZ (see page 492)	Blocked direction (rotation about Z axis).
◆	UX (see page 492)	Blocked direction (X axis).
◆	UY (see page 492)	Blocked direction (Y axis).
◆	UZ (see page 493)	Blocked direction (Z axis).

### II.2.1.2.1.2 IRobotNodeCompatibilityData Fields

The fields of the IRobotNodeCompatibilityData class are listed here.

#### Public Fields

Name	Description
◆ Alpha (see page 486)	Value of the compatibility angle (Z axis).
◆ AX (see page 487)	Damping coefficient (X axis).
◆ AY (see page 487)	Damping coefficient (Y axis).
◆ AZ (see page 487)	Damping coefficient (Z axis).
◆ Beta (see page 488)	Value of the compatibility angle (Y axis).
◆ BX (see page 488)	
◆ BY (see page 488)	
◆ BZ (see page 489)	
◆ Gamma (see page 489)	Value of the compatibility angle (X axis).
◆ HX (see page 489)	
◆ HY (see page 489)	
◆ HZ (see page 490)	
◆ KX (see page 490)	Elastic coefficient.
◆ KY (see page 490)	Elastic coefficient.
◆ KZ (see page 491)	Elastic coefficient.
◆ NonlinearModel (see page 491)	Non-linearity support.
◆ RX (see page 491)	Blocked direction (rotation about X axis).
◆ RY (see page 491)	Blocked direction (rotation about Y axis).
◆ RZ (see page 492)	Blocked direction (rotation about Z axis).
◆ UX (see page 492)	Blocked direction (X axis).
◆ UY (see page 492)	Blocked direction (Y axis).
◆ UZ (see page 493)	Blocked direction (Z axis).

### II.2.1.2.1.2.1 Alpha

#### C++

```
HRESULT get_Alpha(double* );
HRESULT put_Alpha(double);
```

**C#**

```
public double Alpha { get; set; }
```

**Visual Basic**

```
Public Alpha As Double
```

**Description**

Value of the compatibility angle (Z axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.2 AX****C++**

```
HRESULT get_AX(double* );
HRESULT put_AX(double);
```

**C#**

```
public double AX { get; set; }
```

**Visual Basic**

```
Public AX As Double
```

**Description**

Damping coefficient (X axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.3 AY****C++**

```
HRESULT get_AY(double* );
HRESULT put_AY(double);
```

**C#**

```
public double AY { get; set; }
```

**Visual Basic**

```
Public AY As Double
```

**Description**

Damping coefficient (Y axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.4 AZ****C++**

```
HRESULT get_AZ(double* );
HRESULT put_AZ(double);
```

**C#**

```
public double AZ { get; set; }
```

**Visual Basic**

```
Public AZ As double
```

**Description**

Damping coefficient (Z axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.5 Beta****C++**

```
HRESULT get_Beta(double*);  
HRESULT put_Beta(double);
```

**C#**

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As double
```

**Description**

Value of the compatibility angle (Y axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.6 BX****C++**

```
HRESULT get_BX(double*);  
HRESULT put_BX(double);
```

**C#**

```
public double BX { get; set; }
```

**Visual Basic**

```
Public BX As double
```

**Version**

Available since version 2.5.

**II.2.1.2.1.2.7 BY****C++**

```
HRESULT get_BY(double*);  
HRESULT put_BY(double);
```

**C#**

```
public double BY { get; set; }
```

**Visual Basic**

```
Public BY As double
```

**Version**

Available since version 2.5.

### II.2.1.2.1.2.8 BZ

#### C++

```
HRESULT get_BZ(double*);  
HRESULT put_BZ(double);
```

#### C#

```
public double BZ { get; set; }
```

#### Visual Basic

```
Public BZ As Double
```

#### Version

Available since version 2.5.

### II.2.1.2.1.2.9 Gamma

#### C++

```
HRESULT get_Gamma(double*);  
HRESULT put_Gamma(double);
```

#### C#

```
public double Gamma { get; set; }
```

#### Visual Basic

```
Public Gamma As Double
```

#### Description

Value of the compatibility angle (X axis).

#### Version

Available since version 2.5.

### II.2.1.2.1.2.10 HX

#### C++

```
HRESULT get_HX(double*);  
HRESULT put_HX(double);
```

#### C#

```
public double HX { get; set; }
```

#### Visual Basic

```
Public HX As Double
```

#### Version

Available since version 2.5.

### II.2.1.2.1.2.11 HY

#### C++

```
HRESULT get_HY(double*);  
HRESULT put_HY(double);
```

#### C#

```
public double HY { get; set; }
```

**Visual Basic**

```
Public HY As double
```

**Version**

Available since version 2.5.

**II.2.1.2.1.2.12 HZ****C++**

```
HRESULT get_HZ(double* );
HRESULT put_HZ(double);
```

**C#**

```
public double HZ { get; set; }
```

**Visual Basic**

```
Public HZ As double
```

**Version**

Available since version 2.5.

**II.2.1.2.1.2.13 KX****C++**

```
HRESULT get_KX(double* );
HRESULT put_KX(double);
```

**C#**

```
public double KX { get; set; }
```

**Visual Basic**

```
Public KX As double
```

**Description**

Elastic coefficient.

**Version**

Available since version 2.5.

**II.2.1.2.1.2.14 KY****C++**

```
HRESULT get_KY(double* );
HRESULT put_KY(double);
```

**C#**

```
public double KY { get; set; }
```

**Visual Basic**

```
Public KY As double
```

**Description**

Elastic coefficient.

**Version**

Available since version 2.5.

### II.2.1.2.1.2.15 KZ

#### C++

```
HRESULT get_KZ(double* );
HRESULT put_KZ(double);
```

#### C#

```
public double KZ { get; set; }
```

#### Visual Basic

```
Public KZ As Double
```

#### Description

Elastic coefficient.

#### Version

Available since version 2.5.

### II.2.1.2.1.2.16 NonlinearModel

#### C++

```
HRESULT get_NonlinearModel(IRobotNonlinearLinkMngr** );
```

#### C#

```
public IRobotNonlinearLinkMngr NonlinearModel { get; }
```

#### Visual Basic

```
Public ReadOnly NonlinearModel As IRobotNonlinearLinkMngr
```

#### Description

Non-linearity support.

#### Version

Available since version 3.

### II.2.1.2.1.2.17 RX

#### C++

```
HRESULT get_RX(VARIANT_BOOL* );
HRESULT put_RX(VARIANT_BOOL);
```

#### C#

```
public bool RX { get; set; }
```

#### Visual Basic

```
Public RX As Boolean
```

#### Description

Blocked direction (rotation about X axis).

#### Version

Available since version 2.5.

### II.2.1.2.1.2.18 RY

#### C++

```
HRESULT get_RY(VARIANT_BOOL* );
```

```
HRESULT put_RY(VARIANT_BOOL);
```

**C#**

```
public bool RY { get; set; }
```

**Visual Basic**

```
Public RY As Boolean
```

**Description**

Blocked direction (rotation about Y axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.19 RZ****C++**

```
HRESULT get_RZ(VARIANT_BOOL*);  
HRESULT put_RZ(VARIANT_BOOL);
```

**C#**

```
public bool RZ { get; set; }
```

**Visual Basic**

```
Public RZ As Boolean
```

**Description**

Blocked direction (rotation about Z axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.20 UX****C++**

```
HRESULT get_UX(VARIANT_BOOL*);  
HRESULT put_UX(VARIANT_BOOL);
```

**C#**

```
public bool UX { get; set; }
```

**Visual Basic**

```
Public UX As Boolean
```

**Description**

Blocked direction (X axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.21 UY****C++**

```
HRESULT get_UY(VARIANT_BOOL*);  
HRESULT put_UY(VARIANT_BOOL);
```

**C#**

```
public bool UY { get; set; }
```

**Visual Basic**

```
Public UY As Boolean
```

**Description**

Blocked direction (Y axis).

**Version**

Available since version 2.5.

**II.2.1.2.1.2.22 UZ****C++**

```
HRESULT get_UZ(VARIANT_BOOL* );
HRESULT put_UZ(VARIANT_BOOL);
```

**C#**

```
public bool UZ { get; set; }
```

**Visual Basic**

```
Public UZ As Boolean
```

**Description**

Blocked direction (Z axis).

**Version**

Available since version 2.5.

**II.2.1.2.2 IRobotNodeCompatibilityDef****Class Hierarchy****C++**

```
interface IRobotNodeCompatibilityDef : IDispatch;
```

**C#**

```
public interface IRobotNodeCompatibilityDef;
```

**Visual Basic**

```
Public Interface IRobotNodeCompatibilityDef
```

**Description**

Definition of compatible node.

**Version**

Available since version 2.5.

**II.2.1.2.2.1 IRobotNodeCompatibilityDef Members**

The following tables list the members exposed by IRobotNodeCompatibilityDef.

**Public Fields**

	Name	Description
❖	Bars (❑ see page 494)	List of bars for which the compatibility model is defined .
❖	Compatible (❑ see page 494)	Number of compatible node.
❖	LabelName (❑ see page 494)	Name of the label storing data for a compatible node.
❖	Main (❑ see page 495)	Number of master node.

### II.2.1.2.2.2 IRobotNodeCompatibilityDef Fields

The fields of the IRobotNodeCompatibilityDef class are listed here.

#### Public Fields

	Name	Description
◆	Bars ( <a href="#">see page 494</a> )	List of bars for which the compatibility model is defined .
◆	Compatible ( <a href="#">see page 494</a> )	Number of compatible node.
◆	LabelName ( <a href="#">see page 494</a> )	Name of the label storing data for a compatible node.
◆	Main ( <a href="#">see page 495</a> )	Number of master node.

#### II.2.1.2.2.2.1 Bars

##### C++

```
HRESULT get_Bars(String* );
HRESULT put_Bars(String);
```

##### C#

```
public String Bars { get; set; }
```

##### Visual Basic

```
Public Bars As String
```

##### Description

List of bars for which the compatibility model is defined .

##### Version

Available since version 2.5.

#### II.2.1.2.2.2.2 Compatible

##### C++

```
HRESULT get_Compatible(long* );
HRESULT put_Compatible(long);
```

##### C#

```
public long Compatible { get; set; }
```

##### Visual Basic

```
Public Compatible As long
```

##### Description

Number of compatible node.

##### Version

Available since version 2.5.

#### II.2.1.2.2.2.3 LabelName

##### C++

```
HRESULT get_LabelName(BSTR* );
HRESULT put_LabelName(BSTR);
```

##### C#

```
public String LabelName { get; set; }
```

**Visual Basic**

```
Public LabelName As String
```

**Description**

Name of the label storing data for a compatible node.

**Version**

Available since version 2.5.

**II.2.1.2.2.2.4 Main****C++**

```
HRESULT get_Main(long* );
HRESULT put_Main(long);
```

**C#**

```
public long Main { get; set; }
```

**Visual Basic**

```
Public Main As long
```

**Description**

Number of master node.

**Version**

Available since version 2.5.

**II.2.1.2.3 IRobotNodeCompatibilityServer****Class Hierarchy****C++**

```
interface IRobotNodeCompatibilityServer : IDispatch;
```

**C#**

```
public interface IRobotNodeCompatibilityServer;
```

**Visual Basic**

```
Public Interface IRobotNodeCompatibilityServer
```

**Description**

Server allowing definition of compatible nodes.

**Version**

Available since version 2.5.

**II.2.1.2.3.1 IRobotNodeCompatibilityServer Members**

The following tables list the members exposed by IRobotNodeCompatibilityServer.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 496</a> )	Number of compatible nodes defined.

## Public Methods

	Name	Description
💡	Find (see page 497)	Function returns index of the first located compatible node definition in which the specified node occurs either as a master node or compatible node.
💡	FindCompatible (see page 497)	Function returns definition index of the compatible node of the indicated user number. .
💡	FindLabel (see page 497)	Function returns index of the first located compatible node definition with the specified name.
💡	FindMain (see page 498)	Function returns index of the first located definition of a compatible node in the specified master node beginning with the start index. .
💡	Get (see page 498)	Function returns object containing compatible node parameters. By activating this function with indexes from 1 to Count (see page 496), the user may obtain information about all the compatible nodes defined. .
💡	GetLabel (see page 498)	Function returns label describing parameters of the determined compatible node. .
💡	Remove (see page 499)	Function deletes the compatible node whose definition method describes the object of the specified index on the list of all compatible nodes. .
💡	Set (see page 499)	Function defines a compatible node for the selected bars at the indicated node. The selected bars must converge at the given node. .

### II.2.1.2.3.2 IRobotNodeCompatibilityServer Fields

The fields of the IRobotNodeCompatibilityServer class are listed here.

#### Public Fields

	Name	Description
💡	Count (see page 496)	Number of compatible nodes defined.

#### II.2.1.2.3.2.1 Count

##### C++

```
HRESULT get_Count(long* );
HRESULT put_Count(long);
```

##### C#

```
public long Count { get; set; }
```

##### Visual Basic

```
Public Count As long
```

##### Description

Number of compatible nodes defined.

##### Version

Available since version 2.5.

### II.2.1.2.3.3 IRobotNodeCompatibilityServer Methods

The methods of the IRobotNodeCompatibilityServer class are listed here.

#### Public Methods

	Name	Description
💡	Find (see page 497)	Function returns index of the first located compatible node definition in which the specified node occurs either as a master node or compatible node.

	FindCompatible (see page 497)	Function returns definition index of the compatible node of the indicated user number. .
	FindLabel (see page 497)	Function returns index of the first located compatible node definition with the specified name.
	FindMain (see page 498)	Function returns index of the first located definition of a compatible node in the specified master node beginning with the start index. .
	Get (see page 498)	Function returns object containing compatible node parameters. By activating this function with indexes from 1 to Count (see page 496), the user may obtain information about all the compatible nodes defined. .
	GetLabel (see page 498)	Function returns label describing parameters of the determined compatible node. .
	Remove (see page 499)	Function deletes the compatible node whose definition method describes the object of the specified index on the list of all compatible nodes. .
	Set (see page 499)	Function defines a compatible node for the selected bars at the indicated node. The selected bars must converge at the given node. .

### II.2.1.2.3.3.1 Find

**C++**

```
HRESULT Find(long _node, long _start_def_idx = 1, long* ret);
```

**C#**

```
public long Find(long _node, long _start_def_idx = 1);
```

**Visual Basic**

```
Public Function Find(_node As long, Optional _start_def_idx As long = 1) As long
```

**Description**

Function returns index of the first located compatible node definition in which the specified node occurs either as a master node or compatible node.

**Version**

Available since version 2.5.

### II.2.1.2.3.3.2 FindCompatible

**C++**

```
HRESULT FindCompatible(long _compatible_node, long* ret);
```

**C#**

```
public long FindCompatible(long _compatible_node);
```

**Visual Basic**

```
Public Function FindCompatible(_compatible_node As long) As long
```

**Description**

Function returns definition index of the compatible node of the indicated user number. .

**Version**

Available since version 2.5.

### II.2.1.2.3.3.3 FindLabel

**C++**

```
HRESULT FindLabel(BSTR _label_name, long _start_def_idx = 1, long* ret);
```

**C#**

```
public long FindLabel(string _label_name, long _start_def_idx = 1);
```

**Visual Basic**

```
Public Function FindLabel(_label_name As String, Optional _start_def_idx As long = 1) As long
```

**Description**

Function returns index of the first located compatible node definition with the specified name.

**Version**

Available since version 2.5.

**II.2.1.2.3.3.4 FindMain****C++**

```
HRESULT FindMain(long _main_node, long _start_def_idx = 1, long* ret);
```

**C#**

```
public long FindMain(long _main_node, long _start_def_idx = 1);
```

**Visual Basic**

```
Public Function FindMain(_main_node As long, Optional _start_def_idx As long = 1) As long
```

**Description**

Function returns index of the first located definition of a compatible node in the specified master node beginning with the start index. .

**Version**

Available since version 2.5.

**II.2.1.2.3.3.5 Get****C++**

```
HRESULT Get(long _def_idx, IRobotNodeCompatibilityDef** ret);
```

**C#**

```
public IRobotNodeCompatibilityDef Get(long _def_idx);
```

**Visual Basic**

```
Public Function Get(_def_idx As long) As IRobotNodeCompatibilityDef
```

**Description**

Function returns object containing compatible node parameters. By activating this function with indexes from 1 to Count (↗ see page 496), the user may obtain information about all the compatible nodes defined. .

**Version**

Available since version 2.5.

**II.2.1.2.3.3.6 GetLabel****C++**

```
HRESULT GetLabel(long _def_idx, IRobotLabel** ret);
```

**C#**

```
public IRobotLabel GetLabel(long _def_idx);
```

**Visual Basic**

```
Public Function GetLabel(_def_idx As long) As IRobotLabel
```

**Description**

Function returns label describing parameters of the determined compatible node. .

**Version**

Available since version 2.5.

**II.2.1.2.3.3.7 Remove****C++**

```
HRESULT Remove(long _def_idx);
```

**C#**

```
public void Remove(long _def_idx);
```

**Visual Basic**

```
Public Sub Remove(_def_idx As long)
```

**Description**

Function deletes the compatible node whose definition method describes the object of the specified index on the list of all compatible nodes. .

**Version**

Available since version 2.5.

**II.2.1.2.3.3.8 Set****C++**

```
HRESULT Set(long _node, BSTR _bar_list, BSTR _label_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Set(long _node, String _bar_list, String _label_name);
```

**Visual Basic**

```
Public Function Set(_node As long, _bar_list As String, _label_name As String) As Boolean
```

**Description**

Function defines a compatible node for the selected bars at the indicated node. The selected bars must converge at the given node. .

**Version**

Available since version 2.5.

**II.2.1.3 IRobotNodeSupportData****Class Hierarchy****C++**

```
interface IRobotNodeSupportData : IDispatch;
```

**C#**

```
public interface IRobotNodeSupportData;
```

**Visual Basic**

```
Public Interface IRobotNodeSupportData
```

**Description**

Support parameters are described in the Robot Object Model by means of a set of values defined as RobotNodeSupportData. An object of this type contains data with the labels of the ILT\_NODE\_SUPPORT type.

**II.2.1.3.1 IRobotNodeSupportData Members**

The following tables list the members exposed by IRobotNodeSupportData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Advanced (see page 502)	
◆	Alpha (see page 502)	Angle value in radians.
◆	AX (see page 502)	
◆	AY (see page 502)	
◆	AZ (see page 503)	
◆	Beta (see page 503)	Angle value in radians.
◆	BX (see page 503)	
◆	BY (see page 503)	
◆	BZ (see page 504)	
◆	ElasticLinear (see page 504)	Linear elasticity distribution; If the flag assumes value different from zero (True), then values of support elastic coefficients will be given per one running meter of length. Value of the elastic coefficient in a node is determined in the following manner: <ul style="list-style-type: none"> <li>• for each node, the distance between centers of segments between successive nodes is determined</li> <li>• the determined distance is multiplied by the coefficient per one running meter defined by the user</li> <li>• the determined value is defined in the node</li> </ul>
◆	ElasticSurface (see page 504)	Surface elasticity distribution; If the flag assumes value different from zero (True), then value of support elasticity is given per surface unit and it is assigned to nodes in proportion to the area of elements adjoining each of the support nodes .
◆	Gamma (see page 505)	Angle value in radians.
◆	GlobalCoordSystem (see page 505)	Definition of a rigid support in the global / local system.
◆	HX (see page 505)	
◆	HY (see page 505)	
◆	HZ (see page 506)	
◆	KX (see page 506)	
◆	KY (see page 506)	
◆	KZ (see page 506)	
◆	NonlinearModel (see page 506)	Non-linearity support.
◆	RX (see page 507)	
◆	RY (see page 507)	
◆	RZ (see page 507)	

◆	UX ( <a href="#">see page 507</a> )	
◆	UY ( <a href="#">see page 507</a> )	
◆	UZ ( <a href="#">see page 508</a> )	

**Public Methods**

	Name	Description
◆	GetAdvanced ( <a href="#">see page 508</a> )	Function returns support parameters applied during RC structure analysis. .
◆	GetOneDir ( <a href="#">see page 509</a> )	The function returns the manner of unilateral blocking of the indicated direction for support.
◆	IsFixed ( <a href="#">see page 509</a> )	The function checks if the given direction is blocked.
◆	SetAdvanced ( <a href="#">see page 509</a> )	Function defines support parameters applied during RC structure analysis. .
◆	SetFixed ( <a href="#">see page 509</a> )	The function allows one to block / un-block an indicated direction.
◆	SetOneDir ( <a href="#">see page 510</a> )	The function sets the manner of unilateral blocking of the indicated direction for the support.

**II.2.1.3.2 IRobotNodeSupportData Fields**

The fields of the IRobotNodeSupportData class are listed here.

**Public Fields**

	Name	Description
◆	Advanced ( <a href="#">see page 502</a> )	
◆	Alpha ( <a href="#">see page 502</a> )	Angle value in radians.
◆	AX ( <a href="#">see page 502</a> )	
◆	AY ( <a href="#">see page 502</a> )	
◆	AZ ( <a href="#">see page 503</a> )	
◆	Beta ( <a href="#">see page 503</a> )	Angle value in radians.
◆	BX ( <a href="#">see page 503</a> )	
◆	BY ( <a href="#">see page 503</a> )	
◆	BZ ( <a href="#">see page 504</a> )	
◆	ElasticLinear ( <a href="#">see page 504</a> )	Linear elasticity distribution; If the flag assumes value different from zero (True), then values of support elastic coefficients will be given per one running meter of length. Value of the elastic coefficient in a node is determined in the following manner: <ul style="list-style-type: none"> <li>• for each node, the distance between centers of segments between successive nodes is determined</li> <li>• the determined distance is multiplied by the coefficient per one running meter defined by the user</li> <li>• the determined value is defined in the node</li> </ul>
◆	ElasticSurface ( <a href="#">see page 504</a> )	Surface elasticity distribution; If the flag assumes value different from zero (True), then value of support elasticity is given per surface unit and it is assigned to nodes in proportion to the area of elements adjoining each of the support nodes .
◆	Gamma ( <a href="#">see page 505</a> )	Angle value in radians.
◆	GlobalCoordSystem ( <a href="#">see page 505</a> )	Definition of a rigid support in the global / local system.
◆	HX ( <a href="#">see page 505</a> )	
◆	HY ( <a href="#">see page 505</a> )	
◆	HZ ( <a href="#">see page 506</a> )	
◆	KX ( <a href="#">see page 506</a> )	

◆ KY (see page 506)		
◆ KZ (see page 506)		
◆ NonlinearModel (see page 506)	Non-linearity support.	
◆ RX (see page 507)		
◆ RY (see page 507)		
◆ RZ (see page 507)		
◆ UX (see page 507)		
◆ UY (see page 507)		
◆ UZ (see page 508)		

### II.2.1.3.2.1 Advanced

#### C++

```
HRESULT get_Advanced(IRobotAdvancedSupportData**);
```

#### C#

```
public IRobotAdvancedSupportData Advanced { get; }
```

#### Visual Basic

```
Public ReadOnly Advanced As IRobotAdvancedSupportData
```

#### Version

Available since version 8.5.

### II.2.1.3.2.2 Alpha

#### C++

```
HRESULT get_Alpha(double*);  
HRESULT put_Alpha(double);
```

#### C#

```
public double Alpha { get; set; }
```

#### Visual Basic

```
Public Alpha As Double
```

#### Description

Angle value in radians.

### II.2.1.3.2.3 AX

#### C++

```
HRESULT get_AX(double*);  
HRESULT put_AX(double);
```

#### C#

```
public double AX { get; set; }
```

#### Visual Basic

```
Public AX As Double
```

### II.2.1.3.2.4 AY

#### C++

```
HRESULT get_AY(double*);  
HRESULT put_AY(double);
```

**C#**

```
public double AY { get; set; }
```

**Visual Basic**

```
Public AY As Double
```

**II.2.1.3.2.5 AZ****C++**

```
HRESULT get_AZ(double*);  
HRESULT put_AZ(double);
```

**C#**

```
public double AZ { get; set; }
```

**Visual Basic**

```
Public AZ As Double
```

**II.2.1.3.2.6 Beta****C++**

```
HRESULT get_Beta(double*);  
HRESULT put_Beta(double);
```

**C#**

```
public double Beta { get; set; }
```

**Visual Basic**

```
Public Beta As Double
```

**Description**

Angle value in radians.

**II.2.1.3.2.7 BX****C++**

```
HRESULT get_BX(double*);  
HRESULT put_BX(double);
```

**C#**

```
public double BX { get; set; }
```

**Visual Basic**

```
Public BX As Double
```

**II.2.1.3.2.8 BY****C++**

```
HRESULT get_BY(double*);  
HRESULT put_BY(double);
```

**C#**

```
public double BY { get; set; }
```

**Visual Basic**

```
Public BY As Double
```

### II.2.1.3.2.9 BZ

#### C++

```
HRESULT get_BZ(double* );
HRESULT put_BZ(double );
```

#### C#

```
public double BZ { get; set; }
```

#### Visual Basic

```
Public BZ As Double
```

### II.2.1.3.2.10 ElasticLinear

#### C++

```
HRESULT get_ElasticLinear(VARIANT_BOOL* );
HRESULT put_ElasticLinear(VARIANT_BOOL );
```

#### C#

```
public bool ElasticLinear { get; set; }
```

#### Visual Basic

```
Public ElasticLinear As Boolean
```

#### Description

Linear elasticity distribution; If the flag assumes value different from zero (True), then values of support elastic coefficients will be given per one running meter of length. Value of the elastic coefficient in a node is determined in the following manner:

- for each node, the distance between centers of segments between successive nodes is determined
- the determined distance is multiplied by the coefficient per one running meter defined by the user
- the determined value is defined in the node

#### Version

Available since version 2.5.

### II.2.1.3.2.11 ElasticSurface

#### C++

```
HRESULT get_ElasticSurface(VARIANT_BOOL* );
HRESULT put_ElasticSurface(VARIANT_BOOL );
```

#### C#

```
public bool ElasticSurface { get; set; }
```

#### Visual Basic

```
Public ElasticSurface As Boolean
```

#### Description

Surface elasticity distribution; If the flag assumes value different from zero (True), then value of support elasticity is given per surface unit and it is assigned to nodes in proportion to the area of elements adjoining each of the support nodes .

#### Version

Available since version 2.5.

### II.2.1.3.2.12 Gamma

#### C++

```
HRESULT get_Gamma(double*);  
HRESULT put_Gamma(double);
```

#### C#

```
public double Gamma { get; set; }
```

#### Visual Basic

```
Public Gamma As Double
```

#### Description

Angle value in radians.

### II.2.1.3.2.13 GlobalCoordSystem

#### C++

```
HRESULT get_GlobalCoordSystem(VARIANT_BOOL*);  
HRESULT put_GlobalCoordSystem(VARIANT_BOOL);
```

#### C#

```
public bool GlobalCoordSystem { get; set; }
```

#### Visual Basic

```
Public GlobalCoordSystem As Boolean
```

#### Description

Definition of a rigid support in the global / local system.

#### Version

Available since version 2.5.

### II.2.1.3.2.14 HX

#### C++

```
HRESULT get_HX(double*);  
HRESULT put_HX(double);
```

#### C#

```
public double HX { get; set; }
```

#### Visual Basic

```
Public HX As Double
```

### II.2.1.3.2.15 HY

#### C++

```
HRESULT get_HY(double*);  
HRESULT put_HY(double);
```

#### C#

```
public double HY { get; set; }
```

#### Visual Basic

```
Public HY As Double
```

### II.2.1.3.2.16 HZ

**C++**

```
HRESULT get_HZ(double*);  
HRESULT put_HZ(double);
```

**C#**

```
public double HZ { get; set; }
```

**Visual Basic**

```
Public HZ As Double
```

### II.2.1.3.2.17 KX

**C++**

```
HRESULT get_KX(double*);  
HRESULT put_KX(double);
```

**C#**

```
public double KX { get; set; }
```

**Visual Basic**

```
Public KX As Double
```

### II.2.1.3.2.18 KY

**C++**

```
HRESULT get_KY(double*);  
HRESULT put_KY(double);
```

**C#**

```
public double KY { get; set; }
```

**Visual Basic**

```
Public KY As Double
```

### II.2.1.3.2.19 KZ

**C++**

```
HRESULT get_KZ(double*);  
HRESULT put_KZ(double);
```

**C#**

```
public double KZ { get; set; }
```

**Visual Basic**

```
Public KZ As Double
```

### II.2.1.3.2.20 NonlinearModel

**C++**

```
HRESULT get_NonlinearModel(IRobotNonlinearLinkMngr**);
```

**C#**

```
public IRobotNonlinearLinkMngr NonlinearModel { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearModel As IRobotNonlinearLinkMngr
```

**Description**

Non-linearity support.

**Version**

Available since version 3.

**II.2.1.3.2.21 RX****C++**

```
HRESULT get_RX(VARIANT_BOOL* );
HRESULT put_RX(VARIANT_BOOL);
```

**C#**

```
public bool RX { get; set; }
```

**Visual Basic**

```
Public RX As Boolean
```

**II.2.1.3.2.22 RY****C++**

```
HRESULT get_RY(VARIANT_BOOL* );
HRESULT put_RY(VARIANT_BOOL);
```

**C#**

```
public bool RY { get; set; }
```

**Visual Basic**

```
Public RY As Boolean
```

**II.2.1.3.2.23 RZ****C++**

```
HRESULT get_RZ(VARIANT_BOOL* );
HRESULT put_RZ(VARIANT_BOOL);
```

**C#**

```
public bool RZ { get; set; }
```

**Visual Basic**

```
Public RZ As Boolean
```

**II.2.1.3.2.24 UX****C++**

```
HRESULT get_UX(VARIANT_BOOL* );
HRESULT put_UX(VARIANT_BOOL);
```

**C#**

```
public bool UX { get; set; }
```

**Visual Basic**

```
Public UX As Boolean
```

**II.2.1.3.2.25 UY****C++**

```
HRESULT get_UY(VARIANT_BOOL* );
HRESULT put_UY(VARIANT_BOOL);
```

**C#**

```
public bool UY { get; set; }
```

**Visual Basic**

```
Public UY As Boolean
```

**II.2.1.3.2.26 UZ****C++**

```
HRESULT get_UZ(VARIANT_BOOL* );
HRESULT put_UZ(VARIANT_BOOL);
```

**C#**

```
public bool UZ { get; set; }
```

**Visual Basic**

```
Public UZ As Boolean
```

**II.2.1.3.3 IRobotNodeSupportData Methods**

The methods of the IRobotNodeSupportData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetAdvanced ( [ see page 508) )	Function returns support parameters applied during RC structure analysis. .
💡	GetOneDir ( [ see page 509)	The function returns the manner of unilateral blocking of the indicated direction for support.
💡	IsFixed ( [ see page 509)	The function checks if the given direction is blocked.
💡	SetAdvanced ( [ see page 509)	Function defines support parameters applied during RC structure analysis. .
💡	SetFixed ( [ see page 509)	The function allows one to block / un-block an indicated direction.
💡	SetOneDir ( [ see page 510)	The function sets the manner of unilateral blocking of the indicated direction for the support.

**II.2.1.3.3.1 GetAdvanced****C++**

```
HRESULT GetAdvanced(double* _b, double* _h, IRobotAdvancedSupportType* ret);
```

**C#**

```
public IRobotAdvancedSupportType GetAdvanced(double* _b, double* _h);
```

**Visual Basic**

```
Public Function GetAdvanced(ByRef _b As double*, ByRef _h As double*) As
IRobotAdvancedSupportType
```

**Description**

Function returns support parameters applied during RC structure analysis. .

**Version**

Available since version 2.5.

### II.2.1.3.3.2 GetOneDir

**C++**

```
HRESULT GetOneDir(IRobotNodeSupportFixingDirection _direction,
IRobotNodeSupportOneDirectionFixingType* ret);
```

**C#**

```
public IRobotNodeSupportOneDirectionFixingType GetOneDir(IRobotNodeSupportFixingDirection
_direction);
```

**Visual Basic**

```
Public Function GetOneDir(_direction As IRobotNodeSupportFixingDirection) As
IRobotNodeSupportOneDirectionFixingType
```

**Description**

The function returns the manner of unilateral blocking of the indicated direction for support.

### II.2.1.3.3.3 IsFixed

**C++**

```
HRESULT IsFixed(IRobotNodeSupportFixingDirection _direction, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsFixed(IRobotNodeSupportFixingDirection _direction);
```

**Visual Basic**

```
Public Function IsFixed(_direction As IRobotNodeSupportFixingDirection) As Boolean
```

**Description**

The function checks if the given direction is blocked.

### II.2.1.3.3.4 SetAdvanced

**C++**

```
HRESULT SetAdvanced(IRobotAdvancedSupportType _type, double _b, double _h);
```

**C#**

```
public void SetAdvanced(IRobotAdvancedSupportType _type, double _b, double _h);
```

**Visual Basic**

```
Public Sub SetAdvanced(_type As IRobotAdvancedSupportType, _b As Double, _h As Double)
```

**Description**

Function defines support parameters applied during RC structure analysis. .

**Version**

Available since version 2.5.

### II.2.1.3.3.5 SetFixed

**C++**

```
HRESULT SetFixed(IRobotNodeSupportFixingDirection _direction, VARIANT_BOOL _fixed);
```

**C#**

```
public void SetFixed(IRobotNodeSupportFixingDirection _direction, bool _fixed);
```

**Visual Basic**

```
Public Sub SetFixed(_direction As IRobotNodeSupportFixingDirection, _fixed As Boolean)
```

**Description**

The function allows one to block / un-block an indicated direction.

**II.2.1.3.3.6 SetOneDir****C++**

```
HRESULT SetOneDir(IRobotNodeSupportFixingDirection _direction,
IRobotNodeSupportOneDirectionFixingType _how_blocked);
```

**C#**

```
public void SetOneDir(IRobotNodeSupportFixingDirection _direction,
IRobotNodeSupportOneDirectionFixingType _how_blocked);
```

**Visual Basic**

```
Public Sub SetOneDir(_direction As IRobotNodeSupportFixingDirection, _how_blocked As
IRobotNodeSupportOneDirectionFixingType)
```

**Description**

The function sets the manner of unilateral blocking of the indicated direction for the support.

**II.2.1.4 IRobotNodeSupportFixingDirection****C++**

```
enum IRobotNodeSupportFixingDirection;
```

**C#**

```
public enum IRobotNodeSupportFixingDirection;
```

**Visual Basic**

```
Public Enum IRobotNodeSupportFixingDirection
```

**Description**

Identifiers of directions that may be blocked out or released in a support. .

**II.2.1.5 IRobotNodeSupportOneDirectionFixingType****C++**

```
enum IRobotNodeSupportOneDirectionFixingType;
```

**C#**

```
public enum IRobotNodeSupportOneDirectionFixingType;
```

**Visual Basic**

```
Public Enum IRobotNodeSupportOneDirectionFixingType
```

**Description**

Identifiers of the manner of unilateral blocking of the indicated direction for supports.

## II.2.1.6 IRobotEmitter

### Class Hierarchy

#### C++

```
interface IRobotEmitter : IDispatch;
```

#### C#

```
public interface IRobotEmitter;
```

### Visual Basic

```
Public Interface IRobotEmitter
```

### Description

Emitter definition.

## II.2.1.6.1 IRobotEmitter Members

The following tables list the members exposed by IRobotEmitter.

### Public Fields

	Name	Description
◆	EstimatedElemNumber ( <a href="#">see page 511</a> )	Estimated element number in the sphere (diameter R1 ( <a href="#">see page 512</a> )) Available since version 1.7.
◆	H0 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	R1 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	R2 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	VariableMeshDensityIncrement ( <a href="#">see page 512</a> )	Available since version 1.7.

## II.2.1.6.2 IRobotEmitter Fields

The fields of the IRobotEmitter class are listed here.

### Public Fields

	Name	Description
◆	EstimatedElemNumber ( <a href="#">see page 511</a> )	Estimated element number in the sphere (diameter R1 ( <a href="#">see page 512</a> )) Available since version 1.7.
◆	H0 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	R1 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	R2 ( <a href="#">see page 512</a> )	Available since version 1.7.
◆	VariableMeshDensityIncrement ( <a href="#">see page 512</a> )	Available since version 1.7.

## II.2.1.6.2.1 EstimatedElemNumber

#### C++

```
HRESULT get_EstimatedElemNumber(long*);
```

#### C#

```
public long EstimatedElemNumber { get; }
```

### Visual Basic

```
Public ReadOnly EstimatedElemNumber As long
```

**Description**

Estimated element number in the sphere (diameter R1 (see page 512)) Available since version 1.7.

**II.2.1.6.2.2 H0****C++**

```
HRESULT get_H0(double*);  
HRESULT put_H0(double);
```

**C#**

```
public double H0 { get; set; }
```

**Visual Basic**

```
Public H0 As double
```

**Description**

Available since version 1.7.

**II.2.1.6.2.3 R1****C++**

```
HRESULT get_R1(double*);  
HRESULT put_R1(double);
```

**C#**

```
public double R1 { get; set; }
```

**Visual Basic**

```
Public R1 As double
```

**Description**

Available since version 1.7.

**II.2.1.6.2.4 R2****C++**

```
HRESULT get_R2(double*);  
HRESULT put_R2(double);
```

**C#**

```
public double R2 { get; set; }
```

**Visual Basic**

```
Public R2 As double
```

**Description**

Available since version 1.7.

**II.2.1.6.2.5 VariableMeshDensityIncrement****C++**

```
HRESULT get_VariableMeshDensityIncrement(VARIANT_BOOL*);  
HRESULT put_VariableMeshDensityIncrement(VARIANT_BOOL);
```

**C#**

```
public bool VariableMeshDensityIncrement { get; set; }
```

**Visual Basic**

```
Public VariableMeshDensityIncrement As Boolean
```

**Description**

Available since version 1.7.

**II.2.1.7 IRobotAdvancedSupportType****C++**

```
enum IRobotAdvancedSupportType;
```

**C#**

```
public enum IRobotAdvancedSupportType;
```

**Visual Basic**

```
Public Enum IRobotAdvancedSupportType
```

**Members**

Members	Description
I_AST_NODAL = 1	Support defined in a node. Available since version 2.5.
I_AST_COLUMN_RECTANGULAR = 2	Rectangular column - it is required to define width b and height h of a column cross-section . Available since version 2.5.
I_AST_COLUMN_CIRCULAR = 3	Round column - it is required to define diameter b of a column cross-section . Available since version 2.5.
I_AST_WALL_MASONRY = 4	Masonry support defined by specifying the wall width b. Available since version 2.5.
I_AST_WALL_CONCRETE = 5	Concrete support defined by specifying the wall width b . Available since version 2.5.

**Version**

Available since version 2.5.

**II.2.1.8 IRobotAdvancedSupportData****Class Hierarchy****C++**

```
interface IRobotAdvancedSupportData : IDispatch;
```

**C#**

```
public interface IRobotAdvancedSupportData;
```

**Visual Basic**

```
Public Interface IRobotAdvancedSupportData
```

**Version**

Available since version 8.5.

**II.2.1.8.1 IRobotAdvancedSupportData Members**

The following tables list the members exposed by IRobotAdvancedSupportData.

## Public Fields

	Name	Description
◆	B ( <a href="#">see page 514</a> )	
◆	D ( <a href="#">see page 514</a> )	
◆	EquivalentElasticity ( <a href="#">see page 515</a> )	
◆	H ( <a href="#">see page 515</a> )	
◆	IsEquivalentElasticity ( <a href="#">see page 515</a> )	
◆	Type ( <a href="#">see page 515</a> )	Support type.

### II.2.1.8.2 IRobotAdvancedSupportData Fields

The fields of the IRobotAdvancedSupportData class are listed here.

## Public Fields

	Name	Description
◆	B ( <a href="#">see page 514</a> )	
◆	D ( <a href="#">see page 514</a> )	
◆	EquivalentElasticity ( <a href="#">see page 515</a> )	
◆	H ( <a href="#">see page 515</a> )	
◆	IsEquivalentElasticity ( <a href="#">see page 515</a> )	
◆	Type ( <a href="#">see page 515</a> )	Support type.

### II.2.1.8.2.1 B

#### C++

```
HRESULT get_B(double* );
HRESULT put_B(double);
```

#### C#

```
public double B { get; set; }
```

#### Visual Basic

```
Public B As Double
```

#### Version

Available since version 8.5.

### II.2.1.8.2.2 D

#### C++

```
HRESULT get_D(double* );
HRESULT put_D(double);
```

#### C#

```
public double D { get; set; }
```

#### Visual Basic

```
Public D As Double
```

#### Version

Available since version 8.5.

### II.2.1.8.2.3 EquivalentElasticity

**C++**

```
HRESULT get_EquivalentElasticity(IRobotSupportEquivalentElasticity**);
```

**C#**

```
public IRobotSupportEquivalentElasticity EquivalentElasticity { get; }
```

**Visual Basic**

```
Public ReadOnly EquivalentElasticity As IRobotSupportEquivalentElasticity
```

**Version**

Available since version 8.5.

### II.2.1.8.2.4 H

**C++**

```
HRESULT get_H(double*);  
HRESULT put_H(double);
```

**C#**

```
public double H { get; set; }
```

**Visual Basic**

```
Public H As Double
```

**Version**

Available since version 8.5.

### II.2.1.8.2.5 IsEquivalentElasticity

**C++**

```
HRESULT get_IsEquivalentElasticity(VARIANT_BOOL*);  
HRESULT put_IsEquivalentElasticity(VARIANT_BOOL);
```

**C#**

```
public bool IsEquivalentElasticity { get; set; }
```

**Visual Basic**

```
Public IsEquivalentElasticity As Boolean
```

**Version**

Available since version 8.5.

### II.2.1.8.2.6 Type

**C++**

```
HRESULT get_Type(IRobotAdvancedSupportType*);  
HRESULT put_Type(IRobotAdvancedSupportType);
```

**C#**

```
public IRobotAdvancedSupportType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotAdvancedSupportType
```

**Description**

Support type.

**Version**

Available since version 8.5.

**II.2.1.9 IRobotSupportEquivalentElasticity****Class Hierarchy****C++**

```
interface IRobotSupportEquivalentElasticity : IDispatch;
```

**C#**

```
public interface IRobotSupportEquivalentElasticity;
```

**Visual Basic**

```
Public Interface IRobotSupportEquivalentElasticity
```

**Description**

Definition of equivalent elasticity of a support.

**Version**

Available since version 8.5.

**II.2.1.9.1 IRobotSupportEquivalentElasticity Members**

The following tables list the members exposed by IRobotSupportEquivalentElasticity.

**Public Fields**

	Name	Description
◆	E (see page 516)	Young's modulus value.
◆	PoissonRatio (see page 517)	Poisson's ratio value.
◆	Type (see page 517)	

**Public Methods**

	Name	Description
◆	SetMaterial (see page 517)	

**II.2.1.9.2 IRobotSupportEquivalentElasticity Fields**

The fields of the IRobotSupportEquivalentElasticity class are listed here.

**Public Fields**

	Name	Description
◆	E (see page 516)	Young's modulus value.
◆	PoissonRatio (see page 517)	Poisson's ratio value.
◆	Type (see page 517)	

**II.2.1.9.2.1 E****C++**

```
HRESULT get_E(double* );
HRESULT put_E(double);
```

**C#**

```
public double E { get; set; }
```

**Visual Basic**

```
Public E As Double
```

**Description**

Young's modulus value.

**Version**

Available since version 8.5.

**II.2.1.9.2.2 PoissonRatio****C++**

```
HRESULT get_PoissonRatio(double* );
HRESULT put_PoissonRatio(double);
```

**C#**

```
public double PoissonRatio { get; set; }
```

**Visual Basic**

```
Public PoissonRatio As Double
```

**Description**

Poisson's ratio value.

**Version**

Available since version 8.5.

**II.2.1.9.2.3 Type****C++**

```
HRESULT get_Type(IRobotSupportEquivalentElasticityType* );
```

**C#**

```
public IRobotSupportEquivalentElasticityType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotSupportEquivalentElasticityType
```

**Version**

Available since version 8.5.

**II.2.1.9.3 IRobotSupportEquivalentElasticity Methods**

The methods of the IRobotSupportEquivalentElasticity class are listed here.

**Public Methods**

	Name	Description
	SetMaterial (see page 517)	

**II.2.1.9.3.1 SetMaterial****C++**

```
HRESULT SetMaterial(BSTR _material_name);
```

**C#**

```
public void SetMaterial(String _material_name);
```

**Visual Basic**

```
Public Sub SetMaterial(_material_name As String)
```

**Version**

Available since version 8.5.

**II.2.1.10 IRobotSupportEquivalentElasticityType****C++**

```
enum IRobotSupportEquivalentElasticityType;
```

**C#**

```
public enum IRobotSupportEquivalentElasticityType;
```

**Visual Basic**

```
Public Enum IRobotSupportEquivalentElasticityType
```

**Members**

Members	Description
I_SEET_COLUMN = 1	Available since version 8.5.
I_SEET_WALL = 2	Available since version 8.5.

**Version**

Available since version 8.5.

**II.2.1.11 IRobotSupportEquivalentWallElasticity****Class Hierarchy****C++**

```
interface IRobotSupportEquivalentWallElasticity : IRobotSupportEquivalentElasticity;
```

**C#**

```
public interface IRobotSupportEquivalentWallElasticity : IRobotSupportEquivalentElasticity;
```

**Visual Basic**

```
Public Interface IRobotSupportEquivalentWallElasticity
```

**Version**

Available since version 8.5.

**II.2.1.11.1 IRobotSupportEquivalentWallElasticity Members**

The following tables list the members exposed by IRobotSupportEquivalentWallElasticity.

**Public Fields**

	Name	Description
◆	E (see page 516)	Young's modulus value.
◆	L1 (see page 519)	
◆	L2 (see page 519)	
◆	PoissonRatio (see page 517)	Poisson's ratio value.

◆	ThroughTwoStories (see page 519)	
◆	Type (see page 517)	

**Public Methods**

	Name	Description
◆	SetMaterial (see page 517)	

**II.2.1.11.2 IRobotSupportEquivalentWallElasticity Fields**

The fields of the IRobotSupportEquivalentWallElasticity class are listed here.

**Public Fields**

	Name	Description
◆	L1 (see page 519)	
◆	L2 (see page 519)	
◆	ThroughTwoStories (see page 519)	

**II.2.1.11.2.1 L1****C++**

```
HRESULT get_L1(double* );
HRESULT put_L1(double);
```

**C#**

```
public double L1 { get; set; }
```

**Visual Basic**

```
Public L1 As double
```

**Version**

Available since version 8.5.

**II.2.1.11.2.2 L2****C++**

```
HRESULT get_L2(double* );
HRESULT put_L2(double);
```

**C#**

```
public double L2 { get; set; }
```

**Visual Basic**

```
Public L2 As double
```

**Version**

Available since version 8.5.

**II.2.1.11.2.3 ThroughTwoStories****C++**

```
HRESULT get_ThroughTwoStories(VARIANT_BOOL* );
HRESULT put_ThroughTwoStories(VARIANT_BOOL);
```

**C#**

```
public bool ThroughTwoStories { get; set; }
```

## Visual Basic

```
Public ThroughTwoStories As Boolean
```

### Version

Available since version 8.5.

## II.2.1.12 IRobotSupportEquivalentColumnElasticity

### Class Hierarchy

#### C++

```
interface IRobotSupportEquivalentColumnElasticity : IRobotSupportEquivalentElasticity;
```

#### C#

```
public interface IRobotSupportEquivalentColumnElasticity :  
IRobotSupportEquivalentElasticity;
```

## Visual Basic

```
Public Interface IRobotSupportEquivalentColumnElasticity
```

### Version

Available since version 8.5.

## II.2.1.12.1 IRobotSupportEquivalentColumnElasticity Members

The following tables list the members exposed by IRobotSupportEquivalentColumnElasticity.

### Public Fields

	Name	Description
❖	E (☞ see page 516)	Young's modulus value.
❖	Fixing1 (☞ see page 521)	
❖	Fixing2 (☞ see page 521)	
❖	L1 (☞ see page 521)	
❖	L2 (☞ see page 521)	
❖	PoissonRatio (☞ see page 517)	Poisson's ratio value.
❖	ThroughTwoStories (☞ see page 522)	
❖	Type (☞ see page 517)	

### Public Methods

	Name	Description
❖	SetMaterial (☞ see page 517)	

## II.2.1.12.2 IRobotSupportEquivalentColumnElasticity Fields

The fields of the IRobotSupportEquivalentColumnElasticity class are listed here.

### Public Fields

	Name	Description
❖	Fixing1 (☞ see page 521)	
❖	Fixing2 (☞ see page 521)	
❖	L1 (☞ see page 521)	
❖	L2 (☞ see page 521)	
❖	ThroughTwoStories (☞ see page 522)	

### II.2.1.12.2.1 Fixing1

**C++**

```
HRESULT get_Fixing1(IRobotSupportColumnFixingType* );
HRESULT put_Fixing1(IRobotSupportColumnFixingType);
```

**C#**

```
public IRobotSupportColumnFixingType Fixing1 { get; set; }
```

**Visual Basic**

```
Public Fixing1 As IRobotSupportColumnFixingType
```

**Version**

Available since version 8.5.

### II.2.1.12.2.2 Fixing2

**C++**

```
HRESULT get_Fixing2(IRobotSupportColumnFixingType* );
HRESULT put_Fixing2(IRobotSupportColumnFixingType);
```

**C#**

```
public IRobotSupportColumnFixingType Fixing2 { get; set; }
```

**Visual Basic**

```
Public Fixing2 As IRobotSupportColumnFixingType
```

**Version**

Available since version 8.5.

### II.2.1.12.2.3 L1

**C++**

```
HRESULT get_L1(double* );
HRESULT put_L1(double);
```

**C#**

```
public double L1 { get; set; }
```

**Visual Basic**

```
Public L1 As double
```

**Version**

Available since version 8.5.

### II.2.1.12.2.4 L2

**C++**

```
HRESULT get_L2(double* );
HRESULT put_L2(double);
```

**C#**

```
public double L2 { get; set; }
```

**Visual Basic**

```
Public L2 As double
```

**Version**

Available since version 8.5.

**II.2.1.12.2.5 ThroughTwoStories****C++**

```
HRESULT get_ThroughTwoStories(VARIANT_BOOL* );
HRESULT put_ThroughTwoStories(VARIANT_BOOL);
```

**C#**

```
public bool ThroughTwoStories { get; set; }
```

**Visual Basic**

```
Public ThroughTwoStories As Boolean
```

**Version**

Available since version 8.5.

**II.2.1.13 IRobotSupportColumnFixingType****C++**

```
enum IRobotSupportColumnFixingType;
```

**C#**

```
public enum IRobotSupportColumnFixingType;
```

**Visual Basic**

```
Public Enum IRobotSupportColumnFixingType
```

**Members**

Members	Description
I_SCFT_FIXED = 1	Available since version 8.5.
I_SCFT_PINNED = 2	Available since version 8.5.
I_SCFT_ROLLER = 3	Available since version 8.5.

**Version**

Available since version 8.5.

**II.2.2 IRobotNode****Class Hierarchy****C++**

```
interface IRobotNode : IRobotDataObject;
```

**C#**

```
public interface IRobotNode : IRobotDataObject;
```

**Visual Basic**

```
Public Interface IRobotNode
```

**Description**

Each structure node is represented in the Robot Object Model by means of the RobotNode interface. .

### II.2.2.1 IRobotNode Members

The following tables list the members exposed by IRobotNode.

#### Public Fields

	Name	Description
◆	HasCalcSupport (↗ see page 524)	Flag indicating that a support was created automatically at the node during generation of a structure model.
◆	HasEmitter (↗ see page 524)	Flag indicating if emitter is assigned to the node Available since version 1.7.
◆	IsCalc (↗ see page 524)	Flag indicating if the node has been created automatically due to generation of the calculation model.
◆	Number (↗ see page 30)	Object number is assigned by the user; it is unique among all the components of the same type..
◆	UniqeId (↗ see page 524)	Unique identifier of structure component Available since version 1.7.
◆	X (↗ see page 525)	Coordinate x.
◆	Y (↗ see page 525)	Coordinate y.
◆	Z (↗ see page 525)	Coordinate z.

#### Public Methods

	Name	Description
◆	GetCalcSupport (↗ see page 526)	Function makes available data defining a support created automatically at the node during generation of a structure model.
◆	GetEmitter (↗ see page 526)	Function returns the emitter defined in the node. If the emitter has not been defined the function returns empty reference (NULL). Available since version 1.7.
◆	GetLabel (↗ see page 31)	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabelName (↗ see page 31)	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabels (↗ see page 31)	The function returns a collection containing all labels (of different types) defined for a given object. .
◆	HasLabel (↗ see page 32)	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
◆	RemoveEmitter (↗ see page 526)	Function deletes emitter from the node Available since version 1.7.
◆	RemoveLabel (↗ see page 32)	The function deletes a label of the indicated type. .
◆	SetEmitter (↗ see page 527)	Function defines emitter in the node Available since version 1.7.
◆	SetLabel (↗ see page 32)	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .

### II.2.2.2 IRobotNode Fields

The fields of the IRobotNode class are listed here.

#### Public Fields

	Name	Description
◆	HasCalcSupport (↗ see page 524)	Flag indicating that a support was created automatically at the node during generation of a structure model.
◆	HasEmitter (↗ see page 524)	Flag indicating if emitter is assigned to the node Available since version 1.7.
◆	IsCalc (↗ see page 524)	Flag indicating if the node has been created automatically due to generation of the calculation model.

❖	Uniqueid ( [ see page 524 ] )	Unique identifier of structure component Available since version 1.7.
❖	X ( [ see page 525 ] )	Coordinate x.
❖	Y ( [ see page 525 ] )	Coordinate y.
❖	Z ( [ see page 525 ] )	Coordinate z.

### II.2.2.2.1 HasCalcSupport

**C++**

```
HRESULT get_HasCalcSupport(VARIANT_BOOL* );
```

**C#**

```
public bool HasCalcSupport { get; }
```

**Visual Basic**

```
Public ReadOnly HasCalcSupport As Boolean
```

**Description**

Flag indicating that a support was created automatically at the node during generation of a structure model.

**Version**

Available since version 8.

### II.2.2.2.2 HasEmitter

**C++**

```
HRESULT get_HasEmitter(VARIANT_BOOL* );
```

**C#**

```
public bool HasEmitter { get; }
```

**Visual Basic**

```
Public ReadOnly HasEmitter As Boolean
```

**Description**

Flag indicating if emitter is assigned to the node Available since version 1.7.

### II.2.2.2.3 IsCalc

**C++**

```
HRESULT get_IsCalc(VARIANT_BOOL* );
```

**C#**

```
public bool IsCalc { get; }
```

**Visual Basic**

```
Public ReadOnly IsCalc As Boolean
```

**Description**

Flag indicating if the node has been created automatically due to generation of the calculation model.

**Version**

Available since version 2.5.

### II.2.2.2.4 UniqueId

**C++**

```
HRESULT get_UniqueId(long* );
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As Long
```

**Description**

Unique identifier of structure component Available since version 1.7.

**II.2.2.5 X****C++**

```
HRESULT get_X(double*);  
HRESULT put_X(double);
```

**C#**

```
public double X { get; set; }
```

**Visual Basic**

```
Public X As Double
```

**Description**

Coordinate x.

**II.2.2.6 Y****C++**

```
HRESULT get_Y(double*);  
HRESULT put_Y(double);
```

**C#**

```
public double Y { get; set; }
```

**Visual Basic**

```
Public Y As Double
```

**Description**

Coordinate y.

**II.2.2.7 Z****C++**

```
HRESULT get_Z(double*);  
HRESULT put_Z(double);
```

**C#**

```
public double Z { get; set; }
```

**Visual Basic**

```
Public Z As Double
```

**Description**

Coordinate z.

**II.2.2.3 IRobotNode Methods**

The methods of the IRobotNode class are listed here.

## Public Methods

	Name	Description
💡	GetCalcSupport (see page 526)	Function makes available data defining a support created automatically at the node during generation of a structure model.
💡	GetEmitter (see page 526)	Function returns the emitter defined in the node. If the emitter has not been defined the function returns empty reference (NULL). Available since version 1.7.
💡	RemoveEmitter (see page 526)	Function deletes emitter from the node Available since version 1.7.
💡	SetEmitter (see page 527)	Function defines emitter in the node Available since version 1.7.

### II.2.2.3.1 GetCalcSupport

#### C++

```
HRESULT GetCalcSupport(IRobotNodeSupportData** ret);
```

#### C#

```
public IRobotNodeSupportData GetCalcSupport();
```

#### Visual Basic

```
Public Function GetCalcSupport() As IRobotNodeSupportData
```

#### Description

Function makes available data defining a support created automatically at the node during generation of a structure model.

#### Version

Available since version 8.

### II.2.2.3.2 GetEmitter

#### C++

```
HRESULT GetEmitter(IRobotEmitter** ret);
```

#### C#

```
public IRobotEmitter GetEmitter();
```

#### Visual Basic

```
Public Function GetEmitter() As IRobotEmitter
```

#### Description

Function returns the emitter defined in the node. If the emitter has not been defined the function returns empty reference (NULL). Available since version 1.7.

### II.2.2.3.3 RemoveEmitter

#### C++

```
HRESULT RemoveEmitter();
```

#### C#

```
public void RemoveEmitter();
```

#### Visual Basic

```
Public Sub RemoveEmitter()
```

## Description

Function deletes emitter from the node Available since version 1.7.

### II.2.2.3.4 SetEmitter

#### C++

```
HRESULT SetEmitter(IRobotEmitter* _emit);
```

#### C#

```
public void SetEmitter(IRobotEmitter _emit);
```

#### Visual Basic

```
Public Sub SetEmitter(ByRef _emit As IRobotEmitter)
```

#### Description

Function defines emitter in the node Available since version 1.7.

## II.2.3 IRobotNodeServer

#### Class Hierarchy

#### C++

```
interface IRobotNodeServer : IRobotDataObjectServer;
```

#### C#

```
public interface IRobotNodeServer : IRobotDataObjectServer;
```

#### Visual Basic

```
Public Interface IRobotNodeServer
```

#### Description

Server of nodes manages all the structure nodes .

### II.2.3.1 IRobotNodeServer Members

The following tables list the members exposed by IRobotNodeServer.

#### Public Fields

	Name	Description
❖	CompatibleNodes (see page 528)	Compatible node server.
❖	FreeNumber (see page 529)	First available user number which can be assigned to the structure node Available since version 1.7.
❖	NonlinearLinks (see page 529)	Non-linear connection models.
❖	RigidLinks (see page 529)	Rigid link server.

#### Public Methods

	Name	Description
❖	Create (see page 530)	The function creates a new node with the provided parameters. .
❖	Delete (see page 34)	The function deletes the object of the indicated number. .
❖	DeleteMany (see page 34)	The function deletes all objects that meet the criteria of the indicated selection..
❖	Exist (see page 34)	The function returns True (non-zero value) if the object of the indicated name already exists. .

	FindWithId ( <a href="#">see page 530</a> )	Function returns a number of the node with a specified unique identifier. If such a node is not found, then, zero value is returned. Available since version 1.7.
	Get ( <a href="#">see page 34</a> )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .
	GetAll ( <a href="#">see page 35</a> )	The function returns a collection containing all objects managed by the server. .
	GetCalcNodes ( <a href="#">see page 531</a> )	Function returns the collection of all nodes created automatically during generation of the calculation model. .
	GetCalcSupport ( <a href="#">see page 531</a> )	Function makes available definition of an automatic support at a specified node.
	GetConnectedBars ( <a href="#">see page 531</a> )	Function returns the table with numbers of bars adjoining the specified list of nodes. .
	GetMany ( <a href="#">see page 35</a> )	The function returns a collection of objects that meet the criteria of the indicated selection. .
	GetUniqueId ( <a href="#">see page 532</a> )	Function returns a unique identifier for the specified node.
	GetUserNodes ( <a href="#">see page 532</a> )	Function returns the collection of all user-defined nodes.
	HasCalcSupport ( <a href="#">see page 532</a> )	Function returns True value if a support was automatically created at the specified node during generation of a structure model.
	IsCalc ( <a href="#">see page 533</a> )	Function returns a non-zero value (True), if the node of the specified number has been created automatically during generation of the calculation model. .
	RemoveEmitter ( <a href="#">see page 533</a> )	Function deletes emitter definition from nodes belonging to the specified selection. Available since version 1.7.
	RemoveLabel ( <a href="#">see page 35</a> )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .
	SetEmitter ( <a href="#">see page 533</a> )	Function defines a determined emitter on the specified node list. Available since version 1.7.
	SetLabel ( <a href="#">see page 36</a> )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. .

### II.2.3.2 IRobotNodeServer Fields

The fields of the IRobotNodeServer class are listed here.

#### Public Fields

	Name	Description
	CompatibleNodes ( <a href="#">see page 528</a> )	Compatible node server.
	FreeNumber ( <a href="#">see page 529</a> )	First available user number which can be assigned to the structure node Available since version 1.7.
	NonlinearLinks ( <a href="#">see page 529</a> )	Non-linear connection models.
	RigidLinks ( <a href="#">see page 529</a> )	Rigid link server.

### II.2.3.2.1 CompatibleNodes

#### C++

```
HRESULT get_CompatibleNodes(IROBOTNODECOMPATIBILITYSERVER**);
```

#### C#

```
public IROBOTNODECOMPATIBILITYSERVER CompatibleNodes { get; }
```

#### Visual Basic

```
Public ReadOnly CompatibleNodes As IROBOTNODECOMPATIBILITYSERVER
```

**Description**

Compatible node server.

**Version**

Available since version 2.5.

**II.2.3.2.2 FreeNumber****C++**

```
HRESULT get_FreeNumber(long*);
```

**C#**

```
public long FreeNumber { get; }
```

**Visual Basic**

```
Public ReadOnly FreeNumber As long
```

**Description**

First available user number which can be assigned to the structure node Available since version 1.7.

**II.2.3.2.3 NonlinearLinks****C++**

```
HRESULT get_NonlinearLinks(IRobotNonlinearLinkServer**);
```

**C#**

```
public IRobotNonlinearLinkServer NonlinearLinks { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearLinks As IRobotNonlinearLinkServer
```

**Description**

Non-linear connection models.

**Version**

Available since version 3.

**II.2.3.2.4 RigidLinks****C++**

```
HRESULT get_RigidLinks(IRobotNodeRigidLinkServer**);
```

**C#**

```
public IRobotNodeRigidLinkServer RigidLinks { get; }
```

**Visual Basic**

```
Public ReadOnly RigidLinks As IRobotNodeRigidLinkServer
```

**Description**

Rigid link server.

**Version**

Available since version 2.5.

**II.2.3.3 IRobotNodeServer Methods**

The methods of the IRobotNodeServer class are listed here.

## Public Methods

	Name	Description
ESH	Create (see page 530)	The function creates a new node with the provided parameters. .
ESH	FindWithId (see page 530)	Function returns a number of the node with a specified unique identifier. If such a node is not found, then, zero value is returned. Available since version 1.7.
ESH	GetCalcNodes (see page 531)	Function returns the collection of all nodes created automatically during generation of the calculation model. .
ESH	GetCalcSupport (see page 531)	Function makes available definition of an automatic support at a specified node.
ESH	GetConnectedBars (see page 531)	Function returns the table with numbers of bars adjoining the specified list of nodes. .
ESH	GetUniqueld (see page 532)	Function returns a unique identifier for the specified node.
ESH	GetUserNodes (see page 532)	Function returns the collection of all user-defined nodes.
ESH	HasCalcSupport (see page 532)	Function returns True value if a support was automatically created at the specified node during generation of a structure model.
ESH	IsCalc (see page 533)	Function returns a non-zero value (True), if the node of the specified number has been created automatically during generation of the calculation model .
ESH	RemoveEmitter (see page 533)	Function deletes emitter definition from nodes belonging to the specified selection. Available since version 1.7.
ESH	SetEmitter (see page 533)	Function defines a determined emitter on the specified node list. Available since version 1.7.

### II.2.3.3.1 Create

#### C++

```
HRESULT Create(long _node_number, double _x, double _y, double _z);
```

#### C#

```
public void Create(long _node_number, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub Create(_node_number As long, _x As double, _y As double, _z As double)
```

#### Description

The function creates a new node with the provided parameters. .

### II.2.3.3.2 FindWithId

#### C++

```
HRESULT FindWithId(long _unique_id, long* ret);
```

#### C#

```
public long FindWithId(long _unique_id);
```

#### Visual Basic

```
Public Function FindWithId(_unique_id As long) As long
```

#### Description

Function returns a number of the node with a specified unique identifier. If such a node is not found, then, zero value is returned. Available since version 1.7.

### II.2.3.3.3 GetCalcNodes

#### C++

```
HRESULT GetCalcNodes(IRobotCollection** ret);
```

#### C#

```
public IRobotCollection GetCalcNodes();
```

#### Visual Basic

```
Public Function GetCalcNodes() As IRobotCollection
```

#### Description

Function returns the collection of all nodes created automatically during generation of the calculation model. .

#### Version

Available since version 2.5.

### II.2.3.3.4 GetCalcSupport

#### C++

```
HRESULT GetCalcSupport(long _node_number, IRobotNodeSupportData** ret);
```

#### C#

```
public IRobotNodeSupportData GetCalcSupport(long _node_number);
```

#### Visual Basic

```
Public Function GetCalcSupport(_node_number As long) As IRobotNodeSupportData
```

#### Description

Function makes available definition of an automatic support at a specified node.

#### Version

Available since version 8.

### II.2.3.3.5 GetConnectedBars

#### C++

```
HRESULT GetConnectedBars(BSTR _node_sel, IRobotNumbersArray** ret);
```

#### C#

```
public IRobotNumbersArray GetConnectedBars(String _node_sel);
```

#### Visual Basic

```
Public Function GetConnectedBars(_node_sel As String) As IRobotNumbersArray
```

#### Description

Function returns the table with numbers of bars adjoining the specified list of nodes. .

#### Version

Available since version 3.5.

### II.2.3.3.6 GetUniqueId

#### C++

```
HRESULT GetUniqueId(long _node_num, long* ret);
```

#### C#

```
public long GetUniqueId(long _node_num);
```

#### Visual Basic

```
Public Function GetUniqueId(_node_num As long) As long
```

#### Description

Function returns a unique identifier for the specified node.

#### Version

Available since version 8.2.

### II.2.3.3.7 GetUserNodes

#### C++

```
HRESULT GetUserNodes(IRobotCollection** ret);
```

#### C#

```
public IRobotCollection GetUserNodes();
```

#### Visual Basic

```
Public Function GetUserNodes() As IRobotCollection
```

#### Description

Function returns the collection of all user-defined nodes.

#### Version

Available since version 2.5.

### II.2.3.3.8 HasCalcSupport

#### C++

```
HRESULT HasCalcSupport(long _node_number, VARIANT_BOOL* ret);
```

#### C#

```
public bool HasCalcSupport(long _node_number);
```

#### Visual Basic

```
Public Function HasCalcSupport(_node_number As long) As Boolean
```

#### Description

Function returns True value if a support was automatically created at the specified node during generation of a structure model.

#### Version

Available since version 8.

### II.2.3.3.9 IsCalc

**C++**

```
HRESULT IsCalc(long _node_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsCalc(long _node_num);
```

**Visual Basic**

```
Public Function IsCalc(_node_num As long) As Boolean
```

**Description**

Function returns a non-zero value (True), if the node of the specified number has been created automatically during generation of the calculation model .

**Version**

Available since version 2.5.

### II.2.3.3.10 RemoveEmitter

**C++**

```
HRESULT RemoveEmitter(IRobotSelection* _node_sel);
```

**C#**

```
public void RemoveEmitter(IRobotSelection _node_sel);
```

**Visual Basic**

```
Public Sub RemoveEmitter(ByRef _node_sel As IRobotSelection)
```

**Description**

Function deletes emitter definition from nodes belonging to the specified selection. Available since version 1.7.

### II.2.3.3.11 SetEmitter

**C++**

```
HRESULT SetEmitter(IRobotSelection* _node_sel, IRobotEmitter* _emit);
```

**C#**

```
public void SetEmitter(IRobotSelection _node_sel, IRobotEmitter _emit);
```

**Visual Basic**

```
Public Sub SetEmitter(ByRef _node_sel As IRobotSelection, ByRef _emit As IRobotEmitter)
```

**Description**

Function defines a determined emitter on the specified node list. Available since version 1.7.

## II.3 Bars

### Enumerations

	Name	Description
	IRobotBarTensionCompression (see page 646)	

## Interfaces

	Name	Description
↪	IRobotBar (see page 618)	Each bar in a structure is represented in the Object Model by a data object of the type: I_OT_BAR. The data and functionality related to the bar are available through time history RobotBar interface. The following complex bar attributes are defined by means of the labelling mechanism: Attribute Label type
↪	IRobotBarServer (see page 630)	Server of bars manages the bars defined in the structure .
↪	IRobotBarEnd (see page 642)	One can indicate a beginning and an end of each bar. The object of RobotBarEnd type, describing one of the bar ends, is created to make it possible to carry out operations affecting only the beginning or end of a bar (and not the entire bar).
↪	IRobotBarElement (see page 646)	Bar element.
↪	IRobotBarElementData (see page 649)	Bar (see page 649) element data.
↪	IRobotBarElementDataSet (see page 651)	Data set for bar elements.

### II.3.1 Complex attributes of a bar

Complex attributes of a bar are defined by means of the labelling mechanism. .

## Enumerations

	Name	Description
⌚	IRobotBarEndBracketType (see page 615)	Available types of brackets. .
⌚	IRobotBarEndBracketDataValue (see page 615)	A set of identifiers has been defined that determines particular parameters (attributes), describing a bracket at bar end. The identifier is used as a parameter of the function that gets or sets the value of the relevant bracket attribute. .
⌚	IRobotBarOffsetAutoPosition (see page 618)	Available predefined position of an offset with respect to section. .
⌚	IRobotBarOffsetMemberLength (see page 618)	

## Interfaces

	Name	Description
↪	IRobotBarOffsetData (see page 611)	The structure describing an offset defined for a bar. .
↪	IRobotBarEndOffsetData (see page 613)	The structure describing a shift at one of bar ends.
↪	IRobotBarEndBracketData (see page 615)	The structure describing a bracket at bar end. .

### II.3.1.1 Cables

Available since version 2.5.

## Enumerations

	Name	Description
⌚	IRobotBarCableAssemblingParamType (see page 537)	Available types of assembling parameters.

## Interfaces

	Name	Description
↳	IRobotBarCableData (see page 535)	Definition of cable parameters.

### II.3.1.1.1 IRobotBarCableData

#### Class Hierarchy

#### C++

```
interface IRobotBarCableData : IDispatch;
```

#### C#

```
public interface IRobotBarCableData;
```

#### Visual Basic

```
Public Interface IRobotBarCableData
```

#### Description

Definition of cable parameters.

#### Version

Available since version 2.5.

### II.3.1.1.1.1 IRobotBarCableData Members

The following tables list the members exposed by IRobotBarCableData.

#### Public Fields

	Name	Description
↳	AssemblingParam (see page 535)	Assembling parameter.
↳	AssemblingParamValue (see page 536)	Value of the assembling parameter.
↳	MaterialName (see page 536)	Name of the cable material.
↳	SectionAX (see page 536)	Cable section area.

### II.3.1.1.1.2 IRobotBarCableData Fields

The fields of the IRobotBarCableData class are listed here.

#### Public Fields

	Name	Description
↳	AssemblingParam (see page 535)	Assembling parameter.
↳	AssemblingParamValue (see page 536)	Value of the assembling parameter.
↳	MaterialName (see page 536)	Name of the cable material.
↳	SectionAX (see page 536)	Cable section area.

### II.3.1.1.1.2.1 AssemblingParam

#### C++

```
HRESULT get_AssemblingParam(IRobotBarCableAssemblingParamType* );
HRESULT put_AssemblingParam(IRobotBarCableAssemblingParamType);
```

#### C#

```
public IRobotBarCableAssemblingParamType AssemblingParam { get; set; }
```

**Visual Basic**

```
Public AssemblingParam As IRobotBarCableAssemblingParamType
```

**Description**

Assembling parameter.

**Version**

Available since version 2.5.

**II.3.1.1.2.2 AssemblingParamValue****C++**

```
HRESULT get_AssemblingParamValue(double*);  
HRESULT put_AssemblingParamValue(double);
```

**C#**

```
public double AssemblingParamValue { get; set; }
```

**Visual Basic**

```
Public AssemblingParamValue As Double
```

**Description**

Value of the assembling parameter.

**Version**

Available since version 2.5.

**II.3.1.1.2.3 MaterialName****C++**

```
HRESULT get_MaterialName(BSTR*);  
HRESULT put_MaterialName(BSTR);
```

**C#**

```
public BSTR MaterialName { get; set; }
```

**Visual Basic**

```
Public MaterialName As BSTR
```

**Description**

Name of the cable material.

**Version**

Available since version 2.5.

**II.3.1.1.2.4 SectionAX****C++**

```
HRESULT get_SectionAX(double*);  
HRESULT put_SectionAX(double);
```

**C#**

```
public double SectionAX { get; set; }
```

**Visual Basic**

```
Public SectionAX As Double
```

**Description**

Cable section area.

**Version**

Available since version 2.5.

**II.3.1.1.2 IRobotBarCableAssemblingParamType****C++**

```
enum IRobotBarCableAssemblingParamType;
```

**C#**

```
public enum IRobotBarCableAssemblingParamType;
```

**Visual Basic**

```
Public Enum IRobotBarCableAssemblingParamType
```

**Members**

<b>Members</b>	<b>Description</b>
I_BCAPT_STRESSES_SIGMA = 1	Definition of normal stress as a cable parameter for the assembling load case. Available since version 2.5.
I_BCAPT_FORCE_FO = 2	Definition of tension force as a cable parameter for the assembling load case . Available since version 2.5.
I_BCAPT_LENGTH_L = 3	Definition of the unloaded cable length as a cable parameter for the assembling load case . Available since version 2.5.
I_BCAPT_ELONGATION_DL = 5	Definition of dilatation as a cable parameter for the assembling load case . Available since version 2.5.
I_BCAPT_ELONGATION_DL_RELATIVE = 4	Definition of relative dilatation as a cable parameter for the assembling load case. Available since version 2.5.

**Description**

Available types of assembling parameters.

**Version**

Available since version 2.5.

**II.3.1.2 Elastic ground**

Available since version 2.5.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotUpliftDirection ( <a href="#">see page 540</a> )	Available uplift directions.
	IRobotUpliftSense ( <a href="#">see page 541</a> )	Available uplift definitions in one direction.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotBarElasticGroundData ( <a href="#">see page 538</a> )	Elastic ground parameters for bars. .

### II.3.1.2.1 IRobotBarElasticGroundData

#### Class Hierarchy

#### C++

```
interface IRobotBarElasticGroundData : IDispatch;
```

#### C#

```
public interface IRobotBarElasticGroundData;
```

#### Visual Basic

```
Public Interface IRobotBarElasticGroundData
```

#### Description

Elastic ground parameters for bars. .

#### Version

Available since version 2.5.

### II.3.1.2.1.1 IRobotBarElasticGroundData Members

The following tables list the members exposed by IRobotBarElasticGroundData.

#### Public Fields

	Name	Description
❖	HX (see page 538)	Elastic ground coefficient HX in the RX direction of the global coordinate system.
❖	KY (see page 539)	Elastic ground coefficient KY in the UY direction of the global coordinate system .
❖	KZ (see page 539)	Elastic ground coefficient KZ in the UZ direction of the global coordinate system.

#### Public Methods

	Name	Description
☞	GetOneDir (see page 540)	Function returns uplift definition for the given direction.
☞	SetOneDir (see page 540)	Function defines uplift for the given direction.

### II.3.1.2.1.2 IRobotBarElasticGroundData Fields

The fields of the IRobotBarElasticGroundData class are listed here.

#### Public Fields

	Name	Description
❖	HX (see page 538)	Elastic ground coefficient HX in the RX direction of the global coordinate system.
❖	KY (see page 539)	Elastic ground coefficient KY in the UY direction of the global coordinate system .
❖	KZ (see page 539)	Elastic ground coefficient KZ in the UZ direction of the global coordinate system.

### II.3.1.2.1.2.1 HX

#### C++

```
HRESULT get_HX(double* );
HRESULT put_HX(double);
```

**C#**

```
public double HX { get; set; }
```

**Visual Basic**

```
Public HX As Double
```

**Description**

Elastic ground coefficient HX in the RX direction of the global coordinate system.

**Version**

Available since version 2.5.

**II.3.1.2.1.2.2 KY****C++**

```
HRESULT get_KY(double* );
HRESULT put_KY(double);
```

**C#**

```
public double KY { get; set; }
```

**Visual Basic**

```
Public KY As Double
```

**Description**

Elastic ground coefficient KY in the UY direction of the global coordinate system .

**Version**

Available since version 2.5.

**II.3.1.2.1.2.3 KZ****C++**

```
HRESULT get_KZ(double* );
HRESULT put_KZ(double);
```

**C#**

```
public double KZ { get; set; }
```

**Visual Basic**

```
Public KZ As Double
```

**Description**

Elastic ground coefficient KZ in the UZ direction of the global coordinate system.

**Version**

Available since version 2.5.

**II.3.1.2.1.3 IRobotBarElasticGroundData Methods**

The methods of the IRobotBarElasticGroundData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetOneDir ( <a href="#">see page 540</a> )	Function returns uplift definition for the given direction.
	SetOneDir ( <a href="#">see page 540</a> )	Function defines uplift for the given direction.

### II.3.1.2.1.3.1 GetOneDir

#### C++

```
HRESULT GetOneDir(IRobotUpliftDirection _uplift_dir, IRobotUpliftSense* ret);
```

#### C#

```
public IRobotUpliftSense GetOneDir(IRobotUpliftDirection _uplift_dir);
```

#### Visual Basic

```
Public Function GetOneDir(_uplift_dir As IRobotUpliftDirection) As IRobotUpliftSense
```

#### Description

Function returns uplift definition for the given direction.

#### Version

Available since version 11.

### II.3.1.2.1.3.2 SetOneDir

#### C++

```
HRESULT SetOneDir(IRobotUpliftDirection _uplift_dir, IRobotUpliftSense _uplift_type);
```

#### C#

```
public void SetOneDir(IRobotUpliftDirection _uplift_dir, IRobotUpliftSense _uplift_type);
```

#### Visual Basic

```
Public Sub SetOneDir(_uplift_dir As IRobotUpliftDirection, _uplift_type As IRobotUpliftSense)
```

#### Description

Function defines uplift for the given direction.

#### Version

Available since version 11.

### II.3.1.2.2 IRobotUpliftDirection

#### C++

```
enum IRobotUpliftDirection;
```

#### C#

```
public enum IRobotUpliftDirection;
```

#### Visual Basic

```
Public Enum IRobotUpliftDirection
```

#### Members

Members	Description
I_UD_UY = 1	Uplift in the Y direction. Available since version 11.
I_UD_UZ = 2	Uplift in the Z direction. Available since version 11.

**Description**

Available uplift directions.

**Version**

Available since version 11.

**II.3.1.2.3 IRobotUpliftSense****C++**

```
enum IRobotUpliftSense;
```

**C#**

```
public enum IRobotUpliftSense;
```

**Visual Basic**

```
Public Enum IRobotUpliftSense
```

**Members**

Members	Description
I_US_NONE = 0	No uplift. Available since version 11.
I_US_MINUS = 1	Uplift in the negative direction. Available since version 11.
I_US_PLUS = 2	Uplift in the positive direction. Available since version 11.

**Description**

Available uplift definitions in one direction.

**Version**

Available since version 11.

**II.3.1.3 Sections**

Available since version 2.5.

**Enumerations**

	Name	Description
	IRobotBarSectionShapeType (see page 542)	A set of section shape types recognized by the program. .
	IRobotBarSectionDataValue (see page 544)	A set of identifiers is defined to determine particular parameters (attributes) describing a bar section. Such an identifier is provided as the parameter of a function that gets or sets the value of the relevant section attribute.. .
	IRobotBarSectionType (see page 545)	The set of section types recognized by the program has been defined. .
	IRobotBarSectionNonstdDataValue (see page 546)	A set of identifiers describing section parameters is defined for each type of non-standard user-defined section. .
	IRobotBarSectionConcreteDataValue (see page 546)	Characteristic parameters for RC sections. .
	IRobotBarSectionConcreteCutsPosition (see page 547)	Position of cuts on an RC section.
	IRobotBarSectionComponentShape (see page 571)	Shape types of sections being components of a compound section.

	IRobotBarSectionSpecialDataValue (see page 574)	A set of identifiers for parameters of special steel sections. Parameter names come from the section definition dialog box.
-----------------------------------------------------------------------------------	-------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

## Interfaces

	Name	Description
	IRobotBarSectionData (see page 548)	Data defining a bar section. Object of the type is a Data component of each label of the ILT_BAR_SECTION type. .
	IRobotBarSectionNonstdData (see page 559)	Set of data used for describing the cross-section of a non-standard tapered section. .
	IRobotBarSectionConcreteData (see page 560)	Characteristic parameters for an RC member section. .
	IRobotBarSectionElasticParams (see page 565)	Elasto-plastic properties for sections. .
	IRobotBarSectionComplexData (see page 568)	Definition of compound section. .
	IRobotBarSectionSpecialData (see page 572)	Parameters of a special steel section.

### II.3.1.3.1 IRobotBarSectionShapeType

#### C++

```
enum IRobotBarSectionShapeType;
```

#### C#

```
public enum IRobotBarSectionShapeType;
```

#### Visual Basic

```
Public Enum IRobotBarSectionShapeType
```

#### Members

Members	Description
I_BSST_USER_BOX = 91	Available since version 2.5.
I_BSST_USER_RECT = 92	Available since version 2.5.
I_BSST_USER_TUBE = 93	Available since version 2.5.
I_BSST_USER_I_BISYM = 94	Available since version 2.5.
I_BSST_USER_I_MONOSYM = 95	Available since version 2.5.
I_BSST_USER_T_SHAPE = 96	Available since version 2.5.
I_BSST_USER_C_SHAPE = 97	Available since version 2.5.
I_BSST_USER_CROSS = 201	Available since version 2.5.
I_BSST_WOOD_RECT = 41	Timber rectangular section. Available since version 2.5.
I_BSST_WOOD_DRECT = 99	Timber double rectangular section. Available since version 2.5.
I_BSST_RECT_FILLED = 43	Steel rectangular solid section. Available since version 2.5.
I_BSST_CIRC_FILLED = 44	Steel round solid section. Available since version 2.5.
I_BSST_CONCR_COL_R = -108	Available since version 2.5.
I_BSST_CONCR_COL_T = -107	Available since version 2.5.
I_BSST_CONCR_COL_L = -106	Available since version 2.5.
I_BSST_CONCR_COL_Z = -105	Available since version 2.5.
I_BSST_CONCR_COL_P = -104	Available since version 2.5.
I_BSST_CONCR_COL_C = -103	Available since version 2.5.

I_BSST_CONCR_COL_CH = -102	Available since version 2.5.
I_BSST_CONCR_COL_CQ = -101	Available since version 2.5.
I_BSST_CONCR_BEAM_RECT = -3	RC rectangular beam . Available since version 2.5.
I_BSST_CONCR_BEAM_T = -2	Available since version 2.5.
I_BSST_CONCR_BEAM = -1	RC beam of any shape . Available since version 2.5.
I_BSST_COMP_2C_FACE = 1101	Two C-sections face to face. Available since version 4.
I_BSST_COMP_2C_BACK = 1102	Two C-sections back to back. Available since version 4.
I_BSST_COMP_2I = 1103	Two I-sections. Available since version 4.
I_BSST_COMP_CI = 1104	C-section and I-section. Available since version 4.
I_BSST_COMP_2LI = 1105	Two angles and one I-section. Available since version 4.
I_BSST_COMP_4L_FACE = 1106	Four angles face to face. Available since version 4.
I_BSST_COMP_4L_BACK = 1107	Four angles with legs back to back. Available since version 4.
I_BSST_COMP_2L_SHORT = 1108	Two angles with shorter legs back to back . Available since version 4.
I_BSST_COMP_2L_LONG = 1109	Two angles with longer legs back to back . Available since version 4.
I_BSST_COMP_2L_CROSS = 1110	Two angles - cross shape. Available since version 4.
I_BSST_CCL = 45	Semi-closed C-section.
I_BSST_URND = 46	Rounded C-section - rotated.
I_BSST_TRND = 47	Rectangular pipe.
I_BSST CUAP = 48	C-sections set face to face, welded .
I_BSST_WOOD_CIRC = 100	Round timber section - solid.
I_BSST_USER_CIRC_FILLED = 101	User steel section - round, solid.
I_BSST_USER_POLYGONAL = 103	Pipe section - shaped like a regular polygon.
I_BSST_COLD_C_PLUS = 1010	Cold-formed C-section - semi-closed, with bends.
I_BSST_COLD_SIGMA_SL = 1011	Cold-formed Sigma SL section.
I_BSST_COLD_SIGMA = 1012	Cold-formed Sigma section.
I_BSST_COLD_Z = 1013	Cold-formed Z-section.
I_BSST_COLD_L_LIPS = 1014	Cold-formed C-section - semi-closed.
I_BSST_COLD_Z_ROT = 1015	Cold-formed Z-section in the local - central coordinate system .
I_BSST_COMP_2L_FACE_SHORT = 1111	Two angles - shorter legs .
I_BSST_COMP_2L_FACE_LONG = 1112	Two angles - longer legs.
I_BSST_COMP_CI_BACK = 1113	C-section and I-section set back to back.
I_BSST_COMP_2C_FACE_WELD = 1201	Two C-sections set face to face, welded.
I_BSST_COMP_2C_BACK_WELD = 1202	Two C-sections set back to back, welded.
I_BSST_COMP_2I_WELD = 1203	Two welded I-sections.
I_BSST_COMP_CI_WELD = 1204	C-section and I-section set face to face - welded.

I_BSST_COMP_2LI_WELD = 1205	Two angles and I-section - welded.
I_BSST_COMP_4L_FACE_WELD = 1206	Four angles with legs set face to face - welded.
I_BSST_COMP_4L_BACK_WELD = 1207	Four angles set in the form of a cross - welded.
I_BSST_COMP_2L_SHORT_WELD = 1208	Two angles - shorter legs, welded.
I_BSST_COMP_2L_LONG_WELD = 1209	Two angles - longer legs, welded.
I_BSST_COMP_2L_CROSS_WELD = 1210	Two angles set in the form of a cross - welded .
I_BSST_SPEC_CORRUGATED_WEB = 104	Section with a corrugated web. Available since version 8.5.
I_BSST_SPEC_CASTELLATED_WEB_HEXAGONAL_OPENINGS = 50	Section with a castellated web (hexagonal openings) . Available since version 8.5.
I_BSST_SPEC_CASTELLATED_WEB_HEXAGONAL_OPENINGS_SHIFTED = 52	Section with a castellated web (hexagonal openings, with spacer plates). Available since version 8.5.
I_BSST_SPEC_CASTELLATED_WEB_ROUND_OPENINGS = 51	Section with a castellated web (round openings). Available since version 8.5.
I_BSST_CONCR_BEAM_I = -4	Available since version 8.7.
I_BSST_SPEC_SFB = 53	Available since version 8.7.
I_BSST_SPEC_IFBA = 54	Available since version 8.7.
I_BSST_SPEC_IFBB = 55	Available since version 8.7.
I_BSST_JOIST_K = 500	Standard joist of the K type. Available since version 9.
I_BSST_JOIST_LH = 501	Standard joist of the LH type. Available since version 9.
I_BSST_JOIST_KCS = 502	Joist of the KCS type in which a shear force is taken into account. Available since version 9.
I_BSST_JOIST_DLH = 503	Standard joist of the DLH type. Available since version 9.
I_BSST_USER_BOX_3 = 105	Available since version 14.5.
I_BSST_JOIST_G = 510	Available since version 14.5.
I_BSST_JOIST_VG = 511	Available since version 14.5.
I_BSST_JOIST_BG = 512	Available since version 14.5.
I_BSST_JOIST_SLH = 504	Available since version 14.7.

**Description**

A set of section shape types recognized by the program. .

**II.3.1.3.2 IRobotBarSectionDataValue****C++**

```
enum IRobotBarSectionDataValue;
```

**C#**

```
public enum IRobotBarSectionDataValue;
```

**Visual Basic**

```
Public Enum IRobotBarSectionDataValue
```

## Members

Members	Description
I_BSDV_SURFACE = 10	Painting area (perimeter).
I_BSDV_WEIGHT = 11	Nominal weight per length unit.
I_BSDV_D = 12	Section height - basic (maximum) vertical dimension of a section.
I_BSDV_BF = 13	Section width - basic (maximum) horizontal dimension of a section.
I_BSDV_TW = 14	Web thickness / vertical wall thickness.
I_BSDV_TF = 15	Flange thickness / horizontal wall thickness.
I_BSDV_RA = 16	Fillet radius.
I_BSDV_RI = 17	Fillet radius.
I_BSDV_S = 18	Spacing - distance between section elements (double angles).
I_BSDV_ZY = 19	Section plasticity modulus - bending around Y axis.
I_BSDV_ZZ = 20	Section plasticity modulus - bending around Z axis.
I_BSDV_WX = 21	Section modulus for calculation of torsional stresses.
I_BSDV_WY = 22	Section modulus for calculation of limit shear stresses along Y axis.
I_BSDV_WZ = 23	Section modulus for calculation of limit shear stresses along Z axis.
I_BSDV_GAMMA = 24	Angle between principal and main coordinate system axes.
I_BSDV_IOMEGA = 25	First moment of area.
I_BSDV_P1_LENGTH = 26	Length of plate 1 in the CROSS section type.
I_BSDV_P1_THICKNESS = 27	Thickness of plate 1 in the CROSS section type.
I_BSDV_P2_LENGTH = 28	Length of plate 2 in the CROSS section type.
I_BSDV_P2_THICKNESS = 29	Thickness of plate 2 in the CROSS section type.
I_BSDV_P3_LENGTH = 30	Length of plate 3 in the CROSS section type.
I_BSDV_P3_THICKNESS = 31	Thickness of plate 3 in the CROSS section type.
I_BSDV_P4_LENGTH = 32	Length of plate 4 in the CROSS section type.
I_BSDV_P4_THICKNESS = 33	Thickness of plate 4 in the CROSS section type.
I_BSDV_BF2 = 34	Second section width (compare I_BSDV_BF).
I_BSDV_TF2 = 35	Second flange thickness (compare I_BSDV_TF).
I_BSDV_DIM1 = 36	First dimension.
I_BSDV_DIM2 = 37	Second dimension.
I_BSDV_DIM3 = 38	Third dimension.
I_BSDV_ANGLE1 = 39	Additional angle.
I_BSDV_ANGLE2 = 40	Additional angle.

## Description

A set of identifiers is defined to determine particular parameters (attributes) describing a bar section. Such an identifier is provided as the parameter of a function that gets or sets the value of the relevant section attribute. .

### II.3.1.3.3 IRobotBarSectionType

#### C++

```
enum IRobotBarSectionType;
```

#### C#

```
public enum IRobotBarSectionType;
```

## Visual Basic

```
Public Enum IRobotBarSectionType
```

### Members

Members	Description
I_BST_STANDARD = 0	Standard section (no additional list of data sets describing a tapered section); this section type is used for all sections except for non-standard user tapered sections and compound sections. .
I_BST_COMPLEX = 1000	Compound section. Available since version 4.
I_BST_SPECIAL = 1001	Special sections. Available since version 8.5.
I_BST_JOIST = 1002	Available since version 9.
I_BST_NS_BOX_3 = 20	Available since version 14.5.

### Description

The set of section types recognized by the program has been defined. .

### II.3.1.3.4 IRobotBarSectionNonstdDataValue

#### C++

```
enum IRobotBarSectionNonstdDataValue;
```

#### C#

```
public enum IRobotBarSectionNonstdDataValue;
```

## Visual Basic

```
Public Enum IRobotBarSectionNonstdDataValue
```

### Members

Members	Description
I_BSNDV_POLYGONAL_D_IS_INT = 3	Available since version 8.7.
I_BSNDV_BOX_3_H = 0	Available since version 14.5.
I_BSNDV_BOX_3_B = 1	Available since version 14.5.
I_BSNDV_BOX_3_TW = 2	Available since version 14.5.
I_BSNDV_BOX_3_TF = 3	Available since version 14.5.
I_BSNDV_BOX_3_B1 = 4	Available since version 14.5.
I_BSNDV_BOX_3_B2 = 5	Available since version 14.5.
I_BSNDV_BOX_3_TF2 = 6	Available since version 14.5.

### Description

A set of identifiers describing section parameters is defined for each type of non-standard user-defined section. .

### II.3.1.3.5 IRobotBarSectionConcreteDataValue

#### C++

```
enum IRobotBarSectionConcreteDataValue;
```

#### C#

```
public enum IRobotBarSectionConcreteDataValue;
```

## Visual Basic

```
Public Enum IRobotBarSectionConcreteDataValue
```

### Members

Members	Description
I_BSCDV_COL_H = 0	Available since version 2.5.
I_BSCDV_COL_B = 1	Available since version 2.5.
I_BSCDV_COL_H1 = 2	Available since version 2.5.
I_BSCDV_COL_L1 = 3	Available since version 2.5.
I_BSCDV_COL_H2 = 4	Available since version 2.5.
I_BSCDV_COL_L2 = 5	Available since version 2.5.
I_BSCDV_COL_DE = 6	Available since version 2.5.
I_BSCDV_COL_N = 7	Available since version 2.5.
I_BSCDV_BEAM_H = 0	Available since version 2.5.
I_BSCDV_BEAM_B = 1	Available since version 2.5.
I_BSCDV_BEAM_PL = 2	Available since version 2.5.
I_BSCDV_BEAM_RL = 3	Available since version 2.5.
I_BSCDV_BEAM_EL1 = 4	Available since version 2.5.
I_BSCDV_BEAM_EL2 = 5	Available since version 2.5.
I_BSCDV_BEAM_PR = 6	Available since version 2.5.
I_BSCDV_BEAM_RR = 7	Available since version 2.5.
I_BSCDV_BEAM_ER1 = 8	Available since version 2.5.
I_BSCDV_BEAM_ER2 = 9	Available since version 2.5.
I_BSCDV_BEAM_HG1 = 10	Available since version 2.5.
I_BSCDV_BEAM_BG1 = 11	Available since version 2.5.
I_BSCDV_BEAM_HG2 = 12	Available since version 2.5.
I_BSCDV_BEAM_BG2 = 13	Available since version 2.5.
I_BSCDV_BEAM_T_H = 0	Available since version 2.5.
I_BSCDV_BEAM_T_B = 1	Available since version 2.5.
I_BSCDV_BEAM_T_HF = 100	Available since version 2.5.
I_BSCDV_BEAM_T_BF = 101	Available since version 2.5.
I_BSCDV_BEAM_RECT_H = 0	Available since version 2.5.
I_BSCDV_BEAM_RECT_B = 1	Available since version 2.5.
I_BSCDV_BEAM_I_B = 1	Available since version 8.7.
I_BSCDV_BEAM_I_H = 0	Available since version 8.7.
I_BSCDV_BEAM_I_B1 = 11	Available since version 8.7.
I_BSCDV_BEAM_I_B2 = 13	Available since version 8.7.
I_BSCDV_BEAM_I_HF1 = 10	Available since version 8.7.
I_BSCDV_BEAM_I_HF2 = 12	Available since version 8.7.

### Description

Characteristic parameters for RC sections. .

### Version

Available since version 2.5.

## II.3.1.3.6 IRobotBarSectionConcreteCutsPosition

### C++

```
enum IRobotBarSectionConcreteCutsPosition;
```

**C#**

```
public enum IRobotBarSectionConcreteCutsPosition;
```

**Visual Basic**

```
Public Enum IRobotBarSectionConcreteCutsPosition
```

**Members**

Members	Description
I_BSCCP_NONE = 0	No cuts. Available since version 2.5.
I_BSCCP_LEFT = 1	Cuts on the left side. Available since version 2.5.
I_BSCCP_RIGHT = 2	Cuts on the right side. Available since version 2.5.

**Description**

Position of cuts on an RC section.

**Version**

Available since version 2.5.

**II.3.1.3.7 IRobotBarSectionData****Class Hierarchy****C++**

```
interface IRobotBarSectionData : IDispatch;
```

**C#**

```
public interface IRobotBarSectionData;
```

**Visual Basic**

```
Public Interface IRobotBarSectionData
```

**Description**

Data defining a bar section. Object of the type is a Data component of each label of the ILT\_BAR\_SECTION type. .

**II.3.1.3.7.1 IRobotBarSectionData Members**

The following tables list the members exposed by IRobotBarSectionData.

**Public Fields**

	Name	Description
◆	AliasCount ( [ see page 550] )	Number of additional section names.
◆	Concrete ( [ see page 550] )	RC section parameters.
◆	ElasticParams ( [ see page 551] )	Elasto-plastic properties.
◆	IsConcrete ( [ see page 551] )	Flag indicating if it is an RC section that is dealt with .
◆	IsJoist ( [ see page 551] )	
◆	IsSpecial ( [ see page 551] )	Flag indicating if the user is dealing with a special steel section.
◆	MaterialName ( [ see page 552] )	Name ( [ see page 552] ) of the material associated with a bar section (material type must be consistent with the section type) .

◆	Members ( <a href="#">see page 552</a> )	Component enabling definition of a compound section; This component is active after previous setting the I_BST_COMPLEX value for the Type ( <a href="#">see page 554</a> ) field (section type) and for the ShapeType ( <a href="#">see page 553</a> ) field (shape type) - an appropriate shape available to compound sections..
◆	Name ( <a href="#">see page 552</a> )	Section name .
◆	NonstdCount ( <a href="#">see page 553</a> )	Number of records describing parameters of non-standard tapered sections (for each section of the I_BST_STANDARD type, the value equals 0) .
◆	ShapeType ( <a href="#">see page 553</a> )	Section shape type .
◆	Special ( <a href="#">see page 553</a> )	Set of data describing a special steel section.
◆	Type ( <a href="#">see page 554</a> )	Section type .

### Public Methods

	Name	Description
◆	CalcNonstdGeometry ( <a href="#">see page 555</a> )	The function calculates all the parameters determining section geometry on the basis of the non-standard parameters defined by the user. It should be run for a non-standard tapered section after defining all data sets describing sections in the points of variability. .
◆	CreateNonstd ( <a href="#">see page 555</a> )	The function creates and returns an object (data set) describing the section in a point indicated relatively along the bar (0 means bar beginning, 1 - bar end). The newly-created data structure is added at the end of objects describing a tapered section. The function is used only in the case of non-standard tapered sections. .
◆	DrawSymbol ( <a href="#">see page 555</a> )	Function draws a graphical symbol of a section in the specified area of the window.
◆	FindAlias ( <a href="#">see page 556</a> )	Function searches and returns the additional section name based on a given identifier.
◆	FindNonstd ( <a href="#">see page 556</a> )	The function finds an object describing the cross-section at a bar point indicated relative to its length. It returns the index of the object or zero, when the object is not found. .
◆	GetAlias ( <a href="#">see page 556</a> )	The function returns an additional section name.
◆	GetAliasEx ( <a href="#">see page 556</a> )	Function returns an additional section name and its identifier.
◆	GetNonstd ( <a href="#">see page 557</a> )	A definition of a tapered section consists of a list of objects describing a section in particular points of variability. An index from the range (1 to number of items in the list) is ascribed to each object from the list. The function returns an object with the indicated index. .
◆	GetValue ( <a href="#">see page 557</a> )	The function returns the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .
◆	LoadFromDBase ( <a href="#">see page 557</a> )	The function browses databases for all available sections of the indicated name and reads the data of the found section. If the section of the given name is found and properly read, the function returns the value True (non-zero value); otherwise, it returns False (zero)..
◆	LoadFromDBase2 ( <a href="#">see page 558</a> )	Function searches a section of the specified name in the indicated section database, and next reads data of the section found. If a section of the specified name is found and correctly read, function returns the True value (different from zero), otherwise, it returns the False value (zero).
◆	RemoveNonstd ( <a href="#">see page 558</a> )	The function removes the positions with the indicated index from the list of objects (data sets) describing a tapered section. .
◆	SetValue ( <a href="#">see page 558</a> )	The function sets the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .

### II.3.1.3.7.2 IRobotBarSectionData Fields

The fields of the IRobotBarSectionData class are listed here.

## Public Fields

	Name	Description
◆	AliasCount ( [ see page 550])	Number of additional section names.
◆	Concrete ( [ see page 550])	RC section parameters.
◆	ElasticParams ( [ see page 551])	Elasto-plastic properties.
◆	IsConcrete ( [ see page 551])	Flag indicating if it is an RC section that is dealt with .
◆	IsJoist ( [ see page 551])	
◆	IsSpecial ( [ see page 551])	Flag indicating if the user is dealing with a special steel section.
◆	MaterialName ( [ see page 552])	Name ( [ see page 552]) of the material associated with a bar section (material type must be consistent with the section type) .
◆	Members ( [ see page 552])	Component enabling definition of a compound section; This component is active after previous setting the I_BST_COMPLEX value for the Type ( [ see page 554]) field (section type) and for the ShapeType ( [ see page 553]) field (shape type) - an appropriate shape available to compound sections..
◆	Name ( [ see page 552])	Section name .
◆	NonstdCount ( [ see page 553])	Number of records describing parameters of non-standard tapered sections (for each section of the I_BST_STANDARD type, the value equals 0) .
◆	ShapeType ( [ see page 553])	Section shape type .
◆	Special ( [ see page 553])	Set of data describing a special steel section.
◆	Type ( [ see page 554])	Section type .

### II.3.1.3.7.2.1 AliasCount

#### C++

```
HRESULT get_AliasCount(long*);
```

#### C#

```
public long AliasCount { get; }
```

#### Visual Basic

```
Public ReadOnly AliasCount As long
```

#### Description

Number of additional section names.

#### Version

Available since version 6.1.

### II.3.1.3.7.2.2 Concrete

#### C++

```
HRESULT get_Concrete(IRobotBarSectionConcreteData**);
```

#### C#

```
public IRobotBarSectionConcreteData Concrete { get; }
```

#### Visual Basic

```
Public ReadOnly Concrete As IRobotBarSectionConcreteData
```

#### Description

RC section parameters.

**Version**

Available since version 2.5.

**II.3.1.3.7.2.3 ElasticParams****C++**

```
HRESULT get_ElasticParams(IRobotBarSectionElasticParams**);
```

**C#**

```
public IRobotBarSectionElasticParams ElasticParams { get; }
```

**Visual Basic**

```
Public ReadOnly ElasticParams As IRobotBarSectionElasticParams
```

**Description**

Elasto-plastic properties.

**Version**

Available since version 3.5.

**II.3.1.3.7.2.4 IsConcrete****C++**

```
HRESULT get_IsConcrete(VARIANT_BOOL*);
```

**C#**

```
public bool IsConcrete { get; }
```

**Visual Basic**

```
Public ReadOnly IsConcrete As Boolean
```

**Description**

Flag indicating if it is an RC section that is dealt with .

**Version**

Available since version 2.5.

**II.3.1.3.7.2.5 IsJoist****C++**

```
HRESULT get_IsJoist(VARIANT_BOOL*);
```

**C#**

```
public bool IsJoist { get; }
```

**Visual Basic**

```
Public ReadOnly IsJoist As Boolean
```

**Version**

Available since version 9.

**II.3.1.3.7.2.6 IsSpecial****C++**

```
HRESULT get_IsSpecial(VARIANT_BOOL*);
```

**C#**

```
public bool IsSpecial { get; }
```

**Visual Basic**

```
Public ReadOnly IsSpecial As Boolean
```

**Description**

Flag indicating if the user is dealing with a special steel section.

**Version**

Available since version 8.7.

**II.3.1.3.7.2.7 MaterialName****C++**

```
HRESULT get_MaterialName(BSTR* );
HRESULT put_MaterialName(BSTR);
```

**C#**

```
public String MaterialName { get; set; }
```

**Visual Basic**

```
Public MaterialName As String
```

**Description**

Name (see page 552) of the material associated with a bar section (material type must be consistent with the section type)

**Version**

Available since version 3.

**II.3.1.3.7.2.8 Members****C++**

```
HRESULT get_Members(IRobotBarSectionComplexData** );
```

**C#**

```
public IRobotBarSectionComplexData Members { get; }
```

**Visual Basic**

```
Public ReadOnly Members As IRobotBarSectionComplexData
```

**Description**

Component enabling definition of a compound section; This component is active after previous setting the I\_BST\_COMPLEX value for the Type (see page 554) field (section type) and for the ShapeType (see page 553) field (shape type) - an appropriate shape available to compound sections. .

**Version**

Available since version 4.

**II.3.1.3.7.2.9 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Section name .

**II.3.1.3.7.2.10 NonstdCount****C++**

```
HRESULT get_NonstdCount(long*);
```

**C#**

```
public long NonstdCount { get; }
```

**Visual Basic**

```
Public ReadOnly NonstdCount As long
```

**Description**

Number of records describing parameters of non-standard tapered sections (for each section of the I\_BST\_STANDARD type, the value equals 0) .

**II.3.1.3.7.2.11 ShapeType****C++**

```
HRESULT get_ShapeType(IRobotBarSectionShapeType*);  
HRESULT put_ShapeType(IRobotBarSectionShapeType*);
```

**C#**

```
public IRobotBarSectionShapeType ShapeType { get; set; }
```

**Visual Basic**

```
Public ShapeType As IRobotBarSectionShapeType
```

**Description**

Section shape type .

**II.3.1.3.7.2.12 Special****C++**

```
HRESULT get_Special(IRobotBarSectionSpecialData**);
```

**C#**

```
public IRobotBarSectionSpecialData Special { get; }
```

**Visual Basic**

```
Public ReadOnly Special As IRobotBarSectionSpecialData
```

**Description**

Set of data describing a special steel section.

**Version**

Available since version 8.7.

### II.3.1.3.7.2.13 Type

#### C++

```
HRESULT get_Type(IRobotBarSectionType* );
HRESULT put_Type(IRobotBarSectionType);
```

#### C#

```
public IRobotBarSectionType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotBarSectionType
```

#### Description

Section type.

### II.3.1.3.7.3 IRobotBarSectionData Methods

The methods of the IRobotBarSectionData class are listed here.

#### Public Methods

	Name	Description
✳️	CalcNonstdGeometry (see page 555)	The function calculates all the parameters determining section geometry on the basis of the non-standard parameters defined by the user. It should be run for a non-standard tapered section after defining all data sets describing sections in the points of variability. .
✳️	CreateNonstd (see page 555)	The function creates and returns an object (data set) describing the section in a point indicated relatively along the bar (0 means bar beginning, 1 - bar end). The newly-created data structure is added at the end of objects describing a tapered section. The function is used only in the case of non-standard tapered sections. .
✳️	DrawSymbol (see page 555)	Function draws a graphical symbol of a section in the specified area of the window.
✳️	FindAlias (see page 556)	Function searches and returns the additional section name based on a given identifier.
✳️	FindNonstd (see page 556)	The function finds an object describing the cross-section at a bar point indicated relative to its length. It returns the index of the object or zero, when the object is not found. .
✳️	GetAlias (see page 556)	The function returns an additional section name.
✳️	GetAliasEx (see page 556)	Function returns an additional section name and its identifier.
✳️	GetNonstd (see page 557)	A definition of a tapered section consists of a list of objects describing a section in particular points of variability. An index from the range (1 to number of items in the list) is ascribed to each object from the list. The function returns an object with the indicated index. .
✳️	GetValue (see page 557)	The function returns the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .
✳️	LoadFromDBase (see page 557)	The function browses databases for all available sections of the indicated name and reads the data of the found section. If the section of the given name is found and properly read, the function returns the value True (non-zero value); otherwise, it returns False (zero)..
✳️	LoadFromDBase2 (see page 558)	Function searches a section of the specified name in the indicated section database, and next reads data of the section found. If a section of the specified name is found and correctly read, function returns the True value (different from zero), otherwise, it returns the False value (zero).
✳️	RemoveNonstd (see page 558)	The function removes the positions with the indicated index from the list of objects (data sets) describing a tapered section. .

	SetValue (see page 558)	The function sets the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .
-----------------------------------------------------------------------------------	-------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### II.3.1.3.7.3.1 CalcNonstdGeometry

**C++**

```
HRESULT CalcNonstdGeometry();
```

**C#**

```
public void CalcNonstdGeometry();
```

**Visual Basic**

```
Public Sub CalcNonstdGeometry()
```

**Description**

The function calculates all the parameters determining section geometry on the basis of the non-standard parameters defined by the user. It should be run for a non-standard tapered section after defining all data sets describing sections in the points of variability. .

### II.3.1.3.7.3.2 CreateNonstd

**C++**

```
HRESULT CreateNonstd(double _relative_pos, IRobotBarSectionNonstdData** ret);
```

**C#**

```
public IRobotBarSectionNonstdData CreateNonstd(double _relative_pos);
```

**Visual Basic**

```
Public Function CreateNonstd(_relative_pos As double) As IRobotBarSectionNonstdData
```

**Description**

The function creates and returns an object (data set) describing the section in a point indicated relatively along the bar (0 means bar beginning, 1 - bar end). The newly-created data structure is added at the end of objects describing a tapered section. The function is used only in the case of non-standard tapered sections. .

### II.3.1.3.7.3.3 DrawSymbol

**C++**

```
HRESULT DrawSymbol(long _window_handle, long _left, long _top, long _right, long _bottom);
```

**C#**

```
public void DrawSymbol(long _window_handle, long _left, long _top, long _right, long _bottom);
```

**Visual Basic**

```
Public Sub DrawSymbol(_window_handle As long, _left As long, _top As long, _right As long, _bottom As long)
```

**Description**

Function draws a graphical symbol of a section in the specified area of the window.

**Version**

Available since version 7.5.

#### II.3.1.3.7.3.4 FindAlias

**C++**

```
HRESULT FindAlias(BSTR _alias_id, BSTR _alias, VARIANT_BOOL* ret);
```

**C#**

```
public bool FindAlias(String _alias_id, String _alias);
```

**Visual Basic**

```
Public Function FindAlias(_alias_id As String, ByRef _alias As String) As Boolean
```

**Description**

Function searches and returns the additional section name based on a given identifier.

**Version**

Available since version 7.5.

#### II.3.1.3.7.3.5 FindNonstd

**C++**

```
HRESULT FindNonstd(double _relative_pos, long* ret);
```

**C#**

```
public long FindNonstd(double _relative_pos);
```

**Visual Basic**

```
Public Function FindNonstd(_relative_pos As double) As long
```

**Description**

The function finds an object describing the cross-section at a bar point indicated relative to its length. It returns the index of the object or zero, when the object is not found. .

#### II.3.1.3.7.3.6 GetAlias

**C++**

```
HRESULT GetAlias(long _idx, BSTR* ret);
```

**C#**

```
public String GetAlias(long _idx);
```

**Visual Basic**

```
Public Function GetAlias(_idx As long) As String
```

**Description**

The function returns an additional section name.

**Version**

Available since version 6.1.

#### II.3.1.3.7.3.7 GetAliasEx

**C++**

```
HRESULT GetAliasEx(long _idx, BSTR _alias_id, BSTR* ret);
```

**C#**

```
public String GetAliasEx(long _idx, String _alias_id);
```

**Visual Basic**

```
Public Function GetAliasEx(_idx As long, ByRef _alias_id As String) As String
```

**Description**

Function returns an additional section name and its identifier.

**Version**

Available since version 7.5.

**II.3.1.3.7.3.8 GetNonstd****C++**

```
HRESULT GetNonstd(long _index, IRobotBarSectionNonstdData** ret);
```

**C#**

```
public IRobotBarSectionNonstdData GetNonstd(long _index);
```

**Visual Basic**

```
Public Function GetNonstd(_index As long) As IRobotBarSectionNonstdData
```

**Description**

A definition of a tapered section consists of a list of objects describing a section in particular points of variability. An index from the range (1 to number of items in the list) is ascribed to each object from the list. The function returns an object with the indicated index. .

**II.3.1.3.7.3.9 GetValue****C++**

```
HRESULT GetValue(IRobotBarSectionDataValue _attribute, double* ret);
```

**C#**

```
public double GetValue(IRobotBarSectionDataValue _attribute);
```

**Visual Basic**

```
Public Function GetValue(_attribute As IRobotBarSectionDataValue) As double
```

**Description**

The function returns the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .

**II.3.1.3.7.3.10 LoadFromDBase****C++**

```
HRESULT LoadFromDBase(BSTR _section_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromDBase(String _section_name);
```

**Visual Basic**

```
Public Function LoadFromDBase(_section_name As String) As Boolean
```

## Description

The function browses databases for all available sections of the indicated name and reads the data of the found section. If the section of the given name is found and properly read, the function returns the value True (non-zero value); otherwise, it returns False (zero). .

### II.3.1.3.7.3.11 LoadFromDBase2

#### C++

```
HRESULT LoadFromDBase2(BSTR _section_name, BSTR _dbase_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromDBase2(String _section_name, String _dbase_name);
```

#### Visual Basic

```
Public Function LoadFromDBase2(_section_name As String, _dbase_name As String) As Boolean
```

#### Description

Function searches a section of the specified name in the indicated section database, and next reads data of the section found. If a section of the specified name is found and correctly read, function returns the True value (different from zero), otherwise, it returns the False value (zero).

#### Version

Available since version 7.5.

### II.3.1.3.7.3.12 RemoveNonstd

#### C++

```
HRESULT RemoveNonstd(long _index);
```

#### C#

```
public void RemoveNonstd(long _index);
```

#### Visual Basic

```
Public Sub RemoveNonstd(_index As long)
```

#### Description

The function removes the positions with the indicated index from the list of objects (data sets) describing a tapered section. .

### II.3.1.3.7.3.13 SetValue

#### C++

```
HRESULT SetValue(IRobotBarSectionDataValue _attribute, double _value);
```

#### C#

```
public void SetValue(IRobotBarSectionDataValue _attribute, double _value);
```

#### Visual Basic

```
Public Sub SetValue(_attribute As IRobotBarSectionDataValue, _value As double)
```

#### Description

The function sets the value of the indicated section parameter. Each parameter describing a section has got its identifier. The set of possible identifiers is defined by the RobotBarSectionDataValue type. .

### II.3.1.3.8 IRobotBarSectionNonstdData

#### Class Hierarchy

#### C++

```
interface IRobotBarSectionNonstdData : IDispatch;
```

#### C#

```
public interface IRobotBarSectionNonstdData;
```

#### Visual Basic

```
Public Interface IRobotBarSectionNonstdData
```

#### Description

Set of data used for describing the cross-section of a non-standard tapered section. .

### II.3.1.3.8.1 IRobotBarSectionNonstdData Members

The following tables list the members exposed by IRobotBarSectionNonstdData.

#### Public Fields

	Name	Description
◆	Position ( [ see page 559)	Relative position of a cross-section along a bar where a tapered section has been applied (0 - bar beginning, 1 - bar end) .

#### Public Methods

	Name	Description
◆	GetValue ( [ see page 560)	The function returns tha value of the indicated section parameter. .
◆	SetValue ( [ see page 560)	The function sets the value of the indicated parameter .

### II.3.1.3.8.2 IRobotBarSectionNonstdData Fields

The fields of the IRobotBarSectionNonstdData class are listed here.

#### Public Fields

	Name	Description
◆	Position ( [ see page 559)	Relative position of a cross-section along a bar where a tapered section has been applied (0 - bar beginning, 1 - bar end) .

### II.3.1.3.8.2.1 Position

#### C++

```
HRESULT get_Position(double*);
```

#### C#

```
public double Position { get; }
```

#### Visual Basic

```
Public ReadOnly Position As double
```

#### Description

Relative position of a cross-section along a bar where a tapered section has been applied (0 - bar beginning, 1 - bar end) .

### II.3.1.3.8.3 IRobotBarSectionNonstdData Methods

The methods of the IRobotBarSectionNonstdData class are listed here.

## Public Methods

	Name	Description
💡	GetValue (see page 560)	The function returns the value of the indicated section parameter. .
💡	SetValue (see page 560)	The function sets the value of the indicated parameter .

### II.3.1.3.8.3.1 GetValue

#### C++

```
HRESULT GetValue(IRobotBarSectionNonstdDataValue _attribute, double* ret);
```

#### C#

```
public double GetValue(IRobotBarSectionNonstdDataValue _attribute);
```

#### Visual Basic

```
Public Function GetValue(_attribute As IRobotBarSectionNonstdDataValue) As double
```

#### Description

The function returns the value of the indicated section parameter. .

### II.3.1.3.8.3.2 SetValue

#### C++

```
HRESULT SetValue(IRobotBarSectionNonstdDataValue _attribute, double _value);
```

#### C#

```
public void SetValue(IRobotBarSectionNonstdDataValue _attribute, double _value);
```

#### Visual Basic

```
Public Sub SetValue(_attribute As IRobotBarSectionNonstdDataValue, _value As double)
```

#### Description

The function sets the value of the indicated parameter .

### II.3.1.3.9 IRobotBarSectionConcreteData

#### Class Hierarchy

#### C++

```
interface IRobotBarSectionConcreteData : IDispatch;
```

#### C#

```
public interface IRobotBarSectionConcreteData;
```

#### Visual Basic

```
Public Interface IRobotBarSectionConcreteData
```

#### Description

Characteristic parameters for an RC member section. .

#### Version

Available since version 2.5.

### II.3.1.3.9.1 IRobotBarSectionConcreteData Members

The following tables list the members exposed by IRobotBarSectionConcreteData.

## Public Fields

	Name	Description
◆	BeamCutsPosition ( <a href="#">see page 561</a> )	Position of cuts on a beam.
◆	IsBeam ( <a href="#">see page 562</a> )	
◆	IsColumn ( <a href="#">see page 562</a> )	

## Public Methods

	Name	Description
◆	CalcGeometry ( <a href="#">see page 562</a> )	Function calculates section geometry based on the indicated characteristic values. .
◆	GetReduction ( <a href="#">see page 563</a> )	If coefficients of reduction of the moment of inertia were set earlier, function returns True value and specifies values of coefficients. Otherwise function returns False value.
◆	GetTapered ( <a href="#">see page 563</a> )	Function returns value different from zero (True), if a given section describes an RC beam with variable section width. At the same time, value of the variable section width is entered into the specified parameter. If it is not a section of variable width, then the function returns zero value (False). .
◆	GetValue ( <a href="#">see page 563</a> )	Function returns the value of the selected section parameter. .
◆	SetReduction ( <a href="#">see page 564</a> )	Function sets coefficients of reduction of the moment of inertia.
◆	SetTapered ( <a href="#">see page 564</a> )	Function defines variable section width for an RC beam.
◆	SetValue ( <a href="#">see page 564</a> )	Function sets the value of the selected section parameter. .

### II.3.1.3.9.2 IRobotBarSectionConcreteData Fields

The fields of the IRobotBarSectionConcreteData class are listed here.

## Public Fields

	Name	Description
◆	BeamCutsPosition ( <a href="#">see page 561</a> )	Position of cuts on a beam.
◆	IsBeam ( <a href="#">see page 562</a> )	
◆	IsColumn ( <a href="#">see page 562</a> )	

### II.3.1.3.9.2.1 BeamCutsPosition

#### C++

```
HRESULT get_BeamCutsPosition(IRobotBarSectionConcreteCutsPosition* );
HRESULT put_BeamCutsPosition(IRobotBarSectionConcreteCutsPosition);
```

#### C#

```
public IRobotBarSectionConcreteCutsPosition BeamCutsPosition { get; set; }
```

#### Visual Basic

```
Public BeamCutsPosition As IRobotBarSectionConcreteCutsPosition
```

#### Description

Position of cuts on a beam.

#### Version

Available since version 2.5.

### II.3.1.3.9.2.2 IsBeam

#### C++

```
HRESULT get_IsBeam(VARIANT_BOOL* );
```

#### C#

```
public bool IsBeam { get; }
```

#### Visual Basic

```
Public ReadOnly IsBeam As Boolean
```

#### Version

Available since version 2.5.

### II.3.1.3.9.2.3 IsColumn

#### C++

```
HRESULT get_IsColumn(VARIANT_BOOL* );
```

#### C#

```
public bool IsColumn { get; }
```

#### Visual Basic

```
Public ReadOnly IsColumn As Boolean
```

#### Version

Available since version 2.5.

### II.3.1.3.9.3 IRobotBarSectionConcreteData Methods

The methods of the IRobotBarSectionConcreteData class are listed here.

#### Public Methods

	Name	Description
💡	CalcGeometry (see page 562)	Function calculates section geometry based on the indicated characteristic values. .
💡	GetReduction (see page 563)	If coefficients of reduction of the moment of inertia were set earlier, function returns True value and specifies values of coefficients. Otherwise function returns False value.
💡	GetTapered (see page 563)	Function returns value different from zero (True), if a given section describes an RC beam with variable section width. At the same time, value of the variable section width is entered into the specified parameter. If it is not a section of variable width, then the function returns zero value (False). .
💡	GetValue (see page 563)	Function returns the value of the selected section parameter. .
💡	SetReduction (see page 564)	Function sets coefficients of reduction of the moment of inertia.
💡	SetTapered (see page 564)	Function defines variable section width for an RC beam.
💡	SetValue (see page 564)	Function sets the value of the selected section parameter. .

### II.3.1.3.9.3.1 CalcGeometry

#### C++

```
HRESULT CalcGeometry();
```

#### C#

```
public void CalcGeometry();
```

**Visual Basic**

```
Public Sub CalcGeometry()
```

**Description**

Function calculates section geometry based on the indicated characteristic values. .

**Version**

Available since version 2.5.

**II.3.1.3.9.3.2 GetReduction****C++**

```
HRESULT GetReduction(double* _Ix_cf, double* _Iy_cf, double* _Iz_cf, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetReduction(double* _Ix_cf, double* _Iy_cf, double* _Iz_cf);
```

**Visual Basic**

```
Public Function GetReduction(ByRef _Ix_cf As double*, ByRef _Iy_cf As double*, ByRef _Iz_cf As double*) As Boolean
```

**Description**

If coefficients of reduction of the moment of inertia were set earlier, function returns True value and specifies values of coefficients. Otherwise function returns False value.

**Version**

Available since version 8.

**II.3.1.3.9.3.3 GetTapered****C++**

```
HRESULT GetTapered(double* _h2, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetTapered(double* _h2);
```

**Visual Basic**

```
Public Function GetTapered(ByRef _h2 As double*) As Boolean
```

**Description**

Function returns value different from zero (True), if a given section describes an RC beam with variable section width. At the same time, value of the variable section width is entered into the specified parameter. If it is not a section of variable width, then the function returns zero value (False). .

**Version**

Available since version 2.5.

**II.3.1.3.9.3.4 GetValue****C++**

```
HRESULT GetValue(IRobotBarSectionConcreteDataValue _param, double* ret);
```

**C#**

```
public double GetValue(IRobotBarSectionConcreteDataValue _param);
```

**Visual Basic**

```
Public Function GetValue(_param As IRobotBarSectionConcreteDataValue) As double
```

**Description**

Function returns the value of the selected section parameter. .

**Version**

Available since version 2.5.

**II.3.1.3.9.3.5 SetReduction****C++**

```
HRESULT SetReduction(VARIANT_BOOL _set, double _Ix_cf, double _Iy_cf, double _Iz_cf);
```

**C#**

```
public void SetReduction(bool _set, double _Ix_cf, double _Iy_cf, double _Iz_cf);
```

**Visual Basic**

```
Public Sub SetReduction(_set As Boolean, _Ix_cf As double, _Iy_cf As double, _Iz_cf As double)
```

**Description**

Function sets coefficients of reduction of the moment of inertia.

**Version**

Available since version 8.

**II.3.1.3.9.3.6 SetTapered****C++**

```
HRESULT SetTapered(double _h2);
```

**C#**

```
public void SetTapered(double _h2);
```

**Visual Basic**

```
Public Sub SetTapered(_h2 As double)
```

**Description**

Function defines variable section width for an RC beam.

**Version**

Available since version 2.5.

**II.3.1.3.9.3.7 SetValue****C++**

```
HRESULT SetValue(IRobotBarSectionConcreteDataValue _param, double _val);
```

**C#**

```
public void SetValue(IRobotBarSectionConcreteDataValue _param, double _val);
```

**Visual Basic**

```
Public Sub SetValue(_param As IRobotBarSectionConcreteDataValue, _val As double)
```

## Description

Function sets the value of the selected section parameter. .

## Version

Available since version 2.5.

### II.3.1.3.10 IRobotBarSectionElasticParams

#### Class Hierarchy

#### C++

```
interface IRobotBarSectionElasticParams : IDispatch;
```

#### C#

```
public interface IRobotBarSectionElasticParams;
```

#### Visual Basic

```
Public Interface IRobotBarSectionElasticParams
```

## Description

Elasto-plastic properties for sections. .

## Version

Available since version 3.5.

### II.3.1.3.10.1 IRobotBarSectionElasticParams Members

The following tables list the members exposed by IRobotBarSectionElasticParams.

#### Public Fields

	Name	Description
◆	Active ( [ see page 566)	Flag switching on/off definition of elasto-plastic properties .
◆	L1 ( [ see page 566)	
◆	L2 ( [ see page 566)	
◆	MaterialModel ( [ see page 566)	Material model .
◆	N ( [ see page 567)	
◆	N1 ( [ see page 567)	
◆	N2 ( [ see page 567)	

### II.3.1.3.10.2 IRobotBarSectionElasticParams Fields

The fields of the IRobotBarSectionElasticParams class are listed here.

#### Public Fields

	Name	Description
◆	Active ( [ see page 566)	Flag switching on/off definition of elasto-plastic properties .
◆	L1 ( [ see page 566)	
◆	L2 ( [ see page 566)	
◆	MaterialModel ( [ see page 566)	Material model .
◆	N ( [ see page 567)	
◆	N1 ( [ see page 567)	
◆	N2 ( [ see page 567)	

### II.3.1.3.10.2.1 Active

#### C++

```
HRESULT get_Active(VARIANT_BOOL* );
HRESULT put_Active(VARIANT_BOOL);
```

#### C#

```
public bool Active { get; set; }
```

#### Visual Basic

```
Public Active As Boolean
```

#### Description

Flag switching on/off definition of elasto-plastic properties .

#### Version

Available since version 3.5.

### II.3.1.3.10.2.2 L1

#### C++

```
HRESULT get_L1(double* );
HRESULT put_L1(double);
```

#### C#

```
public double L1 { get; set; }
```

#### Visual Basic

```
Public L1 As Double
```

#### Version

Available since version 3.5.

### II.3.1.3.10.2.3 L2

#### C++

```
HRESULT get_L2(double* );
HRESULT put_L2(double);
```

#### C#

```
public double L2 { get; set; }
```

#### Visual Basic

```
Public L2 As Double
```

#### Version

Available since version 3.5.

### II.3.1.3.10.2.4 MaterialModel

#### C++

```
HRESULT get_MaterialModel(IRobotMaterialElasticModel** );
```

#### C#

```
public IRobotMaterialElasticModel MaterialModel { get; }
```

**Visual Basic**

```
Public ReadOnly MaterialModel As IRobotMaterialElasticModel
```

**Description**

Material model .

**Version**

Available since version 3.5.

**II.3.1.3.10.2.5 N****C++**

```
HRESULT get_N(double* );
HRESULT put_N(double);
```

**C#**

```
public double N { get; set; }
```

**Visual Basic**

```
Public N As double
```

**Version**

Available since version 3.5.

**II.3.1.3.10.2.6 N1****C++**

```
HRESULT get_N1(double* );
HRESULT put_N1(double);
```

**C#**

```
public double N1 { get; set; }
```

**Visual Basic**

```
Public N1 As double
```

**Version**

Available since version 3.5.

**II.3.1.3.10.2.7 N2****C++**

```
HRESULT get_N2(double* );
HRESULT put_N2(double);
```

**C#**

```
public double N2 { get; set; }
```

**Visual Basic**

```
Public N2 As double
```

**Version**

Available since version 3.5.

**II.3.1.3.11 IRobotBarSectionComplexData****Class Hierarchy**

**C++**

```
interface IRobotBarSectionComplexData : IDispatch;
```

**C#**

```
public interface IRobotBarSectionComplexData;
```

**Visual Basic**

```
Public Interface IRobotBarSectionComplexData
```

**Description**

Definition of compound section. .

**Version**

Available since version 4.

**II.3.1.3.11.1 IRobotBarSectionComplexData Members**

The following tables list the members exposed by IRobotBarSectionComplexData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	B ( <a href="#">see page 569</a> )	Spacing b.
❖	Count ( <a href="#">see page 569</a> )	Number of different simple sections composing a compound section; This number depends on the selected compound section shape - the ShapeType element of the IRobotBarSectionData ( <a href="#">see page 548</a> ) interface.. .
❖	D ( <a href="#">see page 569</a> )	Spacing d.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Get ( <a href="#">see page 570</a> )	Function returns section database name and section name for the indicated shape identifier. .
❖	GetShape ( <a href="#">see page 570</a> )	Function returns the shape identifier for a simple section being a component of a compound section.
❖	GetValue ( <a href="#">see page 570</a> )	Function returns value of the selected property for the indicated simple section. .
❖	Set ( <a href="#">see page 571</a> )	Function sets section database name and section name for the indicated shape identifier.

**II.3.1.3.11.2 IRobotBarSectionComplexData Fields**

The fields of the IRobotBarSectionComplexData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	B ( <a href="#">see page 569</a> )	Spacing b.
❖	Count ( <a href="#">see page 569</a> )	Number of different simple sections composing a compound section; This number depends on the selected compound section shape - the ShapeType element of the IRobotBarSectionData ( <a href="#">see page 548</a> ) interface.. .
❖	D ( <a href="#">see page 569</a> )	Spacing d.

**II.3.1.3.11.2.1 B****C++**

```
HRESULT get_B(double*);
```

```

HRESULT put_B(double);

C#
public double B { get; set; }

```

**Visual Basic**

```
Public B As double
```

**Description**

Spacing b.

**Version**

Available since version 4.

**II.3.1.3.11.2.2 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of different simple sections composing a compound section; This number depends on the selected compound section shape - the ShapeType element of the IRobotBarSectionData (see page 548) interface. .

**Version**

Available since version 4.

**II.3.1.3.11.2.3 D****C++**

```

HRESULT get_D(double*);
HRESULT put_D(double);

```

**C#**

```
public double D { get; set; }
```

**Visual Basic**

```
Public D As double
```

**Description**

Spacing d.

**Version**

Available since version 4.

**II.3.1.3.11.3 IRobotBarSectionComplexData Methods**

The methods of the IRobotBarSectionComplexData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	Get (see page 570)	Function returns section database name and section name for the indicated shape identifier. .

	GetShape (see page 570)	Function returns the shape identifier for a simple section being a component of a compound section.
	GetValue (see page 570)	Function returns value of the selected property for the indicated simple section. .
	Set (see page 571)	Function sets section database name and section name for the indicated shape identifier.

### II.3.1.3.11.3.1 Get

#### C++

```
HRESULT Get(IRobotBarSectionComponentShape _shape, BSTR _database_name, BSTR _section_name,
VARIANT_BOOL* ret);
```

#### C#

```
public bool Get(IRobotBarSectionComponentShape _shape, String _database_name, String
_section_name);
```

#### Visual Basic

```
Public Function Get(_shape As IRobotBarSectionComponentShape, ByRef _database_name As
String, ByRef _section_name As String) As Boolean
```

#### Description

Function returns section database name and section name for the indicated shape identifier. .

#### Version

Available since version 4.

### II.3.1.3.11.3.2 GetShape

#### C++

```
HRESULT GetShape(long _idx, IRobotBarSectionComponentShape* ret);
```

#### C#

```
public IRobotBarSectionComponentShape GetShape(long _idx);
```

#### Visual Basic

```
Public Function GetShape(_idx As long) As IRobotBarSectionComponentShape
```

#### Description

Function returns the shape identifier for a simple section being a component of a compound section.

#### Version

Available since version 4.

### II.3.1.3.11.3.3 GetValue

#### C++

```
HRESULT GetValue(IRobotBarSectionComponentShape _shape, IRobotBarSectionDataValue _val_id,
double* ret);
```

#### C#

```
public double GetValue(IRobotBarSectionComponentShape _shape, IRobotBarSectionDataValue
_val_id);
```

#### Visual Basic

```
Public Function GetValue(_shape As IRobotBarSectionComponentShape, _val_id As
```

```
IRobotBarSectionDataValue) As double
```

#### Description

Function returns value of the selected property for the indicated simple section. .

#### Version

Available since version 4.

### II.3.1.3.11.3.4 Set

#### C++

```
HRESULT Set(IRobotBarSectionComponentShape _shape, BSTR _database_name, BSTR _section_name,
VARIANT_BOOL* ret);
```

#### C#

```
public bool Set(IRobotBarSectionComponentShape _shape, String _database_name, String
_section_name);
```

#### Visual Basic

```
Public Function Set(_shape As IRobotBarSectionComponentShape, _database_name As String,
_section_name As String) As Boolean
```

#### Description

Function sets section database name and section name for the indicated shape identifier.

#### Version

Available since version 4.

### II.3.1.3.12 IRobotBarSectionComponentShape

#### C++

```
enum IRobotBarSectionComponentShape;
```

#### C#

```
public enum IRobotBarSectionComponentShape;
```

#### Visual Basic

```
Public Enum IRobotBarSectionComponentShape
```

#### Members

Members	Description
I_BSCS_C = 1	C-section. Available since version 4.
I_BSCS_I = 2	I-section. Available since version 4.
I_BSCS_L = 3	Angle. Available since version 4.
I_BSCS_UNDEFINED = 0	Available since version 4.

#### Description

Shape types of sections being components of a compound section.

#### Version

Available since version 4.

### II.3.1.3.13 IRobotBarSectionSpecialData

#### Class Hierarchy

#### C++

```
interface IRobotBarSectionSpecialData : IDispatch;
```

#### C#

```
public interface IRobotBarSectionSpecialData;
```

#### Visual Basic

```
Public Interface IRobotBarSectionSpecialData
```

#### Description

Parameters of a special steel section.

#### Version

Available since version 8.7.

### II.3.1.3.13.1 IRobotBarSectionSpecialData Members

The following tables list the members exposed by IRobotBarSectionSpecialData.

#### Public Fields

	Name	Description
◆	DbFullName ( <a href="#">see page 572</a> )	Complete section database name.
◆	DbName ( <a href="#">see page 573</a> )	Section ( <a href="#">see page 573</a> ) database name.
◆	Section ( <a href="#">see page 573</a> )	Section name.

#### Public Methods

	Name	Description
◆	GetValue ( <a href="#">see page 574</a> )	Function returns a value of the specified section parameter.
◆	SetValue ( <a href="#">see page 574</a> )	Function sets a value of the specified section parameter.

### II.3.1.3.13.2 IRobotBarSectionSpecialData Fields

The fields of the IRobotBarSectionSpecialData class are listed here.

#### Public Fields

	Name	Description
◆	DbFullName ( <a href="#">see page 572</a> )	Complete section database name.
◆	DbName ( <a href="#">see page 573</a> )	Section ( <a href="#">see page 573</a> ) database name.
◆	Section ( <a href="#">see page 573</a> )	Section name.

### II.3.1.3.13.2.1 DbFullName

#### C++

```
HRESULT get_DbFullName(BSTR* );
HRESULT put_DbFullName(BSTR);
```

#### C#

```
public String DbFullName { get; set; }
```

#### Visual Basic

```
Public DbFullName As String
```

**Description**

Complete section database name.

**Version**

Available since version 8.7.

**II.3.1.3.13.2.2 DbName****C++**

```
HRESULT get_DbName(BSTR* );
HRESULT put_DbName(BSTR);
```

**C#**

```
public String DbName { get; set; }
```

**Visual Basic**

```
Public DbName As String
```

**Description**

Section ([see page 573](#)) database name.

**Version**

Available since version 8.7.

**II.3.1.3.13.2.3 Section****C++**

```
HRESULT get_Section(BSTR* );
HRESULT put_Section(BSTR);
```

**C#**

```
public String Section { get; set; }
```

**Visual Basic**

```
Public Section As String
```

**Description**

Section name.

**Version**

Available since version 8.7.

**II.3.1.3.13.3 IRobotBarSectionSpecialData Methods**

The methods of the IRobotBarSectionSpecialData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	GetValue ( <a href="#">see page 574</a> )	Function returns a value of the specified section parameter.
≡	SetValue ( <a href="#">see page 574</a> )	Function sets a value of the specified section parameter.

**II.3.1.3.13.3.1 GetValue****C++**

```
HRESULT GetValue(IRobotBarSectionSpecialDataValue _param_id, double* ret);
```

**C#**

```
public double GetValue(IRobotBarSectionSpecialDataValue _param_id);
```

**Visual Basic**

```
Public Function GetValue(_param_id As IRobotBarSectionSpecialDataValue) As double
```

**Description**

Function returns a value of the specified section parameter.

**Version**

Available since version 8.7.

**II.3.1.3.13.3.2 SetValue****C++**

```
HRESULT SetValue(IRobotBarSectionSpecialDataValue _param_id, double _param_value);
```

**C#**

```
public void SetValue(IRobotBarSectionSpecialDataValue _param_id, double _param_value);
```

**Visual Basic**

```
Public Sub SetValue(_param_id As IRobotBarSectionSpecialDataValue, _param_value As double)
```

**Description**

Function sets a value of the specified section parameter.

**Version**

Available since version 8.7.

**II.3.1.3.14 IRobotBarSectionSpecialDataValue****C++**

```
enum IRobotBarSectionSpecialDataValue;
```

**C#**

```
public enum IRobotBarSectionSpecialDataValue;
```

**Visual Basic**

```
Public Enum IRobotBarSectionSpecialDataValue
```

**Members**

Members	Description
I_BSSDV_TW = 1	Available since version 8.7.
I_BSSDV_B1 = 2	Available since version 8.7.
I_BSSDV_TF1 = 3	Available since version 8.7.
I_BSSDV_H = 4	Available since version 8.7.
I_BSSDV_B2 = 5	Available since version 8.7.
I_BSSDV_TF2 = 6	Available since version 8.7.
I_BSSDV_D = 7	Available since version 8.7.
I_BSSDV_W = 8	Available since version 8.7.
I_BSSDV_C = 9	Available since version 8.7.
I_BSSDV_A = 10	Available since version 8.7.
I_BSSDV_HS = 11	Available since version 8.7.

I_BSSDV_BP = 12	Available since version 8.7.
I_BSSDV_TP = 13	Available since version 8.7.

**Description**

A set of identifiers for parameters of special steel sections. Parameter names come from the section definition dialog box.

**Version**

Available since version 8.7.

**II.3.1.4 Releases**

Available since version 2.5.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotBarEndReleaseValue (see page 575)	Set of value that can be assumed by the parameters of a release at bar end. .

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotBarReleaseData (see page 576)	The set of data describing a release defined for a bar. The definition of the release consists of two sets of data - each one for each bar end. The data describing the entire release are stored as a data component of the labels of the following type: ILT_BAR_RELEASE. .
	IRobotBarEndReleaseData (see page 577)	Set of parameters defining a release at one bar end. The allowable values are determined by the RobotBarEndReleaseValue type. .

**II.3.1.4.1 IRobotBarEndReleaseValue****C++**

```
enum IRobotBarEndReleaseValue;
```

**C#**

```
public enum IRobotBarEndReleaseValue;
```

**Visual Basic**

```
Public Enum IRobotBarEndReleaseValue
```

**Members**

<b>Members</b>	<b>Description</b>
I_BERV_NONE = 0	None .
I_BERV_ELASTIC = 4	Available since version 1.7.
I_BERV_ELASTIC_PLUS = 5	Available since version 1.7.
I_BERV_ELASTIC_MINUS = 6	Available since version 1.7.
I_BERV_NONLINEAR = 7	Non-linear release defined by the function. Available since version 9.
I_BERV_ELASTIC_REDUCED = 8	Elastic release defined by a partial rigidity coefficient. Available since version 9.
I_BERV_ELASTIC_REDUCED_PLUS = 9	Elastic release defined by a partial rigidity coefficient, unilateral plus. Available since version 9.
I_BERV_ELASTIC_REDUCED_MINUS = 10	Elastic release defined by a partial rigidity coefficient, unilateral minus. Available since version 9.

**Description**

Set of value that can be assumed by the parameters of a release at bar end. .

**II.3.1.4.2 IRobotBarReleaseData****Class Hierarchy****C++**

```
interface IRobotBarReleaseData : IDispatch;
```

**C#**

```
public interface IRobotBarReleaseData;
```

**Visual Basic**

```
Public Interface IRobotBarReleaseData
```

**Description**

The set of data describing a release defined for a bar. The definition of the release consists of two sets of data - each one for each bar end. The data describing the entire release are stored as a data component of the labels of the following type: ILT\_BAR\_RELEASE. .

**II.3.1.4.2.1 IRobotBarReleaseData Members**

The following tables list the members exposed by IRobotBarReleaseData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	EndNode ( [ see page 576) )	Data describing a release at bar end .
◆	StartNode ( [ see page 577) )	Data describing a release at bar beginning .

**II.3.1.4.2.2 IRobotBarReleaseData Fields**

The fields of the IRobotBarReleaseData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	EndNode ( [ see page 576) )	Data describing a release at bar end .
◆	StartNode ( [ see page 577) )	Data describing a release at bar beginning .

**II.3.1.4.2.2.1 EndNode****C++**

```
HRESULT get_EndNode( IRobotBarEndReleaseData** );
HRESULT put_EndNode( IRobotBarEndReleaseData* );
```

**C#**

```
public IRobotBarEndReleaseData EndNode { get; set; }
```

**Visual Basic**

```
Public EndNode As IRobotBarEndReleaseData
```

**Description**

Data describing a release at bar end .

### II.3.1.4.2.2.2 StartNode

#### C++

```
HRESULT get_StartNode(IRobotBarEndReleaseData**);
HRESULT put_StartNode(IRobotBarEndReleaseData*);
```

#### C#

```
public IRobotBarEndReleaseData StartNode { get; set; }
```

#### Visual Basic

```
Public StartNode As IRobotBarEndReleaseData
```

#### Description

Data describing a release at bar beginning .

### II.3.1.4.3 IRobotBarEndReleaseData

#### Class Hierarchy

#### C++

```
interface IRobotBarEndReleaseData : IDispatch;
```

#### C#

```
public interface IRobotBarEndReleaseData;
```

#### Visual Basic

```
Public Interface IRobotBarEndReleaseData
```

#### Description

Set of parameters defining a release at one bar end. The allowable values are determined by the RobotBarEndReleaseValue type. .

### II.3.1.4.3.1 IRobotBarEndReleaseData Members

The following tables list the members exposed by IRobotBarEndReleaseData.

#### Public Fields

	Name	Description
◆	AX ( [ see page 578) )	Damping coefficient (x axis direction).
◆	AY ( [ see page 579) )	Damping coefficient (y axis direction) .
◆	AZ ( [ see page 579) )	Damping coefficient (z axis direction) .
◆	BX ( [ see page 579) )	Bx damping coefficient .
◆	BY ( [ see page 580) )	By damping coefficient .
◆	BZ ( [ see page 580) )	Bz damping coefficient .
◆	HX ( [ see page 580) )	Available since version 1.7.
◆	HY ( [ see page 580) )	Available since version 1.7.
◆	HZ ( [ see page 581) )	Available since version 1.7.
◆	KX ( [ see page 581) )	Available since version 1.7.
◆	KY ( [ see page 581) )	Available since version 1.7.
◆	KZ ( [ see page 581) )	Available since version 1.7.
◆	NonlinearModel ( [ see page 582) )	Non-linearity support.
◆	RX ( [ see page 582) )	
◆	RY ( [ see page 582) )	
◆	RZ ( [ see page 582) )	

◆	UX (see page 583)	
◆	UY (see page 583)	
◆	UZ (see page 583)	

### II.3.1.4.3.2 IRobotBarEndReleaseData Fields

The fields of the IRobotBarEndReleaseData class are listed here.

#### Public Fields

	Name	Description
◆	AX (see page 578)	Damping coefficient (x axis direction).
◆	AY (see page 579)	Damping coefficient (y axis direction) .
◆	AZ (see page 579)	Damping coefficient (z axis direction) .
◆	BX (see page 579)	Bx damping coefficient .
◆	BY (see page 580)	By damping coefficient .
◆	BZ (see page 580)	Bz damping coefficient .
◆	HX (see page 580)	Available since version 1.7.
◆	HY (see page 580)	Available since version 1.7.
◆	HZ (see page 581)	Available since version 1.7.
◆	KX (see page 581)	Available since version 1.7.
◆	KY (see page 581)	Available since version 1.7.
◆	KZ (see page 581)	Available since version 1.7.
◆	NonlinearModel (see page 582)	Non-linearity support.
◆	RX (see page 582)	
◆	RY (see page 582)	
◆	RZ (see page 582)	
◆	UX (see page 583)	
◆	UY (see page 583)	
◆	UZ (see page 583)	

### II.3.1.4.3.2.1 AX

#### C++

```
HRESULT get_AX( double* );
HRESULT put_AX( double );
```

#### C#

```
public double AX { get; set; }
```

#### Visual Basic

```
Public AX As Double
```

#### Description

Damping coefficient (x axis direction).

#### Version

Available since version 2.5.

### II.3.1.4.3.2.2 AY

#### C++

```
HRESULT get_AY( double* );
HRESULT put_AY( double );
```

**C#**

```
public double AY { get; set; }
```

**Visual Basic**

```
Public AY As Double
```

**Description**

Damping coefficient (y axis direction) .

**Version**

Available since version 2.5.

**II.3.1.4.3.2.3 AZ****C++**

```
HRESULT get_AZ( double* );
HRESULT put_AZ( double );
```

**C#**

```
public double AZ { get; set; }
```

**Visual Basic**

```
Public AZ As Double
```

**Description**

Damping coefficient (z axis direction) .

**Version**

Available since version 2.5.

**II.3.1.4.3.2.4 BX****C++**

```
HRESULT get_BX( double* );
HRESULT put_BX( double );
```

**C#**

```
public double BX { get; set; }
```

**Visual Basic**

```
Public BX As Double
```

**Description**

Bx damping coefficient .

**Version**

Available since version 2.5.

**II.3.1.4.3.2.5 BY****C++**

```
HRESULT get_BY( double* );
HRESULT put_BY( double );
```

**C#**

```
public double BY { get; set; }
```

**Visual Basic**

```
Public BY As double
```

**Description**

By damping coefficient .

**Version**

Available since version 2.5.

**II.3.1.4.3.2.6 BZ****C++**

```
HRESULT get_BZ( double* );
HRESULT put_BZ( double );
```

**C#**

```
public double BZ { get; set; }
```

**Visual Basic**

```
Public BZ As double
```

**Description**

Bz damping coefficient .

**Version**

Available since version 2.5.

**II.3.1.4.3.2.7 HX****C++**

```
HRESULT get_HX(double* );
HRESULT put_HX(double );
```

**C#**

```
public double HX { get; set; }
```

**Visual Basic**

```
Public HX As double
```

**Description**

Available since version 1.7.

**II.3.1.4.3.2.8 HY****C++**

```
HRESULT get_HY(double* );
HRESULT put_HY(double );
```

**C#**

```
public double HY { get; set; }
```

**Visual Basic**

```
Public HY As double
```

**Description**

Available since version 1.7.

### II.3.1.4.3.2.9 HZ

#### C++

```
HRESULT get_HZ(double*);  
HRESULT put_HZ(double);
```

#### C#

```
public double HZ { get; set; }
```

#### Visual Basic

```
Public HZ As double
```

#### Description

Available since version 1.7.

### II.3.1.4.3.2.10 KX

#### C++

```
HRESULT get_KX(double*);  
HRESULT put_KX(double);
```

#### C#

```
public double KX { get; set; }
```

#### Visual Basic

```
Public KX As double
```

#### Description

Available since version 1.7.

### II.3.1.4.3.2.11 KY

#### C++

```
HRESULT get_KY(double*);  
HRESULT put_KY(double);
```

#### C#

```
public double KY { get; set; }
```

#### Visual Basic

```
Public KY As double
```

#### Description

Available since version 1.7.

### II.3.1.4.3.2.12 KZ

#### C++

```
HRESULT get_KZ(double*);  
HRESULT put_KZ(double);
```

#### C#

```
public double KZ { get; set; }
```

#### Visual Basic

```
Public KZ As double
```

**Description**

Available since version 1.7.

**II.3.1.4.3.2.13 NonlinearModel****C++**

```
HRESULT get_NonlinearModel(IRobotNonlinearLinkMngr**);
```

**C#**

```
public IRobotNonlinearLinkMngr NonlinearModel { get; }
```

**Visual Basic**

```
Public ReadOnly NonlinearModel As IRobotNonlinearLinkMngr
```

**Description**

Non-linearity support.

**Version**

Available since version 3.

**II.3.1.4.3.2.14 RX****C++**

```
HRESULT get_RX(IRobotBarEndReleaseValue*);  
HRESULT put_RX(IRobotBarEndReleaseValue);
```

**C#**

```
public IRobotBarEndReleaseValue RX { get; set; }
```

**Visual Basic**

```
Public RX As IRobotBarEndReleaseValue
```

**II.3.1.4.3.2.15 RY****C++**

```
HRESULT get_RY(IRobotBarEndReleaseValue*);  
HRESULT put_RY(IRobotBarEndReleaseValue);
```

**C#**

```
public IRobotBarEndReleaseValue RY { get; set; }
```

**Visual Basic**

```
Public RY As IRobotBarEndReleaseValue
```

**II.3.1.4.3.2.16 RZ****C++**

```
HRESULT get_RZ(IRobotBarEndReleaseValue*);  
HRESULT put_RZ(IRobotBarEndReleaseValue);
```

**C#**

```
public IRobotBarEndReleaseValue RZ { get; set; }
```

**Visual Basic**

```
Public RZ As IRobotBarEndReleaseValue
```

### II.3.1.4.3.2.17 UX

#### C++

```
HRESULT get_UX(IRobotBarEndReleaseValue* );
HRESULT put_UX(IRobotBarEndReleaseValue);
```

#### C#

```
public IRobotBarEndReleaseValue UX { get; set; }
```

#### Visual Basic

```
Public UX As IRobotBarEndReleaseValue
```

### II.3.1.4.3.2.18 UY

#### C++

```
HRESULT get_UY(IRobotBarEndReleaseValue* );
HRESULT put_UY(IRobotBarEndReleaseValue);
```

#### C#

```
public IRobotBarEndReleaseValue UY { get; set; }
```

#### Visual Basic

```
Public UY As IRobotBarEndReleaseValue
```

### II.3.1.4.3.2.19 UZ

#### C++

```
HRESULT get_UZ(IRobotBarEndReleaseValue* );
HRESULT put_UZ(IRobotBarEndReleaseValue);
```

#### C#

```
public IRobotBarEndReleaseValue UZ { get; set; }
```

#### Visual Basic

```
Public UZ As IRobotBarEndReleaseValue
```

## II.3.1.5 Geometrical imperfections

Available since version 3.

### Enumerations

	Name	Description
	IRobotBarGeoImperfectionsAxis (see page 587)	

### Interfaces

	Name	Description
	IRobotBarGeoImperfectionsData (see page 583)	Data set describing bar geometrical imperfections. .

### II.3.1.5.1 IRobotBarGeoImperfectionsData

#### Class Hierarchy

#### C++

```
interface IRobotBarGeoImperfectionsData : IDispatch;
```

#### C#

```
public interface IRobotBarGeoImperfectionsData;
```

## Visual Basic

```
Public Interface IRobotBarGeoImperfectionsData
```

### Description

Data set describing bar geometrical imperfections. .

### Version

Available since version 3.

#### II.3.1.5.1.1 IRobotBarGeoImperfectionsData Members

The following tables list the members exposed by IRobotBarGeoImperfectionsData.

### Public Methods

	Name	Description
≡	GetBucklingCoeff ( <a href="#">see page 585</a> )	Function returns value of the buckling length coefficient for the selected direction. This value is relevant only if deflection is calculated automatically according to EC3.
≡	GetUser ( <a href="#">see page 585</a> )	Function returns the user-defined value of initial deflection for the indicated axis. .
≡	IsAutomatic ( <a href="#">see page 585</a> )	Function returns a non-zero value (True), if for a selected axis the initial deflection is calculated automatically according to EC3. .
≡	IsMinus ( <a href="#">see page 586</a> )	Function returns a non-zero value (True), if for the selected axis the negative deflection direction has been set. .
≡	SetAutomatic ( <a href="#">see page 586</a> )	Function allows determining - for a selected axis - if initial deflection is to be calculated automatically according to EC3, or not.
≡	SetMinus ( <a href="#">see page 586</a> )	Function enables change of the deflection direction for the indicated axis.
≡	SetUser ( <a href="#">see page 587</a> )	Function sets the user-defined value of initial deflection for the indicated axis.

#### II.3.1.5.1.2 IRobotBarGeoImperfectionsData Methods

The methods of the IRobotBarGeoImperfectionsData class are listed here.

### Public Methods

	Name	Description
≡	GetBucklingCoeff ( <a href="#">see page 585</a> )	Function returns value of the buckling length coefficient for the selected direction. This value is relevant only if deflection is calculated automatically according to EC3.
≡	GetUser ( <a href="#">see page 585</a> )	Function returns the user-defined value of initial deflection for the indicated axis. .
≡	IsAutomatic ( <a href="#">see page 585</a> )	Function returns a non-zero value (True), if for a selected axis the initial deflection is calculated automatically according to EC3. .
≡	IsMinus ( <a href="#">see page 586</a> )	Function returns a non-zero value (True), if for the selected axis the negative deflection direction has been set. .
≡	SetAutomatic ( <a href="#">see page 586</a> )	Function allows determining - for a selected axis - if initial deflection is to be calculated automatically according to EC3, or not.
≡	SetMinus ( <a href="#">see page 586</a> )	Function enables change of the deflection direction for the indicated axis.
≡	SetUser ( <a href="#">see page 587</a> )	Function sets the user-defined value of initial deflection for the indicated axis.

#### II.3.1.5.1.2.1 GetBucklingCoeff

### C++

```
HRESULT GetBucklingCoeff(RobotBarGeoImperfectionAxis _axis, double* ret);
```

**C#**

```
public double GetBucklingCoeff(RobotBarGeoImperfectionAxis _axis);
```

**Visual Basic**

```
Public Function GetBucklingCoeff(_axis As RobotBarGeoImperfectionAxis) As double
```

**Description**

Function returns value of the buckling length coefficient for the selected direction. This value is relevant only if deflection is calculated automatically according to EC3.

**Version**

Available since version 3.

**II.3.1.5.1.2.2 GetUser****C++**

```
HRESULT GetUser(RobotBarGeoImperfectionAxis _axis, double* ret);
```

**C#**

```
public double GetUser(RobotBarGeoImperfectionAxis _axis);
```

**Visual Basic**

```
Public Function GetUser(_axis As RobotBarGeoImperfectionAxis) As double
```

**Description**

Function returns the user-defined value of initial deflection for the indicated axis. .

**Version**

Available since version 3.

**II.3.1.5.1.2.3 IsAutomatic****C++**

```
HRESULT IsAutomatic(RobotBarGeoImperfectionAxis _axis, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsAutomatic(RobotBarGeoImperfectionAxis _axis);
```

**Visual Basic**

```
Public Function IsAutomatic(_axis As RobotBarGeoImperfectionAxis) As Boolean
```

**Description**

Function returns a non-zero value (True), if for a selected axis the initial deflection is calculated automatically according to EC3. .

**Version**

Available since version 3.

**II.3.1.5.1.2.4 IsMinus****C++**

```
HRESULT IsMinus(RobotBarGeoImperfectionAxis _axis, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsMinus(RobotBarGeoImperfectionAxis _axis);
```

**Visual Basic**

```
Public Function IsMinus(_axis As RobotBarGeoImperfectionAxis) As Boolean
```

**Description**

Function returns a non-zero value (True), if for the selected axis the negative deflection direction has been set. .

**Version**

Available since version 3.

**II.3.1.5.1.2.5 SetAutomatic****C++**

```
HRESULT SetAutomatic(RobotBarGeoImperfectionAxis _axis, VARIANT_BOOL _automatic = True,
double _coeff = 1.0);
```

**C#**

```
public void SetAutomatic(RobotBarGeoImperfectionAxis _axis, bool _automatic = True, double
_coeff = 1.0);
```

**Visual Basic**

```
Public Sub SetAutomatic(_axis As RobotBarGeoImperfectionAxis, Optional _automatic As
Boolean = True, Optional _coeff As Double = 1.0)
```

**Description**

Function allows determining - for a selected axis - if initial deflection is to be calculated automatically according to EC3, or not.

**Version**

Available since version 3.

**II.3.1.5.1.2.6 SetMinus****C++**

```
HRESULT SetMinus(RobotBarGeoImperfectionAxis _axis, VARIANT_BOOL _minus = True);
```

**C#**

```
public void SetMinus(RobotBarGeoImperfectionAxis _axis, bool _minus = True);
```

**Visual Basic**

```
Public Sub SetMinus(_axis As RobotBarGeoImperfectionAxis, Optional _minus As Boolean = True)
```

**Description**

Function enables change of the deflection direction for the indicated axis.

**Version**

Available since version 3.

**II.3.1.5.1.2.7 SetUser****C++**

```
HRESULT SetUser(RobotBarGeoImperfectionAxis _axis, double _val);
```

**C#**

```
public void SetUser(RobotBarGeoImperfectionAxis _axis, double _val);
```

**Visual Basic**

```
Public Sub SetUser(_axis As RobotBarGeoImperfectionAxis, _val As Double)
```

**Description**

Function sets the user-defined value of initial deflection for the indicated axis.

**Version**

Available since version 3.

**II.3.1.5.2 IRobotBarGeoImperfectionsAxis****C++**

```
enum IRobotBarGeoImperfectionsAxis;
```

**C#**

```
public enum IRobotBarGeoImperfectionsAxis;
```

**Visual Basic**

```
Public Enum IRobotBarGeoImperfectionsAxis
```

**Members**

Members	Description
I_BGIA_Z = 1	Local Z axis. Available since version 3.
I_BGIA_Y = 0	Local Y axis. Available since version 3.

**Version**

Available since version 3.

**II.3.1.6 Non-linear hinges**

Available since version 3.

**Enumerations**

	Name	Description
	IRobotNonlinearHingeModelType (see page 598)	Types of non-linear hinge models.
	IRobotNonlinearHingeModelUnloadingType (see page 600)	Types of unloading methods.
	IRobotNonlinearHingeComponentType (see page 604)	Types of components characterizing the non-linear hinge.

**Interfaces**

	Name	Description
	IRobotNonlinearHingeModelServer (see page 588)	Server of non-linear hinge models. .
	IRobotNonlinearHingeModel (see page 590)	Definition of non-linear hinge model.
	IRobotNonlinearHingeModelAxisParams (see page 596)	Interface of parameters whose values depend on the semi-axis type.

	IRobotNonlinearHingeModelPoints ( <a href="#">see page 598</a> )	Interface for definition of curve points.
	IRobotNonlinearHingeData ( <a href="#">see page 601</a> )	Interface of non-linear hinge definition.
	IRobotNonlinearHingeDef ( <a href="#">see page 604</a> )	Interface describing a hinge defined in the structure. .
	IRobotNonlinearHingeServer ( <a href="#">see page 606</a> )	Server providing support for hinge definitions in a structure. .

### II.3.1.6.1 IRobotNonlinearHingeModelServer

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearHingeModelServer : IDispatch;
```

#### C#

```
public interface IRobotNonlinearHingeModelServer;
```

#### Visual Basic

```
Public Interface IRobotNonlinearHingeModelServer
```

#### Description

Server of non-linear hinge models. .

#### Version

Available since version 3.

### II.3.1.6.1.1 IRobotNonlinearHingeModelServer Members

The following tables list the members exposed by IRobotNonlinearHingeModelServer.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 589</a> )	Number of available models.

#### Public Methods

	Name	Description
	Create ( <a href="#">see page 589</a> )	Function creates a new non-linear model of a hinge.
	Delete ( <a href="#">see page 589</a> )	Function deletes model of the indicated index.
	Find ( <a href="#">see page 590</a> )	Function returns the model index based on the specified name.
	Get ( <a href="#">see page 590</a> )	Function takes a non-linear hinge model of the indicated index. .

### II.3.1.6.1.2 IRobotNonlinearHingeModelServer Fields

The fields of the IRobotNonlinearHingeModelServer class are listed here.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 589</a> )	Number of available models.

### II.3.1.6.1.2.1 Count

#### C++

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of available models.

**Version**

Available since version 3.

**II.3.1.6.1.3 IRobotNonlinearHingeModelServer Methods**

The methods of the IRobotNonlinearHingeModelServer class are listed here.

**Public Methods**

	Name	Description
≡	Create (see page 589)	Function creates a new non-linear model of a hinge.
≡	Delete (see page 589)	Function deletes model of the indicated index.
≡	Find (see page 590)	Function returns the model index based on the specified name.
≡	Get (see page 590)	Function takes a non-linear hinge model of the indicated index. .

**II.3.1.6.1.3.1 Create****C++**

```
HRESULT Create(BSTR _name, IRobotNonlinearHingeModel** ret);
```

**C#**

```
public IRobotNonlinearHingeModel Create(String _name);
```

**Visual Basic**

```
Public Function Create(_name As String) As IRobotNonlinearHingeModel
```

**Description**

Function creates a new non-linear model of a hinge.

**Version**

Available since version 3.

**II.3.1.6.1.3.2 Delete****C++**

```
HRESULT Delete(long _idx);
```

**C#**

```
public void Delete(long _idx);
```

**Visual Basic**

```
Public Sub Delete(_idx As Long)
```

**Description**

Function deletes model of the indicated index.

**Version**

Available since version 3.

**II.3.1.6.1.3.3 Find****C++**

```
HRESULT Find(BSTR _name, long* ret);
```

**C#**

```
public long Find(String _name);
```

**Visual Basic**

```
Public Function Find(_name As String) As long
```

**Description**

Function returns the model index based on the specified name.

**Version**

Available since version 3.

**II.3.1.6.1.3.4 Get****C++**

```
HRESULT Get(long _idx, IRobotNonlinearHingeModel** ret);
```

**C#**

```
public IRobotNonlinearHingeModel Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotNonlinearHingeModel
```

**Description**

Function takes a non-linear hinge model of the indicated index. .

**Version**

Available since version 3.

**II.3.1.6.2 IRobotNonlinearHingeModel****Class Hierarchy****C++**

```
interface IRobotNonlinearHingeModel : IDispatch;
```

**C#**

```
public interface IRobotNonlinearHingeModel;
```

**Visual Basic**

```
Public Interface IRobotNonlinearHingeModel
```

**Description**

Definition of non-linear hinge model.

**Version**

Available since version 3.

### II.3.1.6.2.1 IRobotNonlinearHingeModel Members

The following tables list the members exposed by IRobotNonlinearHingeModel.

#### Public Fields

	Name	Description
◆	LimitCoordX ( <a href="#">see page 592</a> )	Flag enforcing automatic estimation of a limit value for the quantity represented by abscissa; Example: for a model of the Force -Displacement type, it involves estimation of the force limit value. .
◆	LimitCoordXValue ( <a href="#">see page 592</a> )	Coefficient used for defining a limit value for the quantity represented by abscissa.
◆	LimitCoordY ( <a href="#">see page 592</a> )	Flag enforcing automatic estimation of a limit value for the quantity represented by ordinate; Example: for a model of the Force -Displacement type, it involves estimation of the displacement limit value. .
◆	LimitCoordYValue ( <a href="#">see page 593</a> )	Coefficient used for defining a limit value for the quantity represented by ordinate.
◆	MixedUnloadingValue ( <a href="#">see page 593</a> )	Value of the unloading factor for MIXED method.
◆	ModelType ( <a href="#">see page 593</a> )	Type of a non-linear hinge model.
◆	Name ( <a href="#">see page 594</a> )	Model name.
◆	Symetry ( <a href="#">see page 594</a> )	Curve symmetry .
◆	UnloadingMethod ( <a href="#">see page 594</a> )	Unloading method.

#### Public Methods

	Name	Description
◆	GetAxisParams ( <a href="#">see page 595</a> )	Function takes the interface of the parameters characteristic of the indicated semi-axis. .
◆	GetPoints ( <a href="#">see page 595</a> )	Function takes the interface for definition of curve points. .
◆	SetAxisParams ( <a href="#">see page 596</a> )	Function updates parameters for the indicated semi-axis.
◆	SetPoints ( <a href="#">see page 596</a> )	Function updates a curve definition based on the specified interface. .

### II.3.1.6.2.2 IRobotNonlinearHingeModel Fields

The fields of the IRobotNonlinearHingeModel class are listed here.

#### Public Fields

	Name	Description
◆	LimitCoordX ( <a href="#">see page 592</a> )	Flag enforcing automatic estimation of a limit value for the quantity represented by abscissa; Example: for a model of the Force -Displacement type, it involves estimation of the force limit value. .
◆	LimitCoordXValue ( <a href="#">see page 592</a> )	Coefficient used for defining a limit value for the quantity represented by abscissa.
◆	LimitCoordY ( <a href="#">see page 592</a> )	Flag enforcing automatic estimation of a limit value for the quantity represented by ordinate; Example: for a model of the Force -Displacement type, it involves estimation of the displacement limit value. .
◆	LimitCoordYValue ( <a href="#">see page 593</a> )	Coefficient used for defining a limit value for the quantity represented by ordinate.
◆	MixedUnloadingValue ( <a href="#">see page 593</a> )	Value of the unloading factor for MIXED method.
◆	ModelType ( <a href="#">see page 593</a> )	Type of a non-linear hinge model.
◆	Name ( <a href="#">see page 594</a> )	Model name.
◆	Symetry ( <a href="#">see page 594</a> )	Curve symmetry .
◆	UnloadingMethod ( <a href="#">see page 594</a> )	Unloading method.

### II.3.1.6.2.2.1 LimitCoordX

#### C++

```
HRESULT get_LimitCoordX(VARIANT_BOOL* );
HRESULT put_LimitCoordX(VARIANT_BOOL);
```

#### C#

```
public bool LimitCoordX { get; set; }
```

#### Visual Basic

```
Public LimitCoordX As Boolean
```

#### Description

Flag enforcing automatic estimation of a limit value for the quantity represented by abscissa; Example: for a model of the Force -Displacement type, it involves estimation of the force limit value. .

#### Version

Available since version 3.

### II.3.1.6.2.2.2 LimitCoordXValue

#### C++

```
HRESULT get_LimitCoordXValue(double* );
HRESULT put_LimitCoordXValue(double);
```

#### C#

```
public double LimitCoordXValue { get; set; }
```

#### Visual Basic

```
Public LimitCoordXValue As Double
```

#### Description

Coefficient used for defining a limit value for the quantity represented by abscissa.

#### Version

Available since version 3.

### II.3.1.6.2.2.3 LimitCoordY

#### C++

```
HRESULT get_LimitCoordY(VARIANT_BOOL* );
HRESULT put_LimitCoordY(VARIANT_BOOL);
```

#### C#

```
public bool LimitCoordY { get; set; }
```

#### Visual Basic

```
Public LimitCoordY As Boolean
```

#### Description

Flag enforcing automatic estimation of a limit value for the quantity represented by ordinate; Example: for a model of the Force -Displacement type, it involves estimation of the displacement limit value. .

#### Version

Available since version 3.

#### II.3.1.6.2.2.4 LimitCoordYValue

##### C++

```
HRESULT get_LimitCoordYValue(double*);  
HRESULT put_LimitCoordYValue(double);
```

##### C#

```
public double LimitCoordYValue { get; set; }
```

##### Visual Basic

```
Public LimitCoordYValue As Double
```

##### Description

Coefficient used for defining a limit value for the quantity represented by ordinate.

##### Version

Available since version 3.

#### II.3.1.6.2.2.5 MixedUnloadingValue

##### C++

```
HRESULT get_MixedUnloadingValue(double*);  
HRESULT put_MixedUnloadingValue(double);
```

##### C#

```
public double MixedUnloadingValue { get; set; }
```

##### Visual Basic

```
Public MixedUnloadingValue As Double
```

##### Description

Value of the unloading factor for MIXED method.

##### Version

Available since version 3.

#### II.3.1.6.2.2.6 ModelType

##### C++

```
HRESULT get_ModelType(IRobotNonlinearHingeModelType*);  
HRESULT put_ModelType(IRobotNonlinearHingeModelType);
```

##### C#

```
public IRobotNonlinearHingeModelType ModelType { get; set; }
```

##### Visual Basic

```
Public ModelType As IRobotNonlinearHingeModelType
```

##### Description

Type of a non-linear hinge model.

##### Version

Available since version 3.

### II.3.1.6.2.2.7 Name

#### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Description

Model name.

#### Version

Available since version 3.

### II.3.1.6.2.2.8 Symetry

#### C++

```
HRESULT get_Symetry(VARIANT_BOOL* );
```

#### C#

```
public bool Symetry { get; }
```

#### Visual Basic

```
Public ReadOnly Symetry As Boolean
```

#### Description

Curve symmetry .

#### Version

Available since version 3.

### II.3.1.6.2.2.9 UnloadingMethod

#### C++

```
HRESULT get_UnloadingMethod(IRobotNonlinearHingeModelUnloadingType* );
HRESULT put_UnloadingMethod(IRobotNonlinearHingeModelUnloadingType);
```

#### C#

```
public IRobotNonlinearHingeModelUnloadingType UnloadingMethod { get; set; }
```

#### Visual Basic

```
Public UnloadingMethod As IRobotNonlinearHingeModelUnloadingType
```

#### Description

Unloading method.

#### Version

Available since version 3.

### II.3.1.6.2.3 IRobotNonlinearHingeModel Methods

The methods of the IRobotNonlinearHingeModel class are listed here.

## Public Methods

	Name	Description
◆	GetAxisParams ( [ see page 595] )	Function takes the interface of the parameters characteristic of the indicated semi-axis..
◆	GetPoints ( [ see page 595] )	Function takes the interface for definition of curve points. .
◆	SetAxisParams ( [ see page 596] )	Function updates parameters for the indicated semi-axis.
◆	SetPoints ( [ see page 596] )	Function updates a curve definition based on the specified interface. .

### II.3.1.6.2.3.1 GetAxisParams

#### C++

```
HRESULT GetAxisParams( IRobotNonlinearLinkSemiAxisType _axis,
RobotNonlinearHingeModeAxisParams* ret);
```

#### C#

```
public RobotNonlinearHingeModeAxisParams GetAxisParams( IRobotNonlinearLinkSemiAxisType _axis);
```

#### Visual Basic

```
Public Function GetAxisParams(_axis As IRobotNonlinearLinkSemiAxisType) As
RobotNonlinearHingeModeAxisParams
```

#### Description

Function takes the interface of the parameters characteristic of the indicated semi-axis. .

#### Version

Available since version 3.

### II.3.1.6.2.3.2 GetPoints

#### C++

```
HRESULT GetPoints( IRobotNonlinearLinkSemiAxisType _axis, IRobotNonlinearHingeModelPoints** ret);
```

#### C#

```
public IRobotNonlinearHingeModelPoints GetPoints( IRobotNonlinearLinkSemiAxisType _axis);
```

#### Visual Basic

```
Public Function GetPoints(_axis As IRobotNonlinearLinkSemiAxisType) As
IRobotNonlinearHingeModelPoints
```

#### Description

Function takes the interface for definition of curve points. .

#### Version

Available since version 3.

### II.3.1.6.2.3.3 SetAxisParams

#### C++

```
HRESULT SetAxisParams( IRobotNonlinearLinkSemiAxisType _axis,
IRobotNonlinearHingeModelAxisParams* _params);
```

#### C#

```
public void SetAxisParams( IRobotNonlinearLinkSemiAxisType _axis,
```

```
IRobotNonlinearHingeModelAxisParams _params);
```

#### Visual Basic

```
Public Sub SetAxisParams(_axis As IRobotNonlinearLinkSemiAxisType, ByRef _params As
IRobotNonlinearHingeModelAxisParams)
```

#### Description

Function updates parameters for the indicated semi-axis.

#### Version

Available since version 3.

### II.3.1.6.2.3.4 SetPoints

#### C++

```
HRESULT SetPoints(IRobotNonlinearLinkSemiAxisType _axis, IRobotNonlinearHingeModelPoints*
_points);
```

#### C#

```
public void SetPoints(IRobotNonlinearLinkSemiAxisType _axis,
IRobotNonlinearHingeModelPoints _points);
```

#### Visual Basic

```
Public Sub SetPoints(_axis As IRobotNonlinearLinkSemiAxisType, ByRef _points As
IRobotNonlinearHingeModelPoints)
```

#### Description

Function updates a curve definition based on the specified interface..

#### Version

Available since version 3.

### II.3.1.6.3 IRobotNonlinearHingeModelAxisParams

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearHingeModelAxisParams : IDispatch;
```

#### C#

```
public interface IRobotNonlinearHingeModelAxisParams;
```

#### Visual Basic

```
Public Interface IRobotNonlinearHingeModelAxisParams
```

#### Description

Interface of parameters whose values depend on the semi-axis type.

#### Version

Available since version 3.

### II.3.1.6.3.1 IRobotNonlinearHingeModelAxisParams Members

The following tables list the members exposed by IRobotNonlinearHingeModelAxisParams.

## Public Fields

	Name	Description
◆	ImmediateOccupancy ( <a href="#">see page 597</a> )	Immediate occupancy.
◆	LifeSafety ( <a href="#">see page 597</a> )	Life safety.
◆	StructuralStability ( <a href="#">see page 598</a> )	Structural stability.

### II.3.1.6.3.2 IRobotNonlinearHingeModelAxisParams Fields

The fields of the IRobotNonlinearHingeModelAxisParams class are listed here.

## Public Fields

	Name	Description
◆	ImmediateOccupancy ( <a href="#">see page 597</a> )	Immediate occupancy.
◆	LifeSafety ( <a href="#">see page 597</a> )	Life safety.
◆	StructuralStability ( <a href="#">see page 598</a> )	Structural stability.

### II.3.1.6.3.2.1 ImmediateOccupancy

#### C++

```
HRESULT get_ImmediateOccupancy(double* );
HRESULT put_ImmediateOccupancy(double);
```

#### C#

```
public double ImmediateOccupancy { get; set; }
```

#### Visual Basic

```
Public ImmediateOccupancy As Double
```

#### Description

Immediate occupancy.

#### Version

Available since version 3.

### II.3.1.6.3.2.2 LifeSafety

#### C++

```
HRESULT get_LifeSafety(double* );
HRESULT put_LifeSafety(double);
```

#### C#

```
public double LifeSafety { get; set; }
```

#### Visual Basic

```
Public LifeSafety As Double
```

#### Description

Life safety.

#### Version

Available since version 3.

### II.3.1.6.3.2.3 StructuralStability

#### C++

```
HRESULT get_StructuralStability(double* );
HRESULT put_StructuralStability(double);
```

#### C#

```
public double StructuralStability { get; set; }
```

#### Visual Basic

```
Public StructuralStability As Double
```

#### Description

Structural stability.

#### Version

Available since version 3.

### II.3.1.6.4 IRobotNonlinearHingeModelType

#### C++

```
enum IRobotNonlinearHingeModelType;
```

#### C#

```
public enum IRobotNonlinearHingeModelType;
```

#### Visual Basic

```
Public Enum IRobotNonlinearHingeModelType
```

#### Members

Members	Description
I_NHMT_FORCE_DISPLACEMENT = 1	Force - displacement. Available since version 3.
I_NHMT_MOMENT_ROTATION = 2	Moment - rotation. Available since version 3.
I_NHMT_STRESS_PAIN = 3	Stress - strain. Available since version 3.

#### Description

Types of non-linear hinge models.

#### Version

Available since version 3.

### II.3.1.6.5 IRobotNonlinearHingeModelPoints

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearHingeModelPoints : IDispatch;
```

#### C#

```
public interface IRobotNonlinearHingeModelPoints;
```

**Visual Basic**

```
Public Interface IRobotNonlinearHingeModelPoints
```

**Description**

Interface for definition of curve points.

**Version**

Available since version 3.

**II.3.1.6.5.1 IRobotNonlinearHingeModelPoints Members**

The following tables list the members exposed by IRobotNonlinearHingeModelPoints.

**Public Fields**

	Name	Description
◆	Count (↗ see page 599)	Number of points defining a curve.

**Public Methods**

	Name	Description
♫	Get (↗ see page 600)	Function takes the interface for definition of a point of the indicated index. .
♫	Set (↗ see page 600)	Function updates point definition based on the indicated interface.

**II.3.1.6.5.2 IRobotNonlinearHingeModelPoints Fields**

The fields of the IRobotNonlinearHingeModelPoints class are listed here.

**Public Fields**

	Name	Description
◆	Count (↗ see page 599)	Number of points defining a curve.

**II.3.1.6.5.2.1 Count****C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of points defining a curve.

**Version**

Available since version 3.

**II.3.1.6.5.3 IRobotNonlinearHingeModelPoints Methods**

The methods of the IRobotNonlinearHingeModelPoints class are listed here.

**Public Methods**

	Name	Description
♫	Get (↗ see page 600)	Function takes the interface for definition of a point of the indicated index. .
♫	Set (↗ see page 600)	Function updates point definition based on the indicated interface.

### II.3.1.6.5.3.1 Get

#### C++

```
HRESULT Get(long _idx, IRobotGeoPoint2D** ret);
```

#### C#

```
public IRobotGeoPoint2D Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As IRobotGeoPoint2D
```

#### Description

Function takes the interface for definition of a point of the indicated index. .

#### Version

Available since version 3.

### II.3.1.6.5.3.2 Set

#### C++

```
HRESULT Set(long _idx, IRobotGeoPoint2D* _point);
```

#### C#

```
public void Set(long _idx, IRobotGeoPoint2D _point);
```

#### Visual Basic

```
Public Sub Set(_idx As long, ByRef _point As IRobotGeoPoint2D)
```

#### Description

Function updates point definition based on the indicated interface.

#### Version

Available since version 3.

### II.3.1.6.6 IRobotNonlinearHingeModelUnloadingType

#### C++

```
enum IRobotNonlinearHingeModelUnloadingType;
```

#### C#

```
public enum IRobotNonlinearHingeModelUnloadingType;
```

#### Visual Basic

```
Public Enum IRobotNonlinearHingeModelUnloadingType
```

#### Members

Members	Description
I_NHMUT_ELASTIC = 1	Available since version 3.
I_NHMUT_PLASTIC = 2	Available since version 3.
I_NHMUT_DAMAGE = 3	Available since version 3.
I_NHMUT_MIXED = 4	Available since version 3.

#### Description

Types of unloading methods.

**Version**

Available since version 3.

**II.3.1.6.7 IRobotNonlinearHingeData****Class Hierarchy****C++**

```
interface IRobotNonlinearHingeData : IDispatch;
```

**C#**

```
public interface IRobotNonlinearHingeData;
```

**Visual Basic**

```
Public Interface IRobotNonlinearHingeData
```

**Description**

Interface of non-linear hinge definition.

**Version**

Available since version 3.

**II.3.1.6.7.1 IRobotNonlinearHingeData Members**

The following tables list the members exposed by IRobotNonlinearHingeData.

**Public Fields**

	Name	Description
!	NormalStress ( <a href="#">see page 602</a> )	Flag enforcing analysis of normal stresses in a bar section with the interactions between individual forces and moments considered .

**Public Methods**

	Name	Description
!	GetModel ( <a href="#">see page 602</a> )	Function takes a model name for the indicated component. .
!	IsActive ( <a href="#">see page 603</a> )	Function checks if the indicated component participates in a hinge definition. .
!	Remove ( <a href="#">see page 603</a> )	Function excludes the indicated component from the hinge definition. .
!	SetModel ( <a href="#">see page 603</a> )	Function ascribes a model of the indicated name to the component.

**II.3.1.6.7.2 IRobotNonlinearHingeData Fields**

The fields of the IRobotNonlinearHingeData class are listed here.

**Public Fields**

	Name	Description
!	NormalStress ( <a href="#">see page 602</a> )	Flag enforcing analysis of normal stresses in a bar section with the interactions between individual forces and moments considered .

**II.3.1.6.7.2.1 NormalStress****C++**

```
HRESULT get_NormalStress(VARIANT_BOOL* );
HRESULT put_NormalStress(VARIANT_BOOL);
```

**C#**

```
public bool NormalStress { get; set; }
```

**Visual Basic**

```
Public NormalStress As Boolean
```

**Description**

Flag enforcing analysis of normal stresses in a bar section with the interactions between individual forces and moments considered.

**Version**

Available since version 3.

**II.3.1.6.7.3 IRobotNonlinearHingeData Methods**

The methods of the IRobotNonlinearHingeData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
♫	GetModel ( <a href="#">see page 602</a> )	Function takes a model name for the indicated component. .
♫	IsActive ( <a href="#">see page 603</a> )	Function checks if the indicated component participates in a hinge definition.. .
♫	Remove ( <a href="#">see page 603</a> )	Function excludes the indicated component from the hinge definition. .
♫	SetModel ( <a href="#">see page 603</a> )	Function ascribes a model of the indicated name to the component.

**II.3.1.6.7.3.1 GetModel****C++**

```
HRESULT GetModel( IRobotNonlinearHingeComponentType _cmpnt, BSTR* ret);
```

**C#**

```
public String GetModel( IRobotNonlinearHingeComponentType _cmpnt);
```

**Visual Basic**

```
Public Function GetModel(_cmpnt As IRobotNonlinearHingeComponentType) As String
```

**Description**

Function takes a model name for the indicated component. .

**Version**

Available since version 3.

**II.3.1.6.7.3.2 IsActive****C++**

```
HRESULT IsActive( IRobotNonlinearHingeComponentType _cmpnt, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsActive( IRobotNonlinearHingeComponentType _cmpnt);
```

**Visual Basic**

```
Public Function IsActive(_cmpnt As IRobotNonlinearHingeComponentType) As Boolean
```

**Description**

Function checks if the indicated component participates in a hinge definition. .

**Version**

Available since version 3.

### II.3.1.6.7.3.3 Remove

#### C++

```
HRESULT Remove(IRobotNonlinearHingeComponentType _cmpnt);
```

#### C#

```
public void Remove(IRobotNonlinearHingeComponentType _cmpnt);
```

#### Visual Basic

```
Public Sub Remove(_cmpnt As IRobotNonlinearHingeComponentType)
```

#### Description

Function excludes the indicated component from the hinge definition. .

#### Version

Available since version 3.

### II.3.1.6.7.3.4 SetModel

#### C++

```
HRESULT SetModel(IRobotNonlinearHingeComponentType _cmpnt, BSTR _model);
```

#### C#

```
public void SetModel(IRobotNonlinearHingeComponentType _cmpnt, String _model);
```

#### Visual Basic

```
Public Sub SetModel(_cmpnt As IRobotNonlinearHingeComponentType, _model As String)
```

#### Description

Function ascribes a model of the indicated name to the component.

#### Version

Available since version 3.

### II.3.1.6.8 IRobotNonlinearHingeComponentType

#### C++

```
enum IRobotNonlinearHingeComponentType;
```

#### C#

```
public enum IRobotNonlinearHingeComponentType;
```

#### Visual Basic

```
Public Enum IRobotNonlinearHingeComponentType
```

#### Members

Members	Description
I_NHCT_FX = 1	Available since version 3.
I_NHCT_FY = 2	Available since version 3.
I_NHCT_FZ = 3	Available since version 3.
I_NHCT_MX = 4	Available since version 3.
I_NHCT_MY = 5	Available since version 3.
I_NHCT_MZ = 6	Available since version 3.

**I\_NHCT\_SX = 7**

Available since version 3.

**Description**

Types of components characterizing the non-linear hinge.

**Version**

Available since version 3.

**II.3.1.6.9 IRobotNonlinearHingeDef****Class Hierarchy****C++**

```
interface IRobotNonlinearHingeDef : IDispatch;
```

**C#**

```
public interface IRobotNonlinearHingeDef;
```

**Visual Basic**

```
Public Interface IRobotNonlinearHingeDef
```

**Description**

Interface describing a hinge defined in the structure. .

**Version**

Available since version 3.

**II.3.1.6.9.1 IRobotNonlinearHingeDef Members**

The following tables list the members exposed by IRobotNonlinearHingeDef.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Bar ( <a href="#">see page 605</a> )	Index of the bar on which a hinge is defined.
❖	LabelName ( <a href="#">see page 605</a> )	Name of the label storing hinge data.
❖	Offset ( <a href="#">see page 606</a> )	Coordinate determining the point on a bar where hinge is defined .
❖	Relative ( <a href="#">see page 606</a> )	Flag defining the manner of interpreting the Offset ( <a href="#">see page 606</a> ) value: true value indicates that the offset coordinate is a relative value of the position on a bar, contained in the interval <0, 1>; false value indicates that the offset is treated as an absolute distance from the bar beginning - interval: <0, bar_length>.

**II.3.1.6.9.2 IRobotNonlinearHingeDef Fields**

The fields of the IRobotNonlinearHingeDef class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Bar ( <a href="#">see page 605</a> )	Index of the bar on which a hinge is defined.
❖	LabelName ( <a href="#">see page 605</a> )	Name of the label storing hinge data.
❖	Offset ( <a href="#">see page 606</a> )	Coordinate determining the point on a bar where hinge is defined .
❖	Relative ( <a href="#">see page 606</a> )	Flag defining the manner of interpreting the Offset ( <a href="#">see page 606</a> ) value: true value indicates that the offset coordinate is a relative value of the position on a bar, contained in the interval <0, 1>; false value indicates that the offset is treated as an absolute distance from the bar beginning - interval: <0, bar_length>.

### II.3.1.6.9.2.1 Bar

#### C++

```
HRESULT get_Bar(long*);  
HRESULT put_Bar(long);
```

#### C#

```
public long Bar { get; set; }
```

#### Visual Basic

```
Public Bar As long
```

#### Description

Index of the bar on which a hinge is defined.

#### Version

Available since version 3.

### II.3.1.6.9.2.2 LabelName

#### C++

```
HRESULT get_LabelName(BSTR*);  
HRESULT put_LabelName(BSTR);
```

#### C#

```
public String LabelName { get; set; }
```

#### Visual Basic

```
Public LabelName As String
```

#### Description

Name of the label storing hinge data.

#### Version

Available since version 3.

### II.3.1.6.9.2.3 Offset

#### C++

```
HRESULT get_Offset(long*);  
HRESULT put_Offset(long);
```

#### C#

```
public long Offset { get; set; }
```

#### Visual Basic

```
Public Offset As long
```

#### Description

Coordinate determining the point on a bar where hinge is defined .

#### Version

Available since version 3.

### II.3.1.6.9.2.4 Relative

#### C++

```
HRESULT get_Relative(VARIANT_BOOL* );
HRESULT put_Relative(VARIANT_BOOL);
```

#### C#

```
public bool Relative { get; set; }
```

#### Visual Basic

```
Public Relative As Boolean
```

#### Description

Flag defining the manner of interpreting the Offset (see page 606) value: true value indicates that the offset coordinate is a relative value of the position on a bar, contained in the interval <0, 1>; false value indicates that the offset is treated as an absolute distance from the bar beginning - interval: <0, bar\_length>.

#### Version

Available since version 3.

### II.3.1.6.10 IRobotNonlinearHingeServer

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearHingeServer : IDispatch;
```

#### C#

```
public interface IRobotNonlinearHingeServer;
```

#### Visual Basic

```
Public Interface IRobotNonlinearHingeServer
```

#### Description

Server providing support for hinge definitions in a structure..

#### Version

Available since version 3.

### II.3.1.6.10.1 IRobotNonlinearHingeServer Members

The following tables list the members exposed by IRobotNonlinearHingeServer.

#### Public Methods

	Name	Description
💡	Count (see page 607)	Function returns the number of hinges defined on the indicated bar (local mode) or defined in a whole structure (global mode). .
💡	Get (see page 608)	Function returns the interface of hinge definition in a structure. It enables work in two modes: in the local mode, by providing the index of a hinge on a bar and bar number and in the global mode, by providing the global index of a hinge in a structure.. .
💡	Remove (see page 608)	Function deletes a definition of a non-linear hinge. It enables work in two modes: in the local mode, by providing the local index of hinge definition and bar number and in the global mode, by providing the global index of hinge definition. .

	Set ( <a href="#">see page 608</a> )	Function defines a hinge on the indicated bar with the indicated offset and returns the global (with respect to a whole structure) index of hinge definition..
-----------------------------------------------------------------------------------	--------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

### II.3.1.6.10.2 IRobotNonlinearHingeServer Methods

The methods of the IRobotNonlinearHingeServer class are listed here.

#### Public Methods

	Name	Description
	Count ( <a href="#">see page 607</a> )	Function returns the number of hinges defined on the indicated bar (local mode) or defined in a whole structure (global mode)..
	Get ( <a href="#">see page 608</a> )	Function returns the interface of hinge definition in a structure. It enables work in two modes: in the local mode, by providing the index of a hinge on a bar and bar number and in the global mode, by providing the global index of a hinge in a structure..
	Remove ( <a href="#">see page 608</a> )	Function deletes a definition of a non-linear hinge. It enables work in two modes: in the local mode, by providing the local index of hinge definition and bar number and in the global mode, by providing the global index of hinge definition..
	Set ( <a href="#">see page 608</a> )	Function defines a hinge on the indicated bar with the indicated offset and returns the global (with respect to a whole structure) index of hinge definition..

### II.3.1.6.10.2.1 Count

#### C++

```
HRESULT Count(long _bar_number = 0, long* ret);
```

#### C#

```
public long Count(long _bar_number = 0);
```

#### Visual Basic

```
Public Function Count(Optional _bar_number As long = 0) As long
```

#### Description

Function returns the number of hinges defined on the indicated bar (local mode) or defined in a whole structure (global mode)..

#### Version

Available since version 3.

### II.3.1.6.10.2.2 Get

#### C++

```
HRESULT Get(long _hinge_idx, long _bar_number = 0, IRobotNonlinearHingeDef** ret);
```

#### C#

```
public IRobotNonlinearHingeDef Get(long _hinge_idx, long _bar_number = 0);
```

#### Visual Basic

```
Public Function Get(_hinge_idx As long, Optional _bar_number As long = 0) As IRobotNonlinearHingeDef
```

#### Description

Function returns the interface of hinge definition in a structure. It enables work in two modes: in the local mode, by providing the index of a hinge on a bar and bar number and in the global mode, by providing the global index of a hinge in a structure..

**Version**

Available since version 3.

**II.3.1.6.10.2.3 Remove****C++**

```
HRESULT Remove(long _hinge_idx, long _bar_number = 0);
```

**C#**

```
public void Remove(long _hinge_idx, long _bar_number = 0);
```

**Visual Basic**

```
Public Sub Remove(_hinge_idx As long, Optional _bar_number As long = 0)
```

**Description**

Function deletes a definition of a non-linear hinge. It enables work in two modes: in the local mode, by providing the local index of hinge definition and bar number and in the global mode, by providing the global index of hinge definition.. .

**Version**

Available since version 3.

**II.3.1.6.10.2.4 Set****C++**

```
HRESULT Set(long _bar_number, BSTR _label, double _offset, VARIANT_BOOL _relative, long* ret);
```

**C#**

```
public long Set(long _bar_number, String _label, double _offset, bool _relative);
```

**Visual Basic**

```
Public Function Set(_bar_number As long, _label As String, _offset As double, _relative As Boolean) As long
```

**Description**

Function defines a hinge on the indicated bar with the indicated offset and returns the global (with respect to a whole structure) index of hinge definition. .

**Version**

Available since version 3.

**II.3.1.7 Claddings**

Available since version 3.5.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotCladdingType (see page 610)	Cladding type. .
	IRobotCladdingMethod (see page 611)	Method of load distribution.

## Interfaces

	Name	Description
↳	IRobotCladdingData (see page 609)	Cladding definition interface. .

### II.3.1.7.1 IRobotCladdingData

#### Class Hierarchy

#### C++

```
interface IRobotCladdingData : IDispatch;
```

#### C#

```
public interface IRobotCladdingData;
```

#### Visual Basic

```
Public Interface IRobotCladdingData
```

#### Description

Cladding definition interface. .

#### Version

Available since version 3.5.

### II.3.1.7.1.1 IRobotCladdingData Members

The following tables list the members exposed by IRobotCladdingData.

#### Public Fields

	Name	Description
❖	Method (see page 610)	Method of load distribution.
❖	Type (see page 610)	Cladding type.

### II.3.1.7.1.2 IRobotCladdingData Fields

The fields of the IRobotCladdingData class are listed here.

#### Public Fields

	Name	Description
❖	Method (see page 610)	Method of load distribution.
❖	Type (see page 610)	Cladding type.

### II.3.1.7.1.2.1 Method

#### C++

```
HRESULT get_Method( IRobotCladdingMethod* );
HRESULT put_Method( IRobotCladdingMethod );
```

#### C#

```
public IRobotCladdingMethod Method { get; set; }
```

#### Visual Basic

```
Public Method As IRobotCladdingMethod
```

#### Description

Method of load distribution.

**Version**

Available since version 8.1.

**II.3.1.7.1.2.2 Type****C++**

```
HRESULT get_Type(IRobotCladdingType* );
HRESULT put_Type(IRobotCladdingType);
```

**C#**

```
public IRobotCladdingType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotCladdingType
```

**Description**

Cladding type.

**Version**

Available since version 3.5.

**II.3.1.7.2 IRobotCladdingType****C++**

```
enum IRobotCladdingType;
```

**C#**

```
public enum IRobotCladdingType;
```

**Visual Basic**

```
Public Enum IRobotCladdingType
```

**Members**

Members	Description
I_CT_X = 0	Anisotropic parallel to local x axis . Available since version 3.5.
I_CT_Y = 1	Anisotropic perpendicular to local x axis . Available since version 3.5.
I_CT_XY = 2	Isotropic . Available since version 3.5.

**Description**

Cladding type. .

**Version**

Available since version 3.5.

**II.3.1.7.3 IRobotCladdingMethod****C++**

```
enum IRobotCladdingMethod;
```

**C#**

```
public enum IRobotCladdingMethod;
```

## Visual Basic

```
Public Enum IRobotCladdingMethod
```

### Members

Members	Description
I_CM_TRAPEZOIDAL = 0	Trapezoidal and triangular method. Available since version 8.1.

### Description

Method of load ditribution.

### Version

Available since version 8.1.

## II.3.1.8 IRobotBarOffsetData

### Class Hierarchy

#### C++

```
interface IRobotBarOffsetData : IDispatch;
```

#### C#

```
public interface IRobotBarOffsetData;
```

### Visual Basic

```
Public Interface IRobotBarOffsetData
```

### Description

The structure describing an offset defined for a bar. .

## II.3.1.8.1 IRobotBarOffsetData Members

The following tables list the members exposed by IRobotBarOffsetData.

### Public Fields

	Name	Description
❖	AxisOffset ( <a href="#">see page 612</a> )	
❖	CoordinateSystem ( <a href="#">see page 612</a> )	The coordinate system in which the offsets at the bar ends are defined.
❖	End ( <a href="#">see page 612</a> )	Offset at the end of a bar.
❖	ObjectNumber ( <a href="#">see page 613</a> )	
❖	Position ( <a href="#">see page 613</a> )	Position with respect to bar section (in local coordinate system) .
❖	Start ( <a href="#">see page 613</a> )	Offset at the beginning of a bar.

## II.3.1.8.2 IRobotBarOffsetData Fields

The fields of the IRobotBarOffsetData class are listed here.

### Public Fields

	Name	Description
❖	AxisOffset ( <a href="#">see page 612</a> )	
❖	CoordinateSystem ( <a href="#">see page 612</a> )	The coordinate system in which the offsets at the bar ends are defined.
❖	End ( <a href="#">see page 612</a> )	Offset at the end of a bar.
❖	ObjectNumber ( <a href="#">see page 613</a> )	

Position (see page 613)	Position with respect to bar section (in local coordinate system).
Start (see page 613)	Offset at the beginning of a bar.

### II.3.1.8.2.1 AxisOffset

#### C++

```
HRESULT get_AxisOffset(IRobotBarOffsetAutoPosition* );
HRESULT put_AxisOffset(IRobotBarOffsetAutoPosition);
```

#### C#

```
public IRobotBarOffsetAutoPosition AxisOffset { get; set; }
```

#### Visual Basic

```
Public AxisOffset As IRobotBarOffsetAutoPosition
```

### II.3.1.8.2.2 CoordinateSystem

#### C++

```
HRESULT get_CoordinateSystem(IRobotGeoCoordinateSystem* );
HRESULT put_CoordinateSystem(IRobotGeoCoordinateSystem);
```

#### C#

```
public IRobotGeoCoordinateSystem CoordinateSystem { get; set; }
```

#### Visual Basic

```
Public CoordinateSystem As IRobotGeoCoordinateSystem
```

#### Description

The coordinate system in which the offsets at the bar ends are defined.

### II.3.1.8.2.3 End

#### C++

```
HRESULT get_End(IRobotBarEndOffsetData** );
```

#### C#

```
public IRobotBarEndOffsetData End { get; }
```

#### Visual Basic

```
Public ReadOnly End As IRobotBarEndOffsetData
```

#### Description

Offset at the end of a bar.

### II.3.1.8.2.4 ObjectNumber

#### C++

```
HRESULT get_ObjectNumber(long* );
HRESULT put_ObjectNumber(long);
```

#### C#

```
public long ObjectNumber { get; set; }
```

#### Visual Basic

```
Public ObjectNumber As long
```

### II.3.1.8.2.5 Position

#### C++

```
HRESULT get_Position(IRobotBarOffsetAutoPosition* );
HRESULT put_Position(IRobotBarOffsetAutoPosition);
```

#### C#

```
public IRobotBarOffsetAutoPosition Position { get; set; }
```

#### Visual Basic

```
Public Position As IRobotBarOffsetAutoPosition
```

#### Description

Position with respect to bar section (in local coordinate system) .

#### Version

Available since version 3.5.

### II.3.1.8.2.6 Start

#### C++

```
HRESULT get_Start(IRobotBarEndOffsetData** );
```

#### C#

```
public IRobotBarEndOffsetData Start { get; }
```

#### Visual Basic

```
Public ReadOnly Start As IRobotBarEndOffsetData
```

#### Description

Offset at the beginning of a bar.

### II.3.1.9 IRobotBarEndOffsetData

#### Class Hierarchy

#### C++

```
interface IRobotBarEndOffsetData : IDispatch;
```

#### C#

```
public interface IRobotBarEndOffsetData;
```

#### Visual Basic

```
Public Interface IRobotBarEndOffsetData
```

#### Description

The structure describing a shift at one of bar ends.

### II.3.1.9.1 IRobotBarEndOffsetData Members

The following tables list the members exposed by IRobotBarEndOffsetData.

#### Public Fields

	Name	Description
❖	MemberLength ( <a href="#">see page 614</a> )	
❖	UX ( <a href="#">see page 614</a> )	

◆	UY (see page 615)	
◆	UZ (see page 615)	

### II.3.1.9.2 IRobotBarEndOffsetData Fields

The fields of the IRobotBarEndOffsetData class are listed here.

#### Public Fields

	Name	Description
◆	MemberLength (see page 614)	
◆	UX (see page 614)	
◆	UY (see page 615)	
◆	UZ (see page 615)	

#### II.3.1.9.2.1 MemberLength

##### C++

```
HRESULT get_MemberLength(IRobotBarOffsetMemberLength* );
HRESULT put_MemberLength(IRobotBarOffsetMemberLength);
```

##### C#

```
public IRobotBarOffsetMemberLength MemberLength { get; set; }
```

##### Visual Basic

```
Public MemberLength As IRobotBarOffsetMemberLength
```

#### II.3.1.9.2.2 UX

##### C++

```
HRESULT get_UX(double* );
HRESULT put_UX(double);
```

##### C#

```
public double UX { get; set; }
```

##### Visual Basic

```
Public UX As double
```

#### II.3.1.9.2.3 UY

##### C++

```
HRESULT get_UY(double* );
HRESULT put_UY(double);
```

##### C#

```
public double UY { get; set; }
```

##### Visual Basic

```
Public UY As double
```

#### II.3.1.9.2.4 UZ

##### C++

```
HRESULT get_UZ(double* );
HRESULT put_UZ(double);
```

##### C#

```
public double UZ { get; set; }
```

**Visual Basic**

```
Public UZ As double
```

**II.3.1.10 IRobotBarEndBracketType****C++**

```
enum IRobotBarEndBracketType;
```

**C#**

```
public enum IRobotBarEndBracketType;
```

**Visual Basic**

```
Public Enum IRobotBarEndBracketType
```

**Description**

Available types of brackets..

**II.3.1.11 IRobotBarEndBracketDataValue****C++**

```
enum IRobotBarEndBracketDataValue;
```

**C#**

```
public enum IRobotBarEndBracketDataValue;
```

**Visual Basic**

```
Public Enum IRobotBarEndBracketDataValue
```

**Description**

A set of identifiers has been defined that determines particular parameters (attributes), describing a bracket at bar end. The identifier is used as a parameter of the function that gets or sets the value of the relevant bracket attribute. .

**II.3.1.12 IRobotBarEndBracketData****Class Hierarchy****C++**

```
interface IRobotBarEndBracketData : IDispatch;
```

**C#**

```
public interface IRobotBarEndBracketData;
```

**Visual Basic**

```
Public Interface IRobotBarEndBracketData
```

**Description**

The structure describing a bracket at bar end. .

**II.3.1.12.1 IRobotBarEndBracketData Members**

The following tables list the members exposed by IRobotBarEndBracketData.

**Public Fields**

	Name	Description
?	Type (see page 616)	Bracket type.

## Public Methods

	Name	Description
💡	GetValue (see page 617)	The function returns the value of the indicated bracket parameter. In order to find out if the returned value is of absolute or relative type, one should call the IsValueRelative (see page 617) function, providing the same identifier of bracket attribute.. .
💡	IsValueRelative (see page 617)	The function returns information about the character of the given bracket parameter. If zero is returned (False), it means that the indicated parameter is determined in absolute quantities, while a non-zero value (True) means that the bracket parameter is determined relative to the parameters of the bar section.. .
💡	SetValue (see page 617)	The function sets the value of the indicated bracket parameter. .

### II.3.1.12.2 IRobotBarEndBracketData Fields

The fields of the IRobotBarEndBracketData class are listed here.

#### Public Fields

	Name	Description
💡	Type (see page 616)	Bracket type.

### II.3.1.12.2.1 Type

#### C++

```
HRESULT get_Type(IRobotBarEndBracketType* );
HRESULT put_Type(IRobotBarEndBracketType);
```

#### C#

```
public IRobotBarEndBracketType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotBarEndBracketType
```

#### Description

Bracket type.

### II.3.1.12.3 IRobotBarEndBracketData Methods

The methods of the IRobotBarEndBracketData class are listed here.

#### Public Methods

	Name	Description
💡	GetValue (see page 617)	The function returns the value of the indicated bracket parameter. In order to find out if the returned value is of absolute or relative type, one should call the IsValueRelative (see page 617) function, providing the same identifier of bracket attribute.. .
💡	IsValueRelative (see page 617)	The function returns information about the character of the given bracket parameter. If zero is returned (False), it means that the indicated parameter is determined in absolute quantities, while a non-zero value (True) means that the bracket parameter is determined relative to the parameters of the bar section.. .
💡	SetValue (see page 617)	The function sets the value of the indicated bracket parameter. .

### II.3.1.12.3.1 GetValue

#### C++

```
HRESULT GetValue(IRobotBarEndBracketDataValue _param, double* ret);
```

**C#**

```
public double GetValue(IRobotBarEndBracketDataValue _param);
```

**Visual Basic**

```
Public Function GetValue(_param As IRobotBarEndBracketDataValue) As Double
```

**Description**

The function returns the value of the indicated bracket parameter. In order to find out if the returned value is of absolute or relative type, one should call the IsValueRelative (see page 617) function, providing the same identifier of bracket attribute. .

**II.3.1.12.3.2 IsValueRelative****C++**

```
HRESULT IsValueRelative(IRobotBarEndBracketDataValue _param, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsValueRelative(IRobotBarEndBracketDataValue _param);
```

**Visual Basic**

```
Public Function IsValueRelative(_param As IRobotBarEndBracketDataValue) As Boolean
```

**Description**

The function returns information about the character of the given bracket parameter. If zero is returned (False), it means that the indicated parameter is determined in absolute quantities, while a non-zero value (True) means that the bracket parameter is determined relative to the parameters of the bar section. .

**II.3.1.12.3.3 SetValue****C++**

```
HRESULT SetValue(IRobotBarEndBracketDataValue _param, double _value, VARIANT_BOOL _relative);
```

**C#**

```
public void SetValue(IRobotBarEndBracketDataValue _param, double _value, bool _relative);
```

**Visual Basic**

```
Public Sub SetValue(_param As IRobotBarEndBracketDataValue, _value As Double, _relative As Boolean)
```

**Description**

The function sets the value of the indicated bracket parameter. .

**II.3.1.13 IRobotBarOffsetAutoPosition****C++**

```
enum IRobotBarOffsetAutoPosition;
```

**C#**

```
public enum IRobotBarOffsetAutoPosition;
```

**Visual Basic**

```
Public Enum IRobotBarOffsetAutoPosition
```

## Members

Members	Description
I_BOAP_0_0 = 0	Available since version 3.5.
I_BOAP_VPY_VPZ = 1	Available since version 3.5.
I_BOAP_0_VPZ = 2	Available since version 3.5.
I_BOAP_VY_VPZ = 3	Available since version 3.5.
I_BOAP_VPY_0 = 4	Available since version 3.5.
I_BOAP_VY_0 = 5	Available since version 3.5.
I_BOAP_VPY_VZ = 6	Available since version 3.5.
I_BOAP_0_VZ = 7	Available since version 3.5.
I_BOAP_VY_VZ = 8	Available since version 3.5.

## Description

Available predefined position of an offset with respect to section. .

## Version

Available since version 3.5.

## II.3.1.14 IRobotBarOffsetMemberLength

### C++

```
enum IRobotBarOffsetMemberLength;
```

### C#

```
public enum IRobotBarOffsetMemberLength;
```

### Visual Basic

```
Public Enum IRobotBarOffsetMemberLength
```

## II.3.2 IRobotBar

### Class Hierarchy

### C++

```
interface IRobotBar : IRobotDataObject;
```

### C#

```
public interface IRobotBar : IRobotDataObject;
```

### Visual Basic

```
Public Interface IRobotBar
```

## Description

Each bar in a structure is represented in the Object Model by a data object of the type: I\_OT\_BAR. The data and functionality related to the bar are available through time history RobotBar interface. The following complex bar attributes are defined by means of the labelling mechanism:

Attribute Label type

---

Section I\_LT\_BAR\_SECTION Release I\_LT\_BAR\_RELEASE Material I\_LT\_BAR\_MATERIAL Offset I\_LT\_BAR\_OFFSET .

### II.3.2.1 IRobotBar Members

The following tables list the members exposed by IRobotBar.

#### Public Fields

	Name	Description
◆	AnalyzeTTMethod ( <a href="#">see page 621</a> )	A flag indicating if the member is taken into consideration in the triangular / trapezoidal method.
◆	Elements ( <a href="#">see page 621</a> )	Collection containing one or more bar elements defined for the bar (the collection components are objects of the IRobotBarElement ( <a href="#">see page 646</a> ) type).
◆	End ( <a href="#">see page 622</a> )	Bar end.
◆	EndNode ( <a href="#">see page 622</a> )	Number ( <a href="#">see page 30</a> ) of the end bar node.
◆	Gamma ( <a href="#">see page 622</a> )	Gamma angle value defined for the bar. The gamma angle may be defined both for a bar and for a section. The resultant angle is a sum of both values. The value describes the angle for the bar. .
◆	InactiveBar ( <a href="#">see page 623</a> )	Flag enabling exclusion of a bar from the structure model - making it "inactive".
◆	IsSuperBar ( <a href="#">see page 623</a> )	Flag indicating that the bar is a superbar.
◆	Length ( <a href="#">see page 623</a> )	Bar length.
◆	Name ( <a href="#">see page 623</a> )	Bar name.
◆	NameTemplate ( <a href="#">see page 624</a> )	Bar name pattern.
◆	Number ( <a href="#">see page 30</a> )	Object number is assigned by the user; it is unique among all the components of the same type. .
◆	ReversedOffset ( <a href="#">see page 624</a> )	Flag indicating if offset assigned to a bar is reversed with respect to bar orientation .
◆	ReversedRelease ( <a href="#">see page 624</a> )	Flag indicating if the member release is reversed in relation to member orientation.
◆	ReversedSection ( <a href="#">see page 625</a> )	Flag indicating if section assigned to the bar is inverted (section beginning is applied to the bar end).
◆	ShearForces ( <a href="#">see page 625</a> )	Flag indicating if shearing forces acting on a bar will be included in calculations of structure deformation.
◆	Start ( <a href="#">see page 625</a> )	Bar beginning .
◆	StartNode ( <a href="#">see page 626</a> )	Number ( <a href="#">see page 30</a> ) of the initial bar node .
◆	StructuralType ( <a href="#">see page 626</a> )	Type of structure object.
◆	TensionCompression ( <a href="#">see page 626</a> )	Manner in which the bar carries stresses.
◆	TrussBar ( <a href="#">see page 627</a> )	Flag indicating if the bar is a truss bar.
◆	UniqueId ( <a href="#">see page 627</a> )	Unique identifier of structure component Available since version 1.7.

#### Public Methods

	Name	Description
◆	CalcGamma ( <a href="#">see page 628</a> )	Function calculates a Gamma ( <a href="#">see page 622</a> ) angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.
◆	ChangeOrientation ( <a href="#">see page 628</a> )	Function changes bar orientation to opposite.
◆	GetElemsData ( <a href="#">see page 628</a> )	Function fills out the specified object with data of elements belonging to the bar.
◆	GetLabel ( <a href="#">see page 31</a> )	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabelName ( <a href="#">see page 31</a> )	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .

	GetLabels ( <a href="#">see page 31</a> )	The function returns a collection containing all labels (of different types) defined for a given object. .
	GetLCS ( <a href="#">see page 629</a> )	Function returns Local Coordinate System for the bar.
	GetSimpleBars ( <a href="#">see page 629</a> )	Function returns the selection of simple bars which are the superbar components. Prior to activating the function, the IsSuperBar ( <a href="#">see page 623</a> ) flag should be checked. If the function is activated for a simple bar, it causes a critical error.
	HasLabel ( <a href="#">see page 32</a> )	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
	RemoveLabel ( <a href="#">see page 32</a> )	The function deletes a label of the indicated type. .
	SetLabel ( <a href="#">see page 32</a> )	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .
	SetOffset ( <a href="#">see page 629</a> )	Function assigns offset with a specified name to a bar, identically as the SetLabel ( <a href="#">see page 32</a> ) function. In addition, it enables reversing the offset. .
	SetOrientation ( <a href="#">see page 630</a> )	Function adjusts bar orientation to the defined sense of the coordinate system axis.
	SetSection ( <a href="#">see page 630</a> )	Function assigns section of the indicated name to the bar, just as the SetLabel ( <a href="#">see page 32</a> ) function does. Moreover, it allows inverting the section - applying section end to the bar beginning. .

### II.3.2.2 IRobotBar Fields

The fields of the IRobotBar class are listed here.

#### Public Fields

	Name	Description
	AnalyzeTTMethod ( <a href="#">see page 621</a> )	A flag indicating if the member is taken into consideration in the triangular / trapezoidal method.
	Elements ( <a href="#">see page 621</a> )	Collection containing one or more bar elements defined for the bar (the collection components are objects of the IRobotBarElement ( <a href="#">see page 646</a> ) type).
	End ( <a href="#">see page 622</a> )	Bar end.
	EndNode ( <a href="#">see page 622</a> )	Number ( <a href="#">see page 30</a> ) of the end bar node.
	Gamma ( <a href="#">see page 622</a> )	Gamma angle value defined for the bar. The gamma angle may be defined both for a bar and for a section. The resultant angle is a sum of both values. The value describes the angle for the bar. .
	InactiveBar ( <a href="#">see page 623</a> )	Flag enabling exclusion of a bar from the structure model - making it "inactive".
	IsSuperBar ( <a href="#">see page 623</a> )	Flag indicating that the bar is a superbar.
	Length ( <a href="#">see page 623</a> )	Bar length.
	Name ( <a href="#">see page 623</a> )	Bar name.
	NameTemplate ( <a href="#">see page 624</a> )	Bar name pattern.
	ReversedOffset ( <a href="#">see page 624</a> )	Flag indicating if offset assigned to a bar is reversed with respect to bar orientation .
	ReversedRelease ( <a href="#">see page 624</a> )	Flag indicating if the member release is reversed in relation to member orientation.
	ReversedSection ( <a href="#">see page 625</a> )	Flag indicating if section assigned to the bar is inverted (section beginning is applied to the bar end).
	ShearForces ( <a href="#">see page 625</a> )	Flag indicating if shearing forces acting on a bar will be included in calculations of structure deformation.
	Start ( <a href="#">see page 625</a> )	Bar beginning .
	StartNode ( <a href="#">see page 626</a> )	Number ( <a href="#">see page 30</a> ) of the initial bar node .
	StructuralType ( <a href="#">see page 626</a> )	Type of structure object.

❖	TensionCompression (see page 626)	Manner in which the bar carries stresses.
❖	TrussBar (see page 627)	Flag indicating if the bar is a truss bar.
❖	UniqueId (see page 627)	Unique identifier of structure component Available since version 1.7.

### II.3.2.2.1 AnalyzeTTMethod

#### C++

```
HRESULT get_AnalyzeTTMethod(VARIANT_BOOL* );
HRESULT put_AnalyzeTTMethod(VARIANT_BOOL);
```

#### C#

```
public bool AnalyzeTTMethod { get; set; }
```

#### Visual Basic

```
Public AnalyzeTTMethod As Boolean
```

#### Description

A flag indicating if the member is taken into consideration in the triangular / trapezoidal method.

#### Version

Available since version 11.

### II.3.2.2.2 Elements

#### C++

```
HRESULT get_Elements(IRobotCollection** );
```

#### C#

```
public IRobotCollection Elements { get; }
```

#### Visual Basic

```
Public ReadOnly Elements As IRobotCollection
```

#### Description

Collection containing one or more bar elements defined for the bar (the collection components are objects of the IRobotBarElement (see page 646) type).

#### Version

Available since version 3.

### II.3.2.2.3 End

#### C++

```
HRESULT get_End(IRobotBarEnd** );
```

#### C#

```
public IRobotBarEnd End { get; }
```

#### Visual Basic

```
Public ReadOnly End As IRobotBarEnd
```

#### Description

Bar end.

#### II.3.2.2.4 EndNode

##### C++

```
HRESULT get_EndNode(long* );
HRESULT put_EndNode(long);
```

##### C#

```
public long EndNode { get; set; }
```

##### Visual Basic

```
Public EndNode As long
```

##### Description

Number (see page 30) of the end bar node.

#### II.3.2.2.5 Gamma

##### C++

```
HRESULT get_Gamma(double* );
HRESULT put_Gamma(double);
```

##### C#

```
public double Gamma { get; set; }
```

##### Visual Basic

```
Public Gamma As double
```

##### Description

Gamma angle value defined for the bar. The gamma angle may be defined both for a bar and for a section. The resultant angle is a sum of both values. The value describes the angle for the bar. .

#### II.3.2.2.6 InactiveBar

##### C++

```
HRESULT get_InactiveBar(VARIANT_BOOL* );
HRESULT put_InactiveBar(VARIANT_BOOL);
```

##### C#

```
public bool InactiveBar { get; set; }
```

##### Visual Basic

```
Public InactiveBar As Boolean
```

##### Description

Flag enabling exclusion of a bar from the structure model - making it "inactive".

##### Version

Available since version 4.5.

#### II.3.2.2.7 IsSuperBar

##### C++

```
HRESULT get_IsSuperBar(VARIANT_BOOL* );
```

##### C#

```
public bool IsSuperBar { get; }
```

**Visual Basic**

```
Public ReadOnly IsSuperBar As Boolean
```

**Description**

Flag indicating that the bar is a superbar.

**Version**

Available since version 3.

### II.3.2.2.8 Length

**C++**

```
HRESULT get_Length(double*);
```

**C#**

```
public double Length { get; }
```

**Visual Basic**

```
Public ReadOnly Length As Double
```

**Description**

Bar length.

**Version**

Available since version 2.5.

### II.3.2.2.9 Name

**C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Description**

Bar name.

**Version**

Available since version 7.5.

### II.3.2.2.10 NameTemplate

**C++**

```
HRESULT get_NameTemplate(BSTR*);  
HRESULT put_NameTemplate(BSTR*);
```

**C#**

```
public String NameTemplate { get; set; }
```

**Visual Basic**

```
Public NameTemplate As String
```

**Description**

Bar name pattern.

**Version**

Available since version 7.5.

**II.3.2.2.11 ReversedOffset****C++**

```
HRESULT get_ReversedOffset(VARIANT_BOOL* );
HRESULT put_ReversedOffset(VARIANT_BOOL);
```

**C#**

```
public bool ReversedOffset { get; set; }
```

**Visual Basic**

```
Public ReversedOffset As Boolean
```

**Description**

Flag indicating if offset assigned to a bar is reversed with respect to bar orientation .

**Version**

Available since version 3.5.

**II.3.2.2.12 ReversedRelease****C++**

```
HRESULT get_ReversedRelease(VARIANT_BOOL* );
HRESULT put_ReversedRelease(VARIANT_BOOL);
```

**C#**

```
public bool ReversedRelease { get; set; }
```

**Visual Basic**

```
Public ReversedRelease As Boolean
```

**Description**

Flag indicating if the member release is reversed in relation to member orientation.

**Version**

Available since version 11.

**II.3.2.2.13 ReversedSection****C++**

```
HRESULT get_ReversedSection(VARIANT_BOOL* );
HRESULT put_ReversedSection(VARIANT_BOOL);
```

**C#**

```
public bool ReversedSection { get; set; }
```

**Visual Basic**

```
Public ReversedSection As Boolean
```

**Description**

Flag indicating if section assigned to the bar is inverted (section beginning is applied to the bar end).

**Version**

Available since version 2.5.

### II.3.2.2.14 ShearForces

#### C++

```
HRESULT get_ShearForces(VARIANT_BOOL* );
HRESULT put_ShearForces(VARIANT_BOOL);
```

#### C#

```
public bool ShearForces { get; set; }
```

#### Visual Basic

```
Public ShearForces As Boolean
```

#### Description

Flag indicating if shearing forces acting on a bar will be included in calculations of structure deformation.

#### Version

Available since version 2.5.

### II.3.2.2.15 Start

#### C++

```
HRESULT get_Start(IRobotBarEnd** );
```

#### C#

```
public IRobotBarEnd Start { get; }
```

#### Visual Basic

```
Public ReadOnly Start As IRobotBarEnd
```

#### Description

Bar beginning .

### II.3.2.2.16 StartNode

#### C++

```
HRESULT get_StartNode(long* );
HRESULT put_StartNode(long);
```

#### C#

```
public long StartNode { get; set; }
```

#### Visual Basic

```
Public StartNode As long
```

#### Description

Number (see page 30) of the initial bar node .

### II.3.2.2.17 StructuralType

#### C++

```
HRESULT get_StructuralType(IRobotObjectStructuralType* );
HRESULT put_StructuralType(IRobotObjectStructuralType);
```

#### C#

```
public IRobotObjectStructuralType StructuralType { get; set; }
```

**Visual Basic**

```
Public StructuralType As IRobotObjectStructuralType
```

**Description**

Type of structure object.

**Version**

Available since version 9.7.

**II.3.2.2.18 TensionCompression****C++**

```
HRESULT get_TensionCompression(IRobotBarTensionCompression* );
HRESULT put_TensionCompression(IRobotBarTensionCompression);
```

**C#**

```
public IRobotBarTensionCompression TensionCompression { get; set; }
```

**Visual Basic**

```
Public TensionCompression As IRobotBarTensionCompression
```

**Description**

Manner in which the bar carries stresses.

**Version**

Available since version 2.5.

**II.3.2.2.19 TrussBar****C++**

```
HRESULT get_TrussBar(VARIANT_BOOL* );
HRESULT put_TrussBar(VARIANT_BOOL);
```

**C#**

```
public bool TrussBar { get; set; }
```

**Visual Basic**

```
Public TrussBar As Boolean
```

**Description**

Flag indicating if the bar is a truss bar.

**Version**

Available since version 2.5.

**II.3.2.2.20 UniqueId****C++**

```
HRESULT get_UniqueId(long* );
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

## Description

Unique identifier of structure component Available since version 1.7.

### II.3.2.3 IRobotBar Methods

The methods of the IRobotBar class are listed here.

#### Public Methods

	Name	Description
💡	CalcGamma (see page 628)	Function calculates a Gamma (see page 622) angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.
💡	ChangeOrientation (see page 628)	Function changes bar orientation to opposite.
💡	GetElemsData (see page 628)	Function fills out the specified object with data of elements belonging to the bar.
💡	GetLCS (see page 629)	Function returns Local Coordinate System for the bar.
💡	GetSimpleBars (see page 629)	Function returns the selection of simple bars which are the superbar components. Prior to activating the function, the IsSuperBar (see page 623) flag should be checked. If the function is activated for a simple bar, it causes a critical error.
💡	SetOffset (see page 629)	Function assigns offset with a specified name to a bar, identically as the SetLabel (see page 32) function. In addition, it enables reversing the offset. .
💡	SetOrientation (see page 630)	Function adjusts bar orientation to the defined sense of the coordinate system axis.
💡	SetSection (see page 630)	Function assigns section of the indicated name to the bar, just as the SetLabel (see page 32) function does. Moreover, it allows inverting the section - applying section end to the bar beginning. .

#### II.3.2.3.1 CalcGamma

##### C++

```
HRESULT CalcGamma(double _local_z_x, double _local_z_y, double _local_z_z, double* ret);
```

##### C#

```
public double CalcGamma(double _local_z_x, double _local_z_y, double _local_z_z);
```

##### Visual Basic

```
Public Function CalcGamma(_local_z_x As double, _local_z_y As double, _local_z_z As double)
As double
```

#### Description

Function calculates a Gamma (see page 622) angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.

#### II.3.2.3.2 ChangeOrientation

##### C++

```
HRESULT ChangeOrientation();
```

##### C#

```
public void ChangeOrientation();
```

**Visual Basic**

```
Public Sub ChangeOrientation()
```

**Description**

Function changes bar orientation to opposite.

**Version**

Available since version 4.5.

**II.3.2.3.3 GetElemsData****C++**

```
HRESULT GetElemsData(IRobotBarElementDataSet* _ret_data);
```

**C#**

```
public void GetElemsData(IRobotBarElementDataSet _ret_data);
```

**Visual Basic**

```
Public Sub GetElemsData(ByRef _ret_data As IRobotBarElementDataSet)
```

**Description**

Function fills out the specified object with data of elements belonging to the bar.

**Version**

Available since version 9.

**II.3.2.3.4 GetLCS****C++**

```
HRESULT GetLCS(IRobotGeoPoint3D* _x, IRobotGeoPoint3D* _y, IRobotGeoPoint3D* _z);
```

**C#**

```
public void GetLCS(IRobotGeoPoint3D _x, IRobotGeoPoint3D _y, IRobotGeoPoint3D _z);
```

**Visual Basic**

```
Public Sub GetLCS(ByRef _x As IRobotGeoPoint3D, ByRef _y As IRobotGeoPoint3D, ByRef _z As IRobotGeoPoint3D)
```

**Description**

Function returns Local Coordinate System for the bar.

**Version**

Available since version 14.7.

**II.3.2.3.5 GetSimpleBars****C++**

```
HRESULT GetSimpleBars(IRobotSelection** ret);
```

**C#**

```
public IRobotSelection GetSimpleBars();
```

**Visual Basic**

```
Public Function GetSimpleBars() As IRobotSelection
```

## Description

Function returns the selection of simple bars which are the superbar components. Prior to activating the function, the IsSuperBar (see page 623) flag should be checked. If the function is activated for a simple bar, it causes a critical error.

## Version

Available since version 3.

### II.3.2.3.6 SetOffset

#### C++

```
HRESULT SetOffset(BSTR _offset_name, VARIANT_BOOL _reverse);
```

#### C#

```
public void SetOffset(String _offset_name, bool _reverse);
```

#### Visual Basic

```
Public Sub SetOffset(_offset_name As String, _reverse As Boolean)
```

## Description

Function assigns offset with a specified name to a bar, identically as the SetLabel (see page 32) function. In addition, it enables reversing the offset.

## Version

Available since version 3.5.

### II.3.2.3.7 SetOrientation

#### C++

```
HRESULT SetOrientation(IRobotGeoCoordinateAxisSense _dir);
```

#### C#

```
public void SetOrientation(IRobotGeoCoordinateAxisSense _dir);
```

#### Visual Basic

```
Public Sub SetOrientation(_dir As IRobotGeoCoordinateAxisSense)
```

## Description

Function adjusts bar orientation to the defined sense of the coordinate system axis.

## Version

Available since version 4.5.

### II.3.2.3.8 SetSection

#### C++

```
HRESULT SetSection(BSTR _section_name, VARIANT_BOOL _reverse);
```

#### C#

```
public void SetSection(String _section_name, bool _reverse);
```

#### Visual Basic

```
Public Sub SetSection(_section_name As String, _reverse As Boolean)
```

## Description

Function assigns section of the indicated name to the bar, just as the SetLabel (see page 32) function does. Moreover, it allows inverting the section - applying section end to the bar beginning. .

## Version

Available since version 2.5.

## II.3.3 IRobotBarServer

### Class Hierarchy

#### C++

```
interface IRobotBarServer : IRobotDataObjectServer;
```

#### C#

```
public interface IRobotBarServer : IRobotDataObjectServer;
```

### Visual Basic

```
Public Interface IRobotBarServer
```

## Description

Server of bars manages the bars defined in the structure .

## II.3.3.1 IRobotBarServer Members

The following tables list the members exposed by IRobotBarServer.

### Public Fields

	Name	Description
◆	FreeNumber (see page 632)	First free user number available for bars Available since version 1.7.
◆	NonlinearHingeModels (see page 633)	Server of non-linear hinge models.
◆	NonlinearHinges (see page 633)	Non-linear hinge server.

### Public Methods

	Name	Description
◆	BeginMultiOperation (see page 634)	Function enables speed-up of generation or modification of a whole group of bars through appropriate optimization of some operations. After completing generation of bars, EndMultiOperation (see page 636)() function should be called up to save changes in a structure.
◆	CalcGamma (see page 635)	Function calculates a Gamma angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.
◆	ChangeOrientation (see page 635)	Function changes orientation of selected bars to opposite.
◆	Create (see page 635)	The function creates between the indicated nodes a bar with the defined number. .
◆	CreateSuperBar (see page 636)	Function generates a superbar with the user-defined number, composed of the indicated simple bars.
◆	Delete (see page 34)	The function deletes the object of the indicated number. .
◆	DeleteMany (see page 34)	The function deletes all objects that meet the criteria of the indicated selection. .
◆	EndMultiOperation (see page 636)	Function updates definitions of bars in a structure. It should be called up only if BeginMultiOperation (see page 634)() function has been called up earlier. .

	Exist ( <a href="#">see page 34</a> )	The function returns True (non-zero value) if the object of the indicated name already exists. .
	FindWithId ( <a href="#">see page 636</a> )	Function returns a number of the bar with a specified unique identifier. If such a bar is not found, then, zero value is returned. Available since version 1.7.
	Get ( <a href="#">see page 34</a> )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .
	GetAll ( <a href="#">see page 35</a> )	The function returns a collection containing all objects managed by the server. .
	GetAnalyzeTTMethodEnabled ( <a href="#">see page 636</a> )	Function returns selection of members taking into consideration in the triangular / trapezoid distribution method.
	GetElemsData ( <a href="#">see page 637</a> )	Function fills out the specified object with data of elements belonging to the indicated bars.
	GetLCS ( <a href="#">see page 637</a> )	Function returns Local Coordinate System for given bar.
	GetMany ( <a href="#">see page 35</a> )	The function returns a collection of objects that meet the criteria of the indicated selection. .
	GetName ( <a href="#">see page 637</a> )	Function returns a name of the specified bar.
	GetNameTemplate ( <a href="#">see page 638</a> )	Function returns a name pattern of the specified bar.
	GetStructuralType ( <a href="#">see page 638</a> )	Function returns a structure object type for the specified bar.
	GetUniqueld ( <a href="#">see page 638</a> )	Function returns a unique identifier for a bar with the user-specified number.
	IsInactive ( <a href="#">see page 639</a> )	Function checking if a given bar is assigned the "inactive" status.
	IsTrussBar ( <a href="#">see page 639</a> )	Function returns a value different from zero (True) if a given bar is a truss bar.
	RemoveLabel ( <a href="#">see page 35</a> )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .
	SetAnalyzeTTMethod ( <a href="#">see page 639</a> )	Function allows for adding/removing members to/from list of members taken into consideration in triangular/trapezoidal distribution method.
	SetInactive ( <a href="#">see page 640</a> )	Funtion setting the "inactive" status for the defined bar list.
	SetLabel ( <a href="#">see page 36</a> )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. .
	SetLabelExt ( <a href="#">see page 640</a> )	This method is an extended version of SetLabel ( <a href="#">see page 36</a> ). It can be used to apply given label directly to selected end of the bar. By default when the optional extra parameter is not specified this method is working the same way as SetLabel ( <a href="#">see page 36</a> ).
	SetNameTemplate ( <a href="#">see page 640</a> )	Function sets a name pattern for the specified bar.
	SetOrientation ( <a href="#">see page 641</a> )	Function adjusts orientation of selected bars to the defined sense of the coordinate system axis. .
	SetShearForces ( <a href="#">see page 641</a> )	For the indicated selection of bars, the function sets flag indicating if shearing forces are taken into account for them. .
	SetStructuralType ( <a href="#">see page 641</a> )	Function sets a structure object type for the specified bar list.
	SetTensionCompression ( <a href="#">see page 642</a> )	Function defines manner of carrying stresses for the specified bar selection. .
	SetTrussBar ( <a href="#">see page 642</a> )	Function enables definition of truss bars.
	Update ( <a href="#">see page 642</a> )	Function updates information on mutual relations between bars and nodes in a strucuture. It should be activated after generation of all structure bars and before definition of bar-node properties. .

### II.3.3.2 IRobotBarServer Fields

The fields of the IRobotBarServer class are listed here.

## Public Fields

	Name	Description
◆	FreeNumber ( <a href="#">see page 632</a> )	First free user number available for bars Available since version 1.7.
◆	NonlinearHingeModels ( <a href="#">see page 633</a> )	Server of non-linear hinge models.
◆	NonlinearHinges ( <a href="#">see page 633</a> )	Non-linear hinge server.

### II.3.3.2.1 FreeNumber

#### C++

```
HRESULT get_FreeNumber(long*);
```

#### C#

```
public long FreeNumber { get; }
```

#### Visual Basic

```
Public ReadOnly FreeNumber As long
```

#### Description

First free user number available for bars Available since version 1.7.

### II.3.3.2.2 NonlinearHingeModels

#### C++

```
HRESULT get_NonlinearHingeModels(IRobotNonlinearHingeModelServer**);
```

#### C#

```
public IRobotNonlinearHingeModelServer NonlinearHingeModels { get; }
```

#### Visual Basic

```
Public ReadOnly NonlinearHingeModels As IRobotNonlinearHingeModelServer
```

#### Description

Server of non-linear hinge models.

#### Version

Available since version 3.

### II.3.3.2.3 NonlinearHinges

#### C++

```
HRESULT get_NonlinearHinges(IRobotNonlinearHingeServer**);
```

#### C#

```
public IRobotNonlinearHingeServer NonlinearHinges { get; }
```

#### Visual Basic

```
Public ReadOnly NonlinearHinges As IRobotNonlinearHingeServer
```

#### Description

Non-linear hinge server.

#### Version

Available since version 3.

### II.3.3.3 IRobotBarServer Methods

The methods of the IRobotBarServer class are listed here.

#### Public Methods

	<b>Name</b>	<b>Description</b>
≡	BeginMultiOperation ( <a href="#">see page 634</a> )	Function enables speed-up of generation or modification of a whole group of bars through appropriate optimization of some operations. After completing generation of bars, EndMultiOperation ( <a href="#">see page 636</a> ()) function should be called up to save changes in a structure.
≡	CalcGamma ( <a href="#">see page 635</a> )	Function calculates a Gamma angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.
≡	ChangeOrientation ( <a href="#">see page 635</a> )	Function changes orientation of selected bars to opposite.
≡	Create ( <a href="#">see page 635</a> )	The function creates between the indicated nodes a bar with the defined number. .
≡	CreateSuperBar ( <a href="#">see page 636</a> )	Function generates a superbar with the user-defined number, composed of the indicated simple bars.
≡	EndMultiOperation ( <a href="#">see page 636</a> )	Function updates definitions of bars in a structure. It should be called up only if BeginMultiOperation ( <a href="#">see page 634</a> ()) function has been called up earlier. .
≡	FindWithId ( <a href="#">see page 636</a> )	Function returns a number of the bar with a specified unique identifier. If such a bar is not found, then, zero value is returned. Available since version 1.7.
≡	GetAnalyzeTTMethodEnabled ( <a href="#">see page 636</a> )	Function returns selection of members taking into consideration in the triangular / trapezoid distribution method.
≡	GetElemsData ( <a href="#">see page 637</a> )	Function fills out the specified object with data of elements belonging to the indicated bars.
≡	GetLCS ( <a href="#">see page 637</a> )	Function returns Local Coordinate System for given bar.
≡	GetName ( <a href="#">see page 637</a> )	Function returns a name of the specified bar.
≡	GetNameTemplate ( <a href="#">see page 638</a> )	Function returns a name pattern of the specified bar.
≡	GetStructuralType ( <a href="#">see page 638</a> )	Function returns a structure object type for the specified bar.
≡	GetUniqueld ( <a href="#">see page 638</a> )	Function returns a unique identifier for a bar with the user-specified number.
≡	IsInactive ( <a href="#">see page 639</a> )	Function checking if a given bar is assigned the "inactive" status.
≡	IsTrussBar ( <a href="#">see page 639</a> )	Function returns a value different from zero (True) if a given bar is a truss bar.
≡	SetAnalyzeTTMethod ( <a href="#">see page 639</a> )	Function allows for adding/removing members to/from list of members taken into consideration in triangular/trapezoidal distribution method.
≡	SetInactive ( <a href="#">see page 640</a> )	Funtion setting the "inactive" status for the defined bar list.
≡	SetLabelExt ( <a href="#">see page 640</a> )	This method is an extended version of SetLabel ( <a href="#">see page 36</a> ). It can be used to apply given label directly to selected end of the bar. By default when the optional extra parameter is not specified this method is working the same way as SetLabel ( <a href="#">see page 36</a> ).
≡	SetNameTemplate ( <a href="#">see page 640</a> )	Function sets a name pattern for the specified bar.
≡	SetOrientation ( <a href="#">see page 641</a> )	Function adjusts orientation of selected bars to the defined sense of the coordinate system axis. .
≡	SetShearForces ( <a href="#">see page 641</a> )	For the indicated selection of bars, the function sets flag indicating if shearing forces are taken into account for them. .
≡	SetStructuralType ( <a href="#">see page 641</a> )	Function sets a structure object type for the specified bar list.
≡	SetTensionCompression ( <a href="#">see page 642</a> )	Function defines manner of carrying stresses for the specified bar selection. .
≡	SetTrussBar ( <a href="#">see page 642</a> )	Function enables definition of truss bars.

	Update (see page 642)	Function updates information on mutual relations between bars and nodes in a strucuture. It should be activated after generation of all structure bars and before definition of bar-node properties. .
-----------------------------------------------------------------------------------	-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### II.3.3.1 BeginMultiOperation

#### C++

```
HRESULT BeginMultiOperation();
```

#### C#

```
public void BeginMultiOperation();
```

#### Visual Basic

```
Public Sub BeginMultiOperation()
```

#### Description

Function enables speed-up of generation or modification of a whole group of bars through appropriate optimization of some operations. After completing generation of bars, EndMultiOperation (see page 636)() function should be called up to save changes in a structure.

#### Version

Available since version 14.7.

### II.3.3.2 CalcGamma

#### C++

```
HRESULT CalcGamma(long _bar_num, double _local_Z_x, double _local_Z_y, double _local_Z_z,  
double* ret);
```

#### C#

```
public double CalcGamma(long _bar_num, double _local_Z_x, double _local_Z_y, double  
_local_Z_z);
```

#### Visual Basic

```
Public Function CalcGamma(_bar_num As long, _local_Z_x As double, _local_Z_y As double,  
_local_Z_z As double) As double
```

#### Description

Function calculates a Gamma angle value for a bar based on the coordinates (in the global system) of the local Z direction vector.

### II.3.3.3 ChangeOrientation

#### C++

```
HRESULT ChangeOrientation(BSTR _bars);
```

#### C#

```
public void ChangeOrientation(String _bars);
```

#### Visual Basic

```
Public Sub ChangeOrientation(_bars As String)
```

#### Description

Function changes orientation of selected bars to opposite.

**Version**

Available since version 4.5.

**II.3.3.4 Create****C++**

```
HRESULT Create(long _bar_num, long _start_node, long _end_node);
```

**C#**

```
public void Create(long _bar_num, long _start_node, long _end_node);
```

**Visual Basic**

```
Public Sub Create(_bar_num As long, _start_node As long, _end_node As long)
```

**Description**

The function creates between the indicated nodes a bar with the defined number. .

**II.3.3.5 CreateSuperBar****C++**

```
HRESULT CreateSuperBar(long _number, BSTR _simple_bars);
```

**C#**

```
public void CreateSuperBar(long _number, String _simple_bars);
```

**Visual Basic**

```
Public Sub CreateSuperBar(_number As long, _simple_bars As String)
```

**Description**

Function generates a superbar with the user-defined number, composed of the indicated simple bars.

**Version**

Available since version 3.

**II.3.3.6 EndMultiOperation****C++**

```
HRESULT EndMultiOperation();
```

**C#**

```
public void EndMultiOperation();
```

**Visual Basic**

```
Public Sub EndMultiOperation()
```

**Description**

Function updates definitions of bars in a structure. It should be called up only if BeginMultiOperation ( see page 634)() function has been called up earlier. .

**Version**

Available since version 14.7.

### II.3.3.7 FindWithId

**C++**

```
HRESULT FindWithId(long _unique_id, long* ret);
```

**C#**

```
public long FindWithId(long _unique_id);
```

**Visual Basic**

```
Public Function FindWithId(_unique_id As long) As long
```

**Description**

Function returns a number of the bar with a specified unique identifier. If such a bar is not found, then, zero value is returned.  
Available since version 1.7.

### II.3.3.8 GetAnalyzeTTMethodEnabled

**C++**

```
HRESULT GetAnalyzeTTMethodEnabled(IRobotSelection** ret);
```

**C#**

```
public IRobotSelection GetAnalyzeTTMethodEnabled();
```

**Visual Basic**

```
Public Function GetAnalyzeTTMethodEnabled() As IRobotSelection
```

**Description**

Function returns selection of members taking into consideration in the triangular / trapezoid distribution method.

**Version**

Available since version 11.

### II.3.3.9 GetElemsData

**C++**

```
HRESULT GetElemsData(BSTR _bar_sel_txt, IRobotBarElementDataSet* _ret_data);
```

**C#**

```
public void GetElemsData(String _bar_sel_txt, IRobotBarElementDataSet _ret_data);
```

**Visual Basic**

```
Public Sub GetElemsData(_bar_sel_txt As String, ByRef _ret_data As IRobotBarElementDataSet)
```

**Description**

Function fills out the specified object with data of elements belonging to the indicated bars.

**Version**

Available since version 9.

### II.3.3.10 GetLCS

**C++**

```
HRESULT GetLCS(long _bar_num, IRobotGeoPoint3D* _x, IRobotGeoPoint3D* _y, IRobotGeoPoint3D* _z);
```

**C#**

```
public void GetLCS(long _bar_num, IRobotGeoPoint3D _x, IRobotGeoPoint3D _y,
IRobotGeoPoint3D _z);
```

**Visual Basic**

```
Public Sub GetLCS(_bar_num As long, ByRef _x As IRobotGeoPoint3D, ByRef _y As
IRobotGeoPoint3D, ByRef _z As IRobotGeoPoint3D)
```

**Description**

Function returns Local Coordinate System for given bar.

**Version**

Available since version 14.7.

**II.3.3.11 GetName****C++**

```
HRESULT GetName(long _bar_num, BSTR* ret);
```

**C#**

```
public String GetName(long _bar_num);
```

**Visual Basic**

```
Public Function GetName(_bar_num As long) As String
```

**Description**

Function returns a name of the specified bar.

**Version**

Available since version 7.5.

**II.3.3.12 GetNameTemplate****C++**

```
HRESULT GetNameTemplate(long _bar_num, BSTR* ret);
```

**C#**

```
public String GetNameTemplate(long _bar_num);
```

**Visual Basic**

```
Public Function GetNameTemplate(_bar_num As long) As String
```

**Description**

Function returns a name pattern of the specified bar.

**Version**

Available since version 7.5.

**II.3.3.13 GetStructuralType****C++**

```
HRESULT GetStructuralType(long _bar, IRobotObjectStructuralType* ret);
```

**C#**

```
public IRobotObjectStructuralType GetStructuralType(long _bar);
```

**Visual Basic**

```
Public Function GetStructuralType(_bar As long) As IRobotObjectStructuralType
```

**Description**

Function returns a structure object type for the specified bar.

**Version**

Available since version 9.7.

**II.3.3.3.14 GetUniqueId****C++**

```
HRESULT GetUniqueId(long _bar_num, long* ret);
```

**C#**

```
public long GetUniqueId(long _bar_num);
```

**Visual Basic**

```
Public Function GetUniqueId(_bar_num As long) As long
```

**Description**

Function returns a unique identifier for a bar with the user-specified number.

**Version**

Available since version 8.2.

**II.3.3.3.15 IsInactive****C++**

```
HRESULT IsInactive(long _bar_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsInactive(long _bar_num);
```

**Visual Basic**

```
Public Function IsInactive(_bar_num As long) As Boolean
```

**Description**

Function checking if a given bar is assigned the "inactive" status.

**Version**

Available since version 4.5.

**II.3.3.3.16 IsTrussBar****C++**

```
HRESULT IsTrussBar(long _bar_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsTrussBar(long _bar_num);
```

**Visual Basic**

```
Public Function IsTrussBar(_bar_num As long) As Boolean
```

## Description

Function returns a value different from zero (True) if a given bar is a truss bar.

### II.3.3.17 SetAnalyzeTTMethod

#### C++

```
HRESULT SetAnalyzeTTMethod(BSTR _bars, VARIANT_BOOL _analyze);
```

#### C#

```
public void SetAnalyzeTTMethod(String _bars, bool _analyze);
```

#### Visual Basic

```
Public Sub SetAnalyzeTTMethod(_bars As String, _analyze As Boolean)
```

#### Description

Function allows for adding/removing members to/from list of members taken into consideration in triangular/trapezoidal distribution method.

#### Version

Available since version 11.

### II.3.3.18 SetInactive

#### C++

```
HRESULT SetInactive(BSTR _bars, VARIANT_BOOL _inactive = true);
```

#### C#

```
public void SetInactive(String _bars, bool _inactive = true);
```

#### Visual Basic

```
Public Sub SetInactive(_bars As String, Optional _inactive As Boolean = true)
```

#### Description

Funtion setting the "inactive" status for the defined bar list.

#### Version

Available since version 4.5.

### II.3.3.19 SetLabelExt

#### C++

```
HRESULT SetLabelExt(IRobotSelection* _bars, IRobotLabelType _lab_type, BSTR _lab_name, long _ext_param = -1);
```

#### C#

```
public void SetLabelExt(IRobotSelection _bars, IRobotLabelType _lab_type, String _lab_name, long _ext_param = -1);
```

#### Visual Basic

```
Public Sub SetLabelExt(ByRef _bars As IRobotSelection, _lab_type As IRobotLabelType, _lab_name As String, Optional _ext_param As long = -1)
```

#### Description

This method is an extended version of SetLabel (see page 36). It can be used to apply given label directly to selected end of the bar. By default when the optional extra parameter is not specified this method is working the same way as SetLabel (see page 36).

see page 36).

#### Version

Available since version 15.2.

### II.3.3.20 SetNameTemplate

#### C++

```
HRESULT SetNameTemplate(long _bar_num, BSTR _name_tmpl);
```

#### C#

```
public void SetNameTemplate(long _bar_num, String _name_tmpl);
```

#### Visual Basic

```
Public Sub SetNameTemplate(_bar_num As long, _name_tmpl As String)
```

#### Description

Function sets a name pattern for the specified bar.

#### Version

Available since version 7.5.

### II.3.3.21 SetOrientation

#### C++

```
HRESULT SetOrientation(BSTR _bars, IRobotGeoCoordinateAxisSense _dir);
```

#### C#

```
public void SetOrientation(String _bars, IRobotGeoCoordinateAxisSense _dir);
```

#### Visual Basic

```
Public Sub SetOrientation(_bars As String, _dir As IRobotGeoCoordinateAxisSense)
```

#### Description

Function adjusts orientation of selected bars to the defined sense of the coordinate system axis. .

#### Version

Available since version 4.5.

### II.3.3.22 SetShearForces

#### C++

```
HRESULT SetShearForces(IRobotSelection* _bar_sel, VARIANT_BOOL _set = -1);
```

#### C#

```
public void SetShearForces(IRobotSelection _bar_sel, bool _set = -1);
```

#### Visual Basic

```
Public Sub SetShearForces(ByRef _bar_sel As IRobotSelection, Optional _set As Boolean = -1)
```

#### Description

For the indicated selection of bars, the function sets flag indicating if shearing forces are taken into account for them. .

#### Version

Available since version 2.5.

### II.3.3.23 SetStructuralType

#### C++

```
HRESULT SetStructuralType(BSTR _bar_sel, IRobotObjectStructuralType _str_type);
```

#### C#

```
public void SetStructuralType(String _bar_sel, IRobotObjectStructuralType _str_type);
```

#### Visual Basic

```
Public Sub SetStructuralType(_bar_sel As String, _str_type As IRobotObjectStructuralType)
```

#### Description

Function sets a structure object type for the specified bar list.

#### Version

Available since version 9.7.

### II.3.3.24 SetTensionCompression

#### C++

```
HRESULT SetTensionCompression(IRobotSelection* _bar_sel, IRobotBarTensionCompression _tc);
```

#### C#

```
public void SetTensionCompression(IRobotSelection _bar_sel, IRobotBarTensionCompression _tc);
```

#### Visual Basic

```
Public Sub SetTensionCompression(ByRef _bar_sel As IRobotSelection, _tc As IRobotBarTensionCompression)
```

#### Description

Function defines manner of carrying stresses for the specified bar selection. .

#### Version

Available since version 2.5.

### II.3.3.25 SetTrussBar

#### C++

```
HRESULT SetTrussBar(BSTR _bars, VARIANT_BOOL _truss_bar);
```

#### C#

```
public void SetTrussBar(String _bars, bool _truss_bar);
```

#### Visual Basic

```
Public Sub SetTrussBar(_bars As String, _truss_bar As Boolean)
```

#### Description

Function enables definition of truss bars.

### II.3.3.26 Update

#### C++

```
HRESULT Update();
```

**C#**

```
public void Update();
```

**Visual Basic**

```
Public Sub Update()
```

**Description**

Function updates information on mutual relations between bars and nodes in a structure. It should be activated after generation of all structure bars and before definition of bar-node properties. .

**Version**

Available since version 3.5.

**II.3.4 IRobotBarEnd****Class Hierarchy****C++**

```
interface IRobotBarEnd : IDispatch;
```

**C#**

```
public interface IRobotBarEnd;
```

**Visual Basic**

```
Public Interface IRobotBarEnd
```

**Description**

One can indicate a beginning and an end of each bar. The object of RobotBarEnd type, describing one of the bar ends, is created to make it possible to carry out operations affecting only the beginning or end of a bar (and not the entire bar).

**II.3.4.1 IRobotBarEnd Members**

The following tables list the members exposed by IRobotBarEnd.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Node ( <a href="#">see page 643</a> )	Node number (user-defined).

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	GetLabel ( <a href="#">see page 644</a> )	The function returns the label of the given type defined for the object.
◆	GetLabelName ( <a href="#">see page 644</a> )	The function returns the label of the given type, applied to the object.
◆	GetLabels ( <a href="#">see page 644</a> )	The function returns a collection containing all labels applied to the object. .
◆	GetOffsetValue ( <a href="#">see page 645</a> )	Function calculates and returns total values of offset at a bar end. .
◆	HasLabel ( <a href="#">see page 645</a> )	The function returns a non-zero value (True) if a label of the given type has been defined for the object.
◆	RemoveLabel ( <a href="#">see page 645</a> )	The function removes the label of the given type from the object. .
◆	SetLabel ( <a href="#">see page 646</a> )	The function applies a label of a given type to the object. .

**II.3.4.2 IRobotBarEnd Fields**

The fields of the IRobotBarEnd class are listed here.

## Public Fields

	Name	Description
◆	Node (see page 643)	Node number (user-defined).

### II.3.4.2.1 Node

#### C++

```
HRESULT get_Node(long* );
HRESULT put_Node(long);
```

#### C#

```
public long Node { get; set; }
```

#### Visual Basic

```
Public Node As long
```

#### Description

Node number (user-defined).

### II.3.4.3 IRobotBarEnd Methods

The methods of the IRobotBarEnd class are listed here.

#### Public Methods

	Name	Description
◆	GetLabel (see page 644)	The function returns the label of the given type defined for the object.
◆	GetLabelName (see page 644)	The function returns the label of the given type, applied to the object.
◆	GetLabels (see page 644)	The function returns a collection containing all labels applied to the object. .
◆	GetOffsetValue (see page 645)	Function calculates and returns total values of offset at a bar end. .
◆	HasLabel (see page 645)	The function returns a non-zero value (True) if a label of the given type has been defined for the object.
◆	RemoveLabel (see page 645)	The function removes the label of the given type from the object. .
◆	SetLabel (see page 646)	The function applies a label of a given type to the object. .

### II.3.4.3.1 GetLabel

#### C++

```
HRESULT GetLabel(IRobotLabelType _lab_type, IRobotLabel** ret);
```

#### C#

```
public IRobotLabel GetLabel(IRobotLabelType _lab_type);
```

#### Visual Basic

```
Public Function GetLabel(_lab_type As IRobotLabelType) As IRobotLabel
```

#### Description

The function returns the label of the given type defined for the object.

### II.3.4.3.2 GetLabelName

#### C++

```
HRESULT GetLabelName(IRobotLabelType _lab_type, BSTR* ret);
```

**C#**

```
public String GetLabelName(IRobotLabelType _lab_type);
```

**Visual Basic**

```
Public Function GetLabelName(_lab_type As IRobotLabelType) As String
```

**Description**

The function returns the label of the given type, applied to the object.

**II.3.4.3.3 GetLabels****C++**

```
HRESULT GetLabels(IRobotCollection** ret);
```

**C#**

```
public IRobotCollection GetLabels();
```

**Visual Basic**

```
Public Function GetLabels() As IRobotCollection
```

**Description**

The function returns a collection containing all labels applied to the object. .

**II.3.4.3.4 GetOffsetValue****C++**

```
HRESULT GetOffsetValue(IRobotBarEndOffsetData** ret);
```

**C#**

```
public IRobotBarEndOffsetData GetOffsetValue();
```

**Visual Basic**

```
Public Function GetOffsetValue() As IRobotBarEndOffsetData
```

**Description**

Function calculates and returns total values of offset at a bar end. .

**Version**

Available since version 3.5.

**II.3.4.3.5 HasLabel****C++**

```
HRESULT HasLabel(IRobotLabelType _lab_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool HasLabel(IRobotLabelType _lab_type);
```

**Visual Basic**

```
Public Function HasLabel(_lab_type As IRobotLabelType) As Boolean
```

**Description**

The function returns a non-zero value (True) if a label of the given type has been defined for the object.

### II.3.4.3.6 RemoveLabel

**C++**

```
HRESULT RemoveLabel(IRobotLabelType _lab_type);
```

**C#**

```
public void RemoveLabel(IRobotLabelType _lab_type);
```

**Visual Basic**

```
Public Sub RemoveLabel(_lab_type As IRobotLabelType)
```

**Description**

The function removes the label of the given type from the object. .

### II.3.4.3.7 SetLabel

**C++**

```
HRESULT SetLabel(IRobotLabelType _lab_type, BSTR _lab_name);
```

**C#**

```
public void SetLabel(IRobotLabelType _lab_type, String _lab_name);
```

**Visual Basic**

```
Public Sub SetLabel(_lab_type As IRobotLabelType, _lab_name As String)
```

**Description**

The function applies a label of a given type to the object. .

## II.3.5 IRobotBarTensionCompression

**C++**

```
enum IRobotBarTensionCompression;
```

**C#**

```
public enum IRobotBarTensionCompression;
```

**Visual Basic**

```
Public Enum IRobotBarTensionCompression
```

**Members**

Members	Description
I_BTC_STANDARD = 0	The bar carries both compressive and tensile stresses . Available since version 2.5.
I_BTC_TENSION_ONLY = 1	The bar carries only tensile stresses . Available since version 2.5.
I_BTC_COMPRESSION_ONLY = 2	The bar carries only compressive stresses. Available since version 2.5.

**Version**

Available since version 2.5.

## II.3.6 IRobotBarElement

### Class Hierarchy

#### C++

```
interface IRobotBarElement : IDispatch;
```

#### C#

```
public interface IRobotBarElement;
```

### Visual Basic

```
Public Interface IRobotBarElement
```

### Description

Bar element.

### Version

Available since version 3.

## II.3.6.1 IRobotBarElement Members

The following tables list the members exposed by IRobotBarElement.

### Public Fields

	Name	Description
◆	EndNode (see page 647)	Number (see page 648) of end node.
◆	Inactive (see page 647)	Flag enabling exclusion of a bar element from the structure calculation model.
◆	IsCalc (see page 648)	Flag indicating if the element has been created automatically during generation of the structure calculation model.
◆	Number (see page 648)	
◆	StartNode (see page 648)	Number (see page 648) of origin node.

## II.3.6.2 IRobotBarElement Fields

The fields of the IRobotBarElement class are listed here.

### Public Fields

	Name	Description
◆	EndNode (see page 647)	Number (see page 648) of end node.
◆	Inactive (see page 647)	Flag enabling exclusion of a bar element from the structure calculation model.
◆	IsCalc (see page 648)	Flag indicating if the element has been created automatically during generation of the structure calculation model.
◆	Number (see page 648)	
◆	StartNode (see page 648)	Number (see page 648) of origin node.

## II.3.6.2.1 EndNode

#### C++

```
HRESULT get_EndNode( long* );
```

#### C#

```
public long EndNode { get; }
```

**Visual Basic**

```
Public ReadOnly EndNode As long
```

**Description**

Number (see page 648) of end node.

**Version**

Available since version 3.

**II.3.6.2.2 Inactive****C++**

```
HRESULT get_Inactive(VARIANT_BOOL* );
HRESULT put_Inactive(VARIANT_BOOL);
```

**C#**

```
public bool Inactive { get; set; }
```

**Visual Basic**

```
Public Inactive As Boolean
```

**Description**

Flag enabling exclusion of a bar element from the structure calculation model.

**Version**

Available since version 4.5.

**II.3.6.2.3 IsCalc****C++**

```
HRESULT get_IsCalc(VARIANT_BOOL* );
```

**C#**

```
public bool IsCalc { get; }
```

**Visual Basic**

```
Public ReadOnly IsCalc As Boolean
```

**Description**

Flag indicating if the element has been created automatically during generation of the structure calculation model.

**Version**

Available since version 3.

**II.3.6.2.4 Number****C++**

```
HRESULT get_Number(long* );
```

**C#**

```
public long Number { get; }
```

**Visual Basic**

```
Public ReadOnly Number As long
```

**Version**

Available since version 3.

### II.3.6.2.5 StartNode

#### C++

```
HRESULT get_StartNode(long*);
```

#### C#

```
public long StartNode { get; }
```

#### Visual Basic

```
Public ReadOnly StartNode As long
```

#### Description

Number (see page 648) of origin node.

#### Version

Available since version 3.

### II.3.7 IRobotBarElementData

#### Class Hierarchy

#### C++

```
interface IRobotBarElementData : IDispatch;
```

#### C#

```
public interface IRobotBarElementData;
```

#### Visual Basic

```
Public Interface IRobotBarElementData
```

#### Description

Bar (see page 649) element data.

#### Version

Available since version 9.

### II.3.7.1 IRobotBarElementData Members

The following tables list the members exposed by IRobotBarElementData.

#### Public Fields

	Name	Description
◆	Bar (see page 649)	Number (see page 650) of a bar that the element belongs to.
◆	Number (see page 650)	Element no.

#### Public Methods

	Name	Description
◆	GetEndNode (see page 650)	Function returns a number and coordinates of the end node of the element.
◆	GetStartNode (see page 651)	Function returns a number and coordinates of the beginning node of the element.

### II.3.7.2 IRobotBarElementData Fields

The fields of the IRobotBarElementData class are listed here.

## Public Fields

	Name	Description
◆	Bar (see page 649)	Number (see page 650) of a bar that the element belongs to.
◆	Number (see page 650)	Element no.

### II.3.7.2.1 Bar

#### C++

```
HRESULT get_Bar(long*);
```

#### C#

```
public long Bar { get; }
```

#### Visual Basic

```
Public ReadOnly Bar As long
```

#### Description

Number (see page 650) of a bar that the element belongs to.

#### Version

Available since version 9.

### II.3.7.2.2 Number

#### C++

```
HRESULT get_Number(long*);
```

#### C#

```
public long Number { get; }
```

#### Visual Basic

```
Public ReadOnly Number As long
```

#### Description

Element no.

#### Version

Available since version 9.

### II.3.7.3 IRobotBarElementData Methods

The methods of the IRobotBarElementData class are listed here.

#### Public Methods

	Name	Description
◆	GetEndNode (see page 650)	Function returns a number and coordinates of the end node of the element.
◆	GetStartNode (see page 651)	Function returns a number and coordinates of the beginning node of the element.

### II.3.7.3.1 GetEndNode

#### C++

```
HRESULT GetEndNode(double* _x, double* _y, double* _z, long* ret);
```

**C#**

```
public long GetEndNode(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Function GetEndNode(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As long
```

**Description**

Function returns a number and coordinates of the end node of the element.

**Version**

Available since version 9.

**II.3.7.3.2 GetStartNode****C++**

```
HRESULT GetStartNode(double* _x, double* _y, double* _z, long* ret);
```

**C#**

```
public long GetStartNode(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Function GetStartNode(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As long
```

**Description**

Function returns a number and coordinates of the beginning node of the element.

**Version**

Available since version 9.

**II.3.8 IRobotBarElementDataSet****Class Hierarchy****C++**

```
interface IRobotBarElementDataSet : IDispatch;
```

**C#**

```
public interface IRobotBarElementDataSet;
```

**Visual Basic**

```
Public Interface IRobotBarElementDataSet
```

**Description**

Data set for bar elements.

**Version**

Available since version 9.

**II.3.8.1 IRobotBarElementDataSet Members**

The following tables list the members exposed by IRobotBarElementDataSet.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BarCount (see page 652)	Number of bars whose elements are included in the set.
◆	ElemCount (see page 652)	Number of elements in the set.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	GetBarNumber (see page 653)	Function returns a user number for the bar with the specified index.
◆	GetElem (see page 653)	Function gives access to data of the element with the specified index.
◆	GetElemCountForBar (see page 653)	Function returns a number of elements for the bar with the specified index.
◆	GetElemForBar (see page 654)	Function returns data of the determined element for the specified bar.

**II.3.8.2 IRobotBarElementDataSet Fields**

The fields of the IRobotBarElementDataSet class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BarCount (see page 652)	Number of bars whose elements are included in the set.
◆	ElemCount (see page 652)	Number of elements in the set.

**II.3.8.2.1 BarCount****C++**

```
HRESULT get_BarCount(long*);
```

**C#**

```
public long BarCount { get; }
```

**Visual Basic**

```
Public ReadOnly BarCount As long
```

**Description**

Number of bars whose elements are included in the set.

**Version**

Available since version 9.

**II.3.8.2.2 ElemCount****C++**

```
HRESULT get_ElemCount(long*);
```

**C#**

```
public long ElemCount { get; }
```

**Visual Basic**

```
Public ReadOnly ElemCount As long
```

**Description**

Number of elements in the set.

**Version**

Available since version 9.

### II.3.8.3 IRobotBarElementDataSet Methods

The methods of the IRobotBarElementDataSet class are listed here.

#### Public Methods

	Name	Description
ESH	GetBarNumber (see page 653)	Function returns a user number for the bar with the specified index.
ESH	GetElem (see page 653)	Function gives access to data of the element with the specified index.
ESH	GetElemCountForBar (see page 653)	Function returns a number of elements for the bar with the specified index.
ESH	GetElemForBar (see page 654)	Function returns data of the determined element for the specified bar.

#### II.3.8.3.1 GetBarNumber

##### C++

```
HRESULT GetBarNumber(long _bar_idx, long* ret);
```

##### C#

```
public long GetBarNumber(long _bar_idx);
```

##### Visual Basic

```
Public Function GetBarNumber(_bar_idx As long) As long
```

##### Description

Function returns a user number for the bar with the specified index.

##### Version

Available since version 9.

#### II.3.8.3.2 GetElem

##### C++

```
HRESULT GetElem(long _elem_idx, IRobotBarElementData* _ret_data, long* ret);
```

##### C#

```
public long GetElem(long _elem_idx, IRobotBarElementData _ret_data);
```

##### Visual Basic

```
Public Function GetElem(_elem_idx As long, ByRef _ret_data As IRobotBarElementData) As long
```

##### Description

Function gives access to data of the element with the specified index.

##### Version

Available since version 9.

#### II.3.8.3.3 GetElemCountForBar

##### C++

```
HRESULT GetElemCountForBar(long _bar_idx, long* ret);
```

##### C#

```
public long GetElemCountForBar(long _bar_idx);
```

**Visual Basic**

```
Public Function GetElemCountForBar(_bar_idx As long) As long
```

**Description**

Function returns a number of elements for the bar with the specified index.

**Version**

Available since version 9.

**II.3.8.3.4 GetElemForBar****C++**

```
HRESULT GetElemForBar(long _bar_idx, long _elem_idx, IRobotBarElementData* _ret_data, long* ret);
```

**C#**

```
public long GetElemForBar(long _bar_idx, long _elem_idx, IRobotBarElementData _ret_data);
```

**Visual Basic**

```
Public Function GetElemForBar(_bar_idx As long, _elem_idx As long, ByRef _ret_data As IRobotBarElementData) As long
```

**Description**

Function returns data of the determined element for the specified bar.

**Version**

Available since version 9.

**II.4 Complex attributes shared by objects of different types****II.4.1 Material****Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotMaterialType ( <a href="#">see page 667</a> )	Types of materials accepted by Robot.. .
	IRobotMaterialTimberType ( <a href="#">see page 667</a> )	Types of timber materials defined by Robot. .
	IRobotMaterialModel ( <a href="#">see page 667</a> )	Available material models.
	IRobotMaterialElasticType ( <a href="#">see page 668</a> )	Available models of material elasticity.
	IRobotMaterialElasticUnloadingMethod ( <a href="#">see page 670</a> )	Available methods of material unloading.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotMaterialData ( <a href="#">see page 654</a> )	Material parameters are described by means of the RobotMaterialdata interface. An object of the type contains data of each label (complex attribute) of the I_LT_BAR_MATERIAL type .
	IRobotMaterialElasticModel ( <a href="#">see page 668</a> )	Model ( <a href="#">see page 669</a> ) of material elasticity.. .

## II.4.1.1 IRobotMaterialData

### Class Hierarchy

#### C++

```
interface IRobotMaterialData : IDispatch;
```

#### C#

```
public interface IRobotMaterialData;
```

### Visual Basic

```
Public Interface IRobotMaterialData
```

### Description

Material parameters are described by means of the RobotMaterialdata interface. An object of the type contains data of each label (complex attribute) of the I\_LT\_BAR\_MATERIAL type .

## II.4.1.1.1 IRobotMaterialData Members

The following tables list the members exposed by IRobotMaterialData.

### Public Fields

	Name	Description
◆	CB71_Category ( [ see page 657 ] )	Additional characteristics for CD71 code - 3 categories (I, II, II).
◆	CB71_Humidity ( [ see page 657 ] )	Additional characteristic for CB71 code - humidity in %.
◆	CB71_Nature ( [ see page 658 ] )	Additional characteristics for CD71 code - 2 natures (resinous / non-resinous).
◆	CB71_Retreat ( [ see page 658 ] )	Additional characteristic for CB71 code - retreat in %.
◆	CS ( [ see page 658 ] )	Reduction coefficient for shear limit stress (steel, aluminium) (not implemented).
◆	Default ( [ see page 658 ] )	Default material (1 - yes, 0 - no).
◆	DumpCoef ( [ see page 659 ] )	Dumping coefficient .
◆	E ( [ see page 659 ] )	Young's modulus / axial Young modulus for timber.
◆	E_5 ( [ see page 659 ] )	Young's 5% module (timber) only for EC5 code.
◆	E_Trans ( [ see page 659 ] )	Young transverse module (timber) only for EC5 code.
◆	EC_Deformation ( [ see page 660 ] )	Shear deformation module.
◆	GMean ( [ see page 660 ] )	
◆	Kirchoff ( [ see page 660 ] )	Shear (transversal) module.
◆	LX ( [ see page 660 ] )	Thermal expansion coefficient.
◆	Name ( [ see page 661 ] )	Material name.
◆	NU ( [ see page 661 ] )	Poisson ratio .
◆	Nuance ( [ see page 661 ] )	Additional short material description.
◆	PN_Deformation ( [ see page 661 ] )	Shear deformation module for Polish code.
◆	PN_E_Additional ( [ see page 662 ] )	Additional Young module (timber) for Polish code (not implemented).
◆	PN_E_Trans ( [ see page 662 ] )	Young transverse module (timber) for Polish code (not implemented).
◆	RE ( [ see page 662 ] )	Yield point (steel, aluminum), compression resistance (concrete).
◆	RE_AxCompr ( [ see page 662 ] )	Axial compression resistance (timber) (not implemented).
◆	RE_AxTens ( [ see page 663 ] )	Axial tension resistance ( timber).
◆	RE_Bending ( [ see page 663 ] )	Bending resistance ( timber).
◆	RE_Shear ( [ see page 663 ] )	Shear resistance ( timber).
◆	RE_TrCompr ( [ see page 664 ] )	Transverse compression resistance ( timber).
◆	RE_TrTens ( [ see page 664 ] )	Transverse tension resistance (timber).

◆	RO ( <a href="#">see page 664</a> )	Material density (mass).
◆	RT ( <a href="#">see page 664</a> )	Tension resistance (steel).
◆	SecondName ( <a href="#">see page 665</a> )	Second name of the material, available only during editing material database under MS Access.
◆	Steel_Thermal ( <a href="#">see page 665</a> )	Thermal flag for steel materials. When it is on, it refers to special processing during production..
◆	Timber_Type ( <a href="#">see page 665</a> )	Timber material type (glued multi-layered or normal) .
◆	Type ( <a href="#">see page 665</a> )	Material type.

## Public Methods

	Name	Description
◆	LoadFromDBase ( <a href="#">see page 666</a> )	The function allows one to load the data on a material with the defined name to the material database. The returned logical value (True, False) indicates whether the information on the material with the defined name has been found in the material database or not. .
◆	SaveToDBase ( <a href="#">see page 666</a> )	Function saves material to the user database. .

### II.4.1.1.2 IRobotMaterialData Fields

The fields of the IRobotMaterialData class are listed here.

## Public Fields

	Name	Description
◆	CB71_Category ( <a href="#">see page 657</a> )	Additional characteristics for CD71 code - 3 categories (I, II, II).
◆	CB71_Humidity ( <a href="#">see page 657</a> )	Additional characteristic for CB71 code - humidity in %.
◆	CB71_Nature ( <a href="#">see page 658</a> )	Additional characteristics for CD71 code - 2 natures (resinous / non-resinous).
◆	CB71_Retreat ( <a href="#">see page 658</a> )	Additional characteristic for CB71 code - retreat in %.
◆	CS ( <a href="#">see page 658</a> )	Reduction coefficient for shear limit stress (steel, aluminium) (not implemented).
◆	Default ( <a href="#">see page 658</a> )	Default material (1 - yes, 0 - no).
◆	DumpCoef ( <a href="#">see page 659</a> )	Dumping coefficient .
◆	E ( <a href="#">see page 659</a> )	Young's modulus / axial Young modulus for timber.
◆	E_5 ( <a href="#">see page 659</a> )	Young's 5% module (timber) only for EC5 code.
◆	E_Trans ( <a href="#">see page 659</a> )	Young transverse module (timber) only for EC5 code.
◆	EC_Deformation ( <a href="#">see page 660</a> )	Shear deformation module.
◆	GMean ( <a href="#">see page 660</a> )	
◆	Kirchoff ( <a href="#">see page 660</a> )	Shear (transversal) module.
◆	LX ( <a href="#">see page 660</a> )	Thermal expansion coefficient.
◆	Name ( <a href="#">see page 661</a> )	Material name.
◆	NU ( <a href="#">see page 661</a> )	Poisson ratio .
◆	Nuance ( <a href="#">see page 661</a> )	Additional short material description.
◆	PN_Deformation ( <a href="#">see page 661</a> )	Shear deformation module for Polish code.
◆	PN_E_Additional ( <a href="#">see page 662</a> )	Additional Young module (timber) for Polish code (not implemented).
◆	PN_E_Trans ( <a href="#">see page 662</a> )	Young transverse module (timber) for Polish code (not implemented).
◆	RE ( <a href="#">see page 662</a> )	Yield point (steel, aluminum), compression resistance (concrete).
◆	RE_AxCompr ( <a href="#">see page 662</a> )	Axial compression resistance (timber) (not implemented).
◆	RE_AxTens ( <a href="#">see page 663</a> )	Axial tension resistance ( timber).
◆	RE_Bending ( <a href="#">see page 663</a> )	Bending resistance ( timber).
◆	RE_Shear ( <a href="#">see page 663</a> )	Shear resistance ( timber).
◆	RE_TrCompr ( <a href="#">see page 664</a> )	Transverse compression resistance ( timber).
◆	RE_TrTens ( <a href="#">see page 664</a> )	Transverse tension resistance (timber).

❖	RO (see page 664)	Material density (mass).
❖	RT (see page 664)	Tension resistance (steel).
❖	SecondName (see page 665)	Second name of the material, available only during editing material database under MS Access.
❖	Steel_Thermal (see page 665)	Thermal flag for steel materials. When it is on, it refers to special processing during production..
❖	Timber_Type (see page 665)	Timber material type (glued multi-layered or normal) .
❖	Type (see page 665)	Material type.

#### II.4.1.1.2.1 CB71\_Category

**C++**

```
HRESULT get_CB71_Category(int* );
HRESULT put_CB71_Category(int);
```

**C#**

```
public int CB71_Category { get; set; }
```

**Visual Basic**

```
Public CB71_Category As int
```

**Description**

Additional characteristics for CD71 code - 3 categories (I, II, III).

#### II.4.1.1.2.2 CB71\_Humidity

**C++**

```
HRESULT get_CB71_Humidity(double* );
HRESULT put_CB71_Humidity(double);
```

**C#**

```
public double CB71_Humidity { get; set; }
```

**Visual Basic**

```
Public CB71_Humidity As double
```

**Description**

Additional characteristic for CB71 code - humidity in %.

#### II.4.1.1.2.3 CB71\_Nature

**C++**

```
HRESULT get_CB71_Nature(int* );
HRESULT put_CB71_Nature(int);
```

**C#**

```
public int CB71_Nature { get; set; }
```

**Visual Basic**

```
Public CB71_Nature As int
```

**Description**

Additional characteristics for CD71 code - 2 natures (resinous / non-resinous).

#### II.4.1.1.2.4 CB71\_Retreat

**C++**

```
HRESULT get_CB71_Retreat(double* );
HRESULT put_CB71_Retreat(double);
```

**C#**

```
public double CB71_Retreat { get; set; }
```

**Visual Basic**

```
Public CB71_Retreat As double
```

**Description**

Additional characteristic for CB71 code - retreat in %.

#### II.4.1.1.2.5 CS

**C++**

```
HRESULT get_CS(double* );
HRESULT put_CS(double);
```

**C#**

```
public double CS { get; set; }
```

**Visual Basic**

```
Public CS As double
```

**Description**

Reduction coefficient for shear limit stress (steel, aluminium) (not implemented).

#### II.4.1.1.2.6 Default

**C++**

```
HRESULT get_Default(VARIANT_BOOL* );
HRESULT put_Default(VARIANT_BOOL);
```

**C#**

```
public bool Default { get; set; }
```

**Visual Basic**

```
Public Default As Boolean
```

**Description**

Default material (1 - yes, 0 - no).

#### II.4.1.1.2.7 DumpCoef

**C++**

```
HRESULT get_DumpCoef(double* );
HRESULT put_DumpCoef(double);
```

**C#**

```
public double DumpCoef { get; set; }
```

**Visual Basic**

```
Public DumpCoef As double
```

**Description**

Dumping coefficient .

**II.4.1.1.2.8 E****C++**

```
HRESULT get_E(double*);  
HRESULT put_E(double);
```

**C#**

```
public double E { get; set; }
```

**Visual Basic**

```
Public E As double
```

**Description**

Young's modulus / axial Young modulus for timber.

**II.4.1.1.2.9 E\_5****C++**

```
HRESULT get_E_5(double*);  
HRESULT put_E_5(double);
```

**C#**

```
public double E_5 { get; set; }
```

**Visual Basic**

```
Public E_5 As double
```

**Description**

Young's 5% module (timber) only for EC5 code.

**II.4.1.1.2.10 E\_Trans****C++**

```
HRESULT get_E_Trans(double*);  
HRESULT put_E_Trans(double);
```

**C#**

```
public double E_Trans { get; set; }
```

**Visual Basic**

```
Public E_Trans As double
```

**Description**

Young transverse module (timber) only for EC5 code.

**II.4.1.1.2.11 EC\_Deformation****C++**

```
HRESULT get_EC_Deformation(double*);  
HRESULT put_EC_Deformation(double);
```

**C#**

```
public double EC_Deformation { get; set; }
```

**Visual Basic**

```
Public EC_Deformation As double
```

**Description**

Shear deformation module.

**II.4.1.1.2.12 GMean****C++**

```
HRESULT get_GMean(double*);  
HRESULT put_GMean(double);
```

**C#**

```
public double GMean { get; set; }
```

**Visual Basic**

```
Public GMean As double
```

**II.4.1.1.2.13 Kirchoff****C++**

```
HRESULT get_Kirchoff(double*);  
HRESULT put_Kirchoff(double);
```

**C#**

```
public double Kirchoff { get; set; }
```

**Visual Basic**

```
Public Kirchoff As double
```

**Description**

Shear (transversal) module.

**II.4.1.1.2.14 LX****C++**

```
HRESULT get_LX(double*);  
HRESULT put_LX(double);
```

**C#**

```
public double LX { get; set; }
```

**Visual Basic**

```
Public LX As double
```

**Description**

Thermal expansion coefficient.

**II.4.1.1.2.15 Name****C++**

```
HRESULT get_Name(BSTR*);  
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Material name.

**II.4.1.1.2.16 NU****C++**

```
HRESULT get_NU(double*);  
HRESULT put_NU(double);
```

**C#**

```
public double NU { get; set; }
```

**Visual Basic**

```
Public NU As Double
```

**Description**

Poisson ratio .

**II.4.1.1.2.17 Nuance****C++**

```
HRESULT get_Nuance(BSTR*);  
HRESULT put_Nuance(BSTR);
```

**C#**

```
public String Nuance { get; set; }
```

**Visual Basic**

```
Public Nuance As String
```

**Description**

Additional short material description.

**II.4.1.1.2.18 PN\_Deformation****C++**

```
HRESULT get_PN_Deformation(double*);  
HRESULT put_PN_Deformation(double);
```

**C#**

```
public double PN_Deformation { get; set; }
```

**Visual Basic**

```
Public PN_Deformation As Double
```

**Description**

Shear deformation module for Polish code.

**II.4.1.1.2.19 PN\_E\_Additional****C++**

```
HRESULT get_PN_E_Additional(double*);  
HRESULT put_PN_E_Additional(double);
```

**C#**

```
public double PN_E_Additional { get; set; }
```

**Visual Basic**

```
Public PN_E_Additional As Double
```

**Description**

Additional Young module (timber) for Polish code (not implemented).

#### II.4.1.1.2.20 PN\_E\_Trans

**C++**

```
HRESULT get_PN_E_Trans(double*);  
HRESULT put_PN_E_Trans(double);
```

**C#**

```
public double PN_E_Trans { get; set; }
```

**Visual Basic**

```
Public PN_E_Trans As Double
```

**Description**

Young transverse module (timber) for Polish code (not implemented).

#### II.4.1.1.2.21 RE

**C++**

```
HRESULT get_RE(double*);  
HRESULT put_RE(double);
```

**C#**

```
public double RE { get; set; }
```

**Visual Basic**

```
Public RE As Double
```

**Description**

Yield point (steel, aluminum), compression resistance (concrete).

#### II.4.1.1.2.22 RE\_AxCompr

**C++**

```
HRESULT get_RE_AxCompr(double*);  
HRESULT put_RE_AxCompr(double);
```

**C#**

```
public double RE_AxCompr { get; set; }
```

**Visual Basic**

```
Public RE_AxCompr As Double
```

**Description**

Axial compression resistance (timber) (not implemented).

#### II.4.1.1.2.23 RE\_AxTens

**C++**

```
HRESULT get_RE_AxTens(double*);
```

```
HRESULT put_RE_AxTens(double);
```

**C#**

```
public double RE_AxTens { get; set; }
```

**Visual Basic**

```
Public RE_AxTens As double
```

**Description**

Axial tension resistance ( timber).

#### II.4.1.1.2.24 RE\_Bending

**C++**

```
HRESULT get_RE_Bending(double*);  
HRESULT put_RE_Bending(double);
```

**C#**

```
public double RE_Bending { get; set; }
```

**Visual Basic**

```
Public RE_Bending As double
```

**Description**

Bending resistance ( timber).

#### II.4.1.1.2.25 RE\_Shear

**C++**

```
HRESULT get_RE_Shear(double*);  
HRESULT put_RE_Shear(double);
```

**C#**

```
public double RE_Shear { get; set; }
```

**Visual Basic**

```
Public RE_Shear As double
```

**Description**

Shear resistance ( timber).

#### II.4.1.1.2.26 RE\_TrCompr

**C++**

```
HRESULT get_RE_TrCompr(double*);  
HRESULT put_RE_TrCompr(double);
```

**C#**

```
public double RE_TrCompr { get; set; }
```

**Visual Basic**

```
Public RE_TrCompr As double
```

**Description**

Transverse compression resistance ( timber).

**II.4.1.1.2.27 RE\_TrTens****C++**

```
HRESULT get_RE_TrTens(double* );
HRESULT put_RE_TrTens(double);
```

**C#**

```
public double RE_TrTens { get; set; }
```

**Visual Basic**

```
Public RE_TrTens As double
```

**Description**

Transverse tension resistance (timber).

**II.4.1.1.2.28 RO****C++**

```
HRESULT get_RO(double* );
HRESULT put_RO(double);
```

**C#**

```
public double RO { get; set; }
```

**Visual Basic**

```
Public RO As double
```

**Description**

Material density (mass).

**II.4.1.1.2.29 RT****C++**

```
HRESULT get_RT(double* );
HRESULT put_RT(double);
```

**C#**

```
public double RT { get; set; }
```

**Visual Basic**

```
Public RT As double
```

**Description**

Tension resistance (steel).

**II.4.1.1.2.30 SecondName****C++**

```
HRESULT get_SecondName(BSTR* );
HRESULT put_SecondName(BSTR);
```

**C#**

```
public String SecondName { get; set; }
```

**Visual Basic**

```
Public SecondName As String
```

**Description**

Second name of the material, available only during editing material database under MS Access.

**II.4.1.1.2.31 Steel\_Thermal****C++**

```
HRESULT get_Steel_Thermal(VARIANT_BOOL* );
HRESULT put_Steel_Thermal(VARIANT_BOOL);
```

**C#**

```
public bool Steel_Thermal { get; set; }
```

**Visual Basic**

```
Public Steel_Thermal As Boolean
```

**Description**

Thermal flag for steel materials. When it is on, it refers to special processing during production. .

**II.4.1.1.2.32 Timber\_Type****C++**

```
HRESULT get_Timber_Type(IRobotMaterialTimberType* );
HRESULT put_Timber_Type(IRobotMaterialTimberType);
```

**C#**

```
public IRobotMaterialTimberType Timber_Type { get; set; }
```

**Visual Basic**

```
Public Timber_Type As IRobotMaterialTimberType
```

**Description**

Timber material type (glued multi-layered or normal) .

**II.4.1.1.2.33 Type****C++**

```
HRESULT get_Type(IRobotMaterialType* );
HRESULT put_Type(IRobotMaterialType);
```

**C#**

```
public IRobotMaterialType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotMaterialType
```

**Description**

Material type.

**II.4.1.1.3 IRobotMaterialData Methods**

The methods of the IRobotMaterialData class are listed here.

## Public Methods

	Name	Description
💡	LoadFromDBase (see page 666)	The function allows one to load the data on a material with the defined name to the material database. The returned logical value (True, False) indicates whether the information on the material with the defined name has been found in the material database or not. .
💡	SaveToDBase (see page 666)	Function saves material to the user database. .

### II.4.1.1.3.1 LoadFromDBase

#### C++

```
HRESULT LoadFromDBase(BSTR _material_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromDBase(String _material_name);
```

#### Visual Basic

```
Public Function LoadFromDBase(_material_name As String) As Boolean
```

#### Description

The function allows one to load the data on a material with the defined name to the material database. The returned logical value (True, False) indicates whether the information on the material with the defined name has been found in the material database or not. .

### II.4.1.1.3.2 SaveToDBase

#### C++

```
HRESULT SaveToDBase();
```

#### C#

```
public void SaveToDBase();
```

#### Visual Basic

```
Public Sub SaveToDBase()
```

#### Description

Function saves material to the user database. .

#### Version

Available since version 3.

### II.4.1.2 IRobotMaterialType

#### C++

```
enum IRobotMaterialType;
```

#### C#

```
public enum IRobotMaterialType;
```

#### Visual Basic

```
Public Enum IRobotMaterialType
```

## Members

Members	Description
I_MT_STEEL = 1	Steel .
I_MT_ALUMINIUM = 3	Aluminum .
I_MT_TIMBER = 4	Timber .
I_MT_CONCRETE = 2	Concrete .
I_MT_OTHER = 5	Other .
I_MT_ALL = 0	Identifier denoting all material types Available since version 1.7.

## Description

Types of materials accepted by Robot. .

### II.4.1.3 IRobotMaterialTimberType

#### C++

```
enum IRobotMaterialTimberType;
```

#### C#

```
public enum IRobotMaterialTimberType;
```

#### Visual Basic

```
Public Enum IRobotMaterialTimberType
```

## Members

Members	Description
I_MTT_NORMAL = 0	Normal timber .
I_MTT_GLUE_LAMINATED = 1	Multi-layered glued timber .
I_MTT_KERTO_S = 2	Available since version 1.7.
I_MTT_KERTO_Q = 3	Available since version 1.7.
I_MTT_KERTO_S_OLD = 4	Available since version 2.5.
I_MTT_KERTO_Q_OLD = 5	Available since version 2.5.

## Description

Types of timber materials defined by Robot. .

### II.4.1.4 IRobotMaterialModel

#### C++

```
enum IRobotMaterialModel;
```

#### C#

```
public enum IRobotMaterialModel;
```

#### Visual Basic

```
Public Enum IRobotMaterialModel
```

## Members

Members	Description
I_MM_ELASTIC = 0	Available since version 3.
I_MM_HUBER_MISES = 1	Available since version 3.
I_MM_DRUCKER_PRAGER = 2	Available since version 3.

I_MM_RANKIN = 3	Available since version 3.
I_MM_MOHR_COULOMB = 4	Available since version 3.

**Description**

Available material models.

**Version**

Available since version 3.

**II.4.1.5 IRobotMaterialElasticType****C++**

```
enum IRobotMaterialElasticType;
```

**C#**

```
public enum IRobotMaterialElasticType;
```

**Visual Basic**

```
Public Enum IRobotMaterialElasticType
```

**Members**

Members	Description
I_MET_PERFECTLY_PLASTIC = 0	Elastic-perfectly plastic model. Available since version 3.5.
I_MET_PLASTIC_WITH_HARDENING = 1	Elasto-plastic model with hardening . Available since version 3.5.

**Description**

Available models of material elasticity.

**Version**

Available since version 3.5.

**II.4.1.6 IRobotMaterialElasticModel****Class Hierarchy****C++**

```
interface IRobotMaterialElasticModel : IDispatch;
```

**C#**

```
public interface IRobotMaterialElasticModel;
```

**Visual Basic**

```
Public Interface IRobotMaterialElasticModel
```

**Description**

Model (see page 669) of material elasticity. .

**Version**

Available since version 3.5.

**II.4.1.6.1 IRobotMaterialElasticModel Members**

The following tables list the members exposed by IRobotMaterialElasticModel.

## Public Fields

	Name	Description
◆	Coeff (see page 669)	Coefficient E1 / E for the elasto-plastic model with hardening .
◆	Model (see page 669)	
◆	UnloadingCoeff (see page 670)	Factor for the mixed unloading method .
◆	UnloadingMethod (see page 670)	Unloading method.

### II.4.1.6.2 IRobotMaterialElasticModel Fields

The fields of the IRobotMaterialElasticModel class are listed here.

## Public Fields

	Name	Description
◆	Coeff (see page 669)	Coefficient E1 / E for the elasto-plastic model with hardening .
◆	Model (see page 669)	
◆	UnloadingCoeff (see page 670)	Factor for the mixed unloading method .
◆	UnloadingMethod (see page 670)	Unloading method.

#### II.4.1.6.2.1 Coeff

##### C++

```
HRESULT get_Coeff(double* );
HRESULT put_Coeff(double);
```

##### C#

```
public double Coeff { get; set; }
```

##### Visual Basic

```
Public Coeff As Double
```

##### Description

Coefficient E1 / E for the elasto-plastic model with hardening .

##### Version

Available since version 3.5.

#### II.4.1.6.2.2 Model

##### C++

```
HRESULT get_Model(IRobotMaterialElasticType* );
HRESULT put_Model(IRobotMaterialElasticType);
```

##### C#

```
public IRobotMaterialElasticType Model { get; set; }
```

##### Visual Basic

```
Public Model As IRobotMaterialElasticType
```

##### Version

Available since version 3.5.

#### II.4.1.6.2.3 UnloadingCoeff

##### C++

```
HRESULT get_UnloadingCoeff(double* );
HRESULT put_UnloadingCoeff(double);
```

**C#**

```
public double UnloadingCoeff { get; set; }
```

**Visual Basic**

```
Public UnloadingCoeff As Double
```

**Description**

Factor for the mixed unloading method .

**Version**

Available since version 3.5.

**II.4.1.6.2.4 UnloadingMethod****C++**

```
HRESULT get_UnloadingMethod(IRobotMaterialElasticUnloadingMethod* );
HRESULT put_UnloadingMethod(IRobotMaterialElasticUnloadingMethod);
```

**C#**

```
public IRobotMaterialElasticUnloadingMethod UnloadingMethod { get; set; }
```

**Visual Basic**

```
Public UnloadingMethod As IRobotMaterialElasticUnloadingMethod
```

**Description**

Unloading method.

**Version**

Available since version 3.5.

**II.4.1.7 IRobotMaterialElasticUnloadingMethod****C++**

```
enum IRobotMaterialElasticUnloadingMethod;
```

**C#**

```
public enum IRobotMaterialElasticUnloadingMethod;
```

**Visual Basic**

```
Public Enum IRobotMaterialElasticUnloadingMethod
```

**Members**

Members	Description
I_MEUM_ELASTIC = 1	Elastic method of unloading . Available since version 3.5.
I_MEUM_PLASTIC = 2	Plastic method of unloading . Available since version 3.5.
I_MEUM_DAMAGE = 3	Damage method of unloading . Available since version 3.5.
I_MEUM_MIXED = 4	Mixed method of unloading . Available since version 3.5.

**Description**

Available methods of material unloading.

**Version**

Available since version 3.5.

**II.4.2 Non-linear constraints**

Non-linear connection models.

**Version**

Available since version 3.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotNonlinearLinkCurveType ( <a href="#">see page 681</a> )	Available curve types:..
	IRobotNonlinearLinkModelType ( <a href="#">see page 693</a> )	Available relation types described by a non-linear model.
	IRobotNonlinearLinkSemiAxisType ( <a href="#">see page 693</a> )	Semi-axis types.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotNonlinearLinkServer ( <a href="#">see page 672</a> )	Server of non-linear connection models. .
	IRobotNonlinearLink ( <a href="#">see page 674</a> )	Definintion of a non-linear connection model. .
	IRobotNonlinearLinkParams ( <a href="#">see page 678</a> )	Set of parameters describing any curve.
	IRobotNonlinearLinkParamsLinear ( <a href="#">see page 679</a> )	Interface specific to the LINEAR curve type. .
	IRobotNonlinearLinkParamsBLinear ( <a href="#">see page 679</a> )	Interface specific to the B_LINEAR curve type.
	IRobotNonlinearLinkParamsParabolic ( <a href="#">see page 681</a> )	Interface specific to the PARABOLIC and PARABOLIC_EC2 curve types.
	IRobotNonlinearLinkParamsPlastic ( <a href="#">see page 683</a> )	Interface specific to the PERFECTLY_PLASTIC curve type.
	IRobotNonlinearLinkParamsPlasticHardening ( <a href="#">see page 685</a> )	Interface specific to the PLASTIC_HARDENING curve type.
	IRobotNonlinearLinkParamsGapHook ( <a href="#">see page 687</a> )	Interface specific to the GAP_HOOK curve type.
	IRobotNonlinearLinkParamsCustom ( <a href="#">see page 688</a> )	Interface specific to the CUSTOM curve type; it describes a curve composed of freely defined segments.
	IRobotNonlinearLinkParamsCustomSegment ( <a href="#">see page 691</a> )	Interface defining segment of the CUSTOM type curve.
	IRobotNonlinearLinkMngr ( <a href="#">see page 694</a> )	Support for relations of the non-linear model - direction type.

**II.4.2.1 IRobotNonlinearLinkServer****Class Hierarchy****C++**

```
interface IRobotNonlinearLinkServer : IDispatch;
```

**C#**

```
public interface IRobotNonlinearLinkServer;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkServer
```

**Description**

Server of non-linear connection models. .

**Version**

Available since version 3.

**II.4.2.1.1 IRobotNonlinearLinkServer Members**

The following tables list the members exposed by IRobotNonlinearLinkServer.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Count (☞ see page 672)	Number of available models.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Create (☞ see page 673)	Function creates and returns a new non-linear connection model.
❖	Find (☞ see page 673)	Function returns index of a model of the specified name. .
❖	Get (☞ see page 673)	Function takes the interface for defining a model of the specified index. .
❖	Remove (☞ see page 674)	Function deletes model of the specified index. .

**II.4.2.1.2 IRobotNonlinearLinkServer Fields**

The fields of the IRobotNonlinearLinkServer class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Count (☞ see page 672)	Number of available models.

**II.4.2.1.2.1 Count****C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of available models.

**Version**

Available since version 3.

**II.4.2.1.3 IRobotNonlinearLinkServer Methods**

The methods of the IRobotNonlinearLinkServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Create (☞ see page 673)	Function creates and returns a new non-linear connection model.

	Find (see page 673)	Function returns index of a model of the specified name. .
	Get (see page 673)	Function takes the interface for defining a model of the specified index. .
	Remove (see page 674)	Function deletes model of the specified index. .

#### II.4.2.1.3.1 Create

##### C++

```
HRESULT Create(BSTR _name, IRobotNonlinearLink** ret);
```

##### C#

```
public IRobotNonlinearLink Create(String _name);
```

##### Visual Basic

```
Public Function Create(_name As String) As IRobotNonlinearLink
```

##### Description

Function creates and returns a new non-linear connection model.

##### Version

Available since version 3.

#### II.4.2.1.3.2 Find

##### C++

```
HRESULT Find(BSTR _name, long* ret);
```

##### C#

```
public long Find(String _name);
```

##### Visual Basic

```
Public Function Find(_name As String) As long
```

##### Description

Function returns index of a model of the specified name. .

##### Version

Available since version 3.

#### II.4.2.1.3.3 Get

##### C++

```
HRESULT Get(long _idx, IRobotNonlinearLink** ret);
```

##### C#

```
public IRobotNonlinearLink Get(long _idx);
```

##### Visual Basic

```
Public Function Get(_idx As long) As IRobotNonlinearLink
```

##### Description

Function takes the interface for defining a model of the specified index. .

##### Version

Available since version 3.

#### II.4.2.1.3.4 Remove

##### C++

```
HRESULT Remove(long _idx);
```

##### C#

```
public void Remove(long _idx);
```

##### Visual Basic

```
Public Sub Remove(_idx As long)
```

##### Description

Function deletes model of the specified index. .

##### Version

Available since version 3.

#### II.4.2.2 IRobotNonlinearLink

##### Class Hierarchy

##### C++

```
interface IRobotNonlinearLink : IDispatch;
```

##### C#

```
public interface IRobotNonlinearLink;
```

##### Visual Basic

```
Public Interface IRobotNonlinearLink
```

##### Description

Definintion of a non-linear connection model. .

##### Version

Available since version 3.

#### II.4.2.2.1 IRobotNonlinearLink Members

The following tables list the members exposed by IRobotNonlinearLink.

##### Public Fields

	Name	Description
◆	ModelType ( <a href="#">see page 675</a> )	Type of the relationship described by the model.
◆	Name ( <a href="#">see page 675</a> )	Model name.
◆	Symetry ( <a href="#">see page 676</a> )	Curve symmetry.

##### Public Methods

	Name	Description
◆	GetCurveType ( <a href="#">see page 676</a> )	Function takes the curve type for the indicated semi-axis. .
◆	GetParams ( <a href="#">see page 677</a> )	Function takes a set of parameters describing a given curve. .
◆	SetCurveType ( <a href="#">see page 677</a> )	Function sets the curve type for the indicated semi-axis. .
◆	SetParams ( <a href="#">see page 677</a> )	Function updates the set of parameters describing a curve..

### II.4.2.2.2 IRobotNonlinearLink Fields

The fields of the IRobotNonlinearLink class are listed here.

#### Public Fields

	Name	Description
◆	ModelType (see page 675)	Type of the relationship described by the model.
◆	Name (see page 675)	Model name.
◆	Symetry (see page 676)	Curve symmetry.

#### II.4.2.2.2.1 ModelType

##### C++

```
HRESULT get_ModelType(IRobotNonlinearLinkModelType* );
HRESULT put_ModelType(IRobotNonlinearLinkModelType);
```

##### C#

```
public IRobotNonlinearLinkModelType ModelType { get; set; }
```

##### Visual Basic

```
Public ModelType As IRobotNonlinearLinkModelType
```

##### Description

Type of the relationship described by the model.

##### Version

Available since version 3.

#### II.4.2.2.2.2 Name

##### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

##### C#

```
public String Name { get; set; }
```

##### Visual Basic

```
Public Name As String
```

##### Description

Model name.

##### Version

Available since version 3.

#### II.4.2.2.2.3 Symetry

##### C++

```
HRESULT get_Symetry(VARIANT_BOOL* );
```

##### C#

```
public bool Symetry { get; }
```

##### Visual Basic

```
Public ReadOnly Symetry As Boolean
```

**Description**

Curve symmetry.

**Version**

Available since version 3.

**II.4.2.2.3 IRobotNonlinearLink Methods**

The methods of the IRobotNonlinearLink class are listed here.

**Public Methods**

	Name	Description
ESH	GetCurveType (see page 676)	Function takes the curve type for the indicated semi-axis. .
ESH	GetParams (see page 677)	Function takes a set of parameters describing a given curve. .
ESH	SetCurveType (see page 677)	Function sets the curve type for the indicated semi-axis. .
ESH	SetParams (see page 677)	Function updates the set of parameters describing a curve. .

**II.4.2.2.3.1 GetCurveType****C++**

```
HRESULT GetCurveType(IRobotNonlinearLinkSemiAxisType _axis_type = I_NLSAT_ANY,
IRobotNonlinearLinkCurveType* ret);
```

**C#**

```
public IRobotNonlinearLinkCurveType GetCurveType(IRobotNonlinearLinkSemiAxisType _axis_type =
I_NLSAT_ANY);
```

**Visual Basic**

```
Public Function GetCurveType(Optional _axis_type As IRobotNonlinearLinkSemiAxisType =
I_NLSAT_ANY) As IRobotNonlinearLinkCurveType
```

**Description**

Function takes the curve type for the indicated semi-axis. .

**Version**

Available since version 3.

**II.4.2.2.3.2 GetParams****C++**

```
HRESULT GetParams(IRobotNonlinearLinkSemiAxisType _axis_type = I_NLSAT_ANY,
IRobotNonlinearLinkParams** ret);
```

**C#**

```
public IRobotNonlinearLinkParams GetParams(IRobotNonlinearLinkSemiAxisType _axis_type =
I_NLSAT_ANY);
```

**Visual Basic**

```
Public Function GetParams(Optional _axis_type As IRobotNonlinearLinkSemiAxisType =
I_NLSAT_ANY) As IRobotNonlinearLinkParams
```

**Description**

Function takes a set of parameters describing a given curve. .

**Version**

Available since version 3.

### II.4.2.2.3.3 SetCurveType

#### C++

```
HRESULT SetCurveType(IRobotNonlinearLinkCurveType _curve_type,
IRobotNonlinearLinkSemiAxisType _axis_type = I_NLSAT_ANY);
```

#### C#

```
public void SetCurveType(IRobotNonlinearLinkCurveType _curve_type,
IRobotNonlinearLinkSemiAxisType _axis_type = I_NLSAT_ANY);
```

#### Visual Basic

```
Public Sub SetCurveType(_curve_type As IRobotNonlinearLinkCurveType, Optional _axis_type As
IRobotNonlinearLinkSemiAxisType = I_NLSAT_ANY)
```

#### Description

Function sets the curve type for the indicated semi-axis. .

#### Version

Available since version 3.

### II.4.2.2.3.4 SetParams

#### C++

```
HRESULT SetParams(IRobotNonlinearLinkParams _params, IRobotNonlinearLinkSemiAxisType
_axis_type = I_NLSAT_ANY);
```

#### C#

```
public void SetParams(IRobotNonlinearLinkParams _params, IRobotNonlinearLinkSemiAxisType
_axis_type = I_NLSAT_ANY);
```

#### Visual Basic

```
Public Sub SetParams(_params As IRobotNonlinearLinkParams, Optional _axis_type As
IRobotNonlinearLinkSemiAxisType = I_NLSAT_ANY)
```

#### Description

Function updates the set of parameters describing a curve. .

#### Version

Available since version 3.

### II.4.2.3 IRobotNonlinearLinkParams

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearLinkParams : IDispatch;
```

#### C#

```
public interface IRobotNonlinearLinkParams;
```

#### Visual Basic

```
Public Interface IRobotNonlinearLinkParams
```

#### Description

Set of parameters describing any curve.

**Version**

Available since version 3.

**II.4.2.3.1 IRobotNonlinearLinkParams Members**

The following tables list the members exposed by IRobotNonlinearLinkParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CurveType (see page 678)	Curve type.

**II.4.2.3.2 IRobotNonlinearLinkParams Fields**

The fields of the IRobotNonlinearLinkParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CurveType (see page 678)	Curve type.

**II.4.2.3.2.1 CurveType****C++**

```
HRESULT get_CurveType( IRobotNonlinearLinkCurveType* );
```

**C#**

```
public IRobotNonlinearLinkCurveType CurveType { get; }
```

**Visual Basic**

```
Public ReadOnly CurveType As IRobotNonlinearLinkCurveType
```

**Description**

Curve type.

**Version**

Available since version 3.

**II.4.2.4 IRobotNonlinearLinkParamsLinear****Class Hierarchy****C++**

```
interface IRobotNonlinearLinkParamsLinear : IRobotNonlinearLinkParams;
```

**C#**

```
public interface IRobotNonlinearLinkParamsLinear : IRobotNonlinearLinkParams;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkParamsLinear
```

**Description**

Interface specific to the LINEAR curve type. .

**Version**

Available since version 3.

#### II.4.2.4.1 IRobotNonlinearLinkParamsLinear Members

The following tables list the members exposed by IRobotNonlinearLinkParamsLinear.

##### Public Fields

	Name	Description
◆	CurveType (see page 678)	Curve type.
◆	K (see page 679)	

#### II.4.2.4.2 IRobotNonlinearLinkParamsLinear Fields

The fields of the IRobotNonlinearLinkParamsLinear class are listed here.

##### Public Fields

	Name	Description
◆	K (see page 679)	

#### II.4.2.4.2.1 K

##### C++

```
HRESULT get_K( double* );
HRESULT put_K( double );
```

##### C#

```
public double K { get; set; }
```

##### Visual Basic

```
Public K As Double
```

##### Version

Available since version 3.

#### II.4.2.5 IRobotNonlinearLinkParamsBLinear

##### Class Hierarchy

##### C++

```
interface IRobotNonlinearLinkParamsBLinear : IRobotNonlinearLinkParams;
```

##### C#

```
public interface IRobotNonlinearLinkParamsBLinear : IRobotNonlinearLinkParams;
```

##### Visual Basic

```
Public Interface IRobotNonlinearLinkParamsBLinear
```

##### Description

Interface specific to the B\_LINEAR curve type.

##### Version

Available since version 3.

#### II.4.2.5.1 IRobotNonlinearLinkParamsBLinear Members

The following tables list the members exposed by IRobotNonlinearLinkParamsBLinear.

## Public Fields

	Name	Description
◆	CurveType (see page 678)	Curve type.
◆	D1 (see page 680)	
◆	K1 (see page 680)	
◆	K2 (see page 681)	

### II.4.2.5.2 IRobotNonlinearLinkParamsBLinear Fields

The fields of the IRobotNonlinearLinkParamsBLinear class are listed here.

## Public Fields

	Name	Description
◆	D1 (see page 680)	
◆	K1 (see page 680)	
◆	K2 (see page 681)	

### II.4.2.5.2.1 D1

#### C++

```
HRESULT get_D1( double* );
HRESULT put_D1( double );
```

#### C#

```
public double D1 { get; set; }
```

#### Visual Basic

```
Public D1 As double
```

#### Version

Available since version 3.

### II.4.2.5.2.2 K1

#### C++

```
HRESULT get_K1( double* );
HRESULT put_K1( double );
```

#### C#

```
public double K1 { get; set; }
```

#### Visual Basic

```
Public K1 As double
```

#### Version

Available since version 3.

### II.4.2.5.2.3 K2

#### C++

```
HRESULT get_K2( double* );
HRESULT put_K2( double );
```

#### C#

```
public double K2 { get; set; }
```

**Visual Basic**

```
Public K2 As double
```

**Version**

Available since version 3.

**II.4.2.6 IRobotNonlinearLinkCurveType****C++**

```
enum IRobotNonlinearLinkCurveType;
```

**C#**

```
public enum IRobotNonlinearLinkCurveType;
```

**Visual Basic**

```
Public Enum IRobotNonlinearLinkCurveType
```

**Members**

Members	Description
I_NLCT_LINEAR = 1	Available since version 3.
I_NLCT_B_LINEAR = 2	Available since version 3.
I_NLCT_PARABOLIC = 3	Available since version 3.
I_NLCT_PARABOLIC_EC2 = 4	Available since version 3.
I_NLCT_PERFECTLY_PLASTIC = 5	Available since version 3.
I_NLCT_PLASTIC_WITH_HARDENING = 6	Available since version 3.
I_NLCT_GAP_HOOK = 7	Available since version 3.
I_NLCT_CUSTOM = 8	Available since version 3.

**Description**

Available curve types:.

**Version**

Available since version 3.

**II.4.2.7 IRobotNonlinearLinkParamsParabolic****Class Hierarchy****C++**

```
interface IRobotNonlinearLinkParamsParabolic : IRobotNonlinearLinkParams;
```

**C#**

```
public interface IRobotNonlinearLinkParamsParabolic : IRobotNonlinearLinkParams;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkParamsParabolic
```

**Description**

Interface specific to the PARABOLIC and PARABOLIC\_EC2 curve types.

**Version**

Available since version 3.

### II.4.2.7.1 IRobotNonlinearLinkParamsParabolic Members

The following tables list the members exposed by IRobotNonlinearLinkParamsParabolic.

#### Public Fields

	Name	Description
◆	CurveType (see page 678)	Curve type.
◆	Dlim (see page 682)	
◆	Dmax (see page 683)	
◆	Flim (see page 683)	
◆	K (see page 683)	

### II.4.2.7.2 IRobotNonlinearLinkParamsParabolic Fields

The fields of the IRobotNonlinearLinkParamsParabolic class are listed here.

#### Public Fields

	Name	Description
◆	Dlim (see page 682)	
◆	Dmax (see page 683)	
◆	Flim (see page 683)	
◆	K (see page 683)	

#### II.4.2.7.2.1 Dlim

##### C++

```
HRESULT get_Dlim( double* );
HRESULT put_Dlim( double );
```

##### C#

```
public double Dlim { get; set; }
```

##### Visual Basic

```
Public Dlim As Double
```

##### Version

Available since version 3.

#### II.4.2.7.2.2 Dmax

##### C++

```
HRESULT get_Dmax( double* );
HRESULT put_Dmax( double );
```

##### C#

```
public double Dmax { get; set; }
```

##### Visual Basic

```
Public Dmax As Double
```

##### Version

Available since version 3.

### II.4.2.7.2.3 Flim

#### C++

```
HRESULT get_Flim( double* );
HRESULT put_Flim( double );
```

#### C#

```
public double Flim { get; set; }
```

#### Visual Basic

```
Public Flim As Double
```

#### Version

Available since version 3.

### II.4.2.7.2.4 K

#### C++

```
HRESULT get_K( double* );
HRESULT put_K( double );
```

#### C#

```
public double K { get; set; }
```

#### Visual Basic

```
Public K As Double
```

#### Version

Available since version 3.

## II.4.2.8 IRobotNonlinearLinkParamsPlastic

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearLinkParamsPlastic : IRobotNonlinearLinkParams;
```

#### C#

```
public interface IRobotNonlinearLinkParamsPlastic : IRobotNonlinearLinkParams;
```

#### Visual Basic

```
Public Interface IRobotNonlinearLinkParamsPlastic
```

#### Description

Interface specific to the PERFECTLY\_PLASTIC curve type.

#### Version

Available since version 3.

### II.4.2.8.1 IRobotNonlinearLinkParamsPlastic Members

The following tables list the members exposed by IRobotNonlinearLinkParamsPlastic.

#### Public Fields

	Name	Description
◆	CurveType (see page 678)	Curve type.
◆	Dlim (see page 684)	

◆	Flim (see page 684)	
◆	K (see page 685)	
◆	W (see page 685)	

#### II.4.2.8.2 IRobotNonlinearLinkParamsPlastic Fields

The fields of the IRobotNonlinearLinkParamsPlastic class are listed here.

##### Public Fields

	Name	Description
◆	Dlim (see page 684)	
◆	Flim (see page 684)	
◆	K (see page 685)	
◆	W (see page 685)	

#### II.4.2.8.2.1 Dlim

##### C++

```
HRESULT get_Dlim( double* );
HRESULT put_Dlim( double );
```

##### C#

```
public double Dlim { get; set; }
```

##### Visual Basic

```
Public Dlim As Double
```

##### Version

Available since version 3.

#### II.4.2.8.2.2 Flim

##### C++

```
HRESULT get_Flim( double* );
HRESULT put_Flim( double );
```

##### C#

```
public double Flim { get; set; }
```

##### Visual Basic

```
Public Flim As Double
```

##### Version

Available since version 3.

#### II.4.2.8.2.3 K

##### C++

```
HRESULT get_K( double* );
HRESULT put_K( double );
```

##### C#

```
public double K { get; set; }
```

##### Visual Basic

```
Public K As Double
```

**Version**

Available since version 3.

**II.4.2.8.2.4 W****C++**

```
HRESULT get_W( double* );
HRESULT put_W( double );
```

**C#**

```
public double W { get; set; }
```

**Visual Basic**

```
Public W As Double
```

**Version**

Available since version 3.

**II.4.2.9 IRobotNonlinearLinkParamsPlasticHardening****Class Hierarchy****C++**

```
interface IRobotNonlinearLinkParamsPlasticHardening : IRobotNonlinearLinkParams;
```

**C#**

```
public interface IRobotNonlinearLinkParamsPlasticHardening : IRobotNonlinearLinkParams;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkParamsPlasticHardening
```

**Description**

Interface specific to the PLASTIC\_HARDENING curve type.

**Version**

Available since version 3.

**II.4.2.9.1 IRobotNonlinearLinkParamsPlasticHardening Members**

The following tables list the members exposed by IRobotNonlinearLinkParamsPlasticHardening.

**Public Fields**

	Name	Description
❖	CurveType ( <a href="#">see page 678</a> )	Curve type.
❖	Dlim ( <a href="#">see page 686</a> )	
❖	Flim ( <a href="#">see page 686</a> )	
❖	K0 ( <a href="#">see page 686</a> )	
❖	K1 ( <a href="#">see page 687</a> )	
❖	W ( <a href="#">see page 687</a> )	

**II.4.2.9.2 IRobotNonlinearLinkParamsPlasticHardening Fields**

The fields of the IRobotNonlinearLinkParamsPlasticHardening class are listed here.

## Public Fields

	Name	Description
◆	Dlim ( <a href="#">see page 686</a> )	
◆	Flim ( <a href="#">see page 686</a> )	
◆	K0 ( <a href="#">see page 686</a> )	
◆	K1 ( <a href="#">see page 687</a> )	
◆	W ( <a href="#">see page 687</a> )	

### II.4.2.9.2.1 Dlim

#### C++

```
HRESULT get_Dlim( double* );
HRESULT put_Dlim( double );
```

#### C#

```
public double Dlim { get; set; }
```

#### Visual Basic

```
Public Dlim As Double
```

#### Version

Available since version 3.

### II.4.2.9.2.2 Flim

#### C++

```
HRESULT get_Flim( double* );
HRESULT put_Flim( double );
```

#### C#

```
public double Flim { get; set; }
```

#### Visual Basic

```
Public Flim As Double
```

#### Version

Available since version 3.

### II.4.2.9.2.3 K0

#### C++

```
HRESULT get_K0( double* );
HRESULT put_K0( double );
```

#### C#

```
public double K0 { get; set; }
```

#### Visual Basic

```
Public K0 As Double
```

#### Version

Available since version 3.

#### II.4.2.9.2.4 K1

**C++**

```
HRESULT get_K1( double* );
HRESULT put_K1( double );
```

**C#**

```
public double K1 { get; set; }
```

**Visual Basic**

```
Public K1 As Double
```

**Version**

Available since version 3.

#### II.4.2.9.2.5 W

**C++**

```
HRESULT get_W( double* );
HRESULT put_W( double );
```

**C#**

```
public double W { get; set; }
```

**Visual Basic**

```
Public W As Double
```

**Version**

Available since version 3.

#### II.4.2.10 IRobotNonlinearLinkParamsGapHook

**Class Hierarchy****C++**

```
interface IRobotNonlinearLinkParamsGapHook : IRobotNonlinearLinkParams;
```

**C#**

```
public interface IRobotNonlinearLinkParamsGapHook : IRobotNonlinearLinkParams;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkParamsGapHook
```

**Description**

Interface specific to the GAP\_HOOK curve type.

**Version**

Available since version 3.

#### II.4.2.10.1 IRobotNonlinearLinkParamsGapHook Members

The following tables list the members exposed by IRobotNonlinearLinkParamsGapHook.

**Public Fields**

	Name	Description
◆	CurveType (see page 678)	Curve type.
◆	D (see page 688)	

	K ( <a href="#">see page 688</a> )	
--	------------------------------------	--

#### II.4.2.10.2 IRobotNonlinearLinkParamsGapHook Fields

The fields of the IRobotNonlinearLinkParamsGapHook class are listed here.

##### Public Fields

	Name	Description
	D ( <a href="#">see page 688</a> )	
	K ( <a href="#">see page 688</a> )	

#### II.4.2.10.2.1 D

##### C++

```
HRESULT get_D( double* );
HRESULT put_D( double );
```

##### C#

```
public double D { get; set; }
```

##### Visual Basic

```
Public D As Double
```

##### Version

Available since version 3.

#### II.4.2.10.2.2 K

##### C++

```
HRESULT get_K( double* );
HRESULT put_K( double );
```

##### C#

```
public double K { get; set; }
```

##### Visual Basic

```
Public K As Double
```

##### Version

Available since version 3.

#### II.4.2.11 IRobotNonlinearLinkParamsCustom

##### Class Hierarchy

##### C++

```
interface IRobotNonlinearLinkParamsCustom : IRobotNonlinearLinkParams;
```

##### C#

```
public interface IRobotNonlinearLinkParamsCustom : IRobotNonlinearLinkParams;
```

##### Visual Basic

```
Public Interface IRobotNonlinearLinkParamsCustom
```

##### Description

Interface specific to the CUSTOM curve type; it describes a curve composed of freely defined segments.

## Version

Available since version 3.

### II.4.2.11.1 IRobotNonlinearLinkParamsCustom Members

The following tables list the members exposed by IRobotNonlinearLinkParamsCustom.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 689</a> )	Number of function segments.
◆	CurveType ( <a href="#">see page 678</a> )	Curve type.

#### Public Methods

	Name	Description
◆	Get ( <a href="#">see page 690</a> )	Function takes a set of parameters for defining a segment of a given index. .
◆	New ( <a href="#">see page 690</a> )	Function creates and returns a new curve segment. .
◆	Remove ( <a href="#">see page 690</a> )	Function deletes a segment of a given index. .
◆	Set ( <a href="#">see page 691</a> )	Function updates parameters describing a segment of a given index. .

### II.4.2.11.2 IRobotNonlinearLinkParamsCustom Fields

The fields of the IRobotNonlinearLinkParamsCustom class are listed here.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 689</a> )	Number of function segments.

### II.4.2.11.2.1 Count

#### C++

```
HRESULT get_Count( long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of function segments.

#### Version

Available since version 3.

### II.4.2.11.3 IRobotNonlinearLinkParamsCustom Methods

The methods of the IRobotNonlinearLinkParamsCustom class are listed here.

#### Public Methods

	Name	Description
◆	Get ( <a href="#">see page 690</a> )	Function takes a set of parameters for defining a segment of a given index. .
◆	New ( <a href="#">see page 690</a> )	Function creates and returns a new curve segment. .
◆	Remove ( <a href="#">see page 690</a> )	Function deletes a segment of a given index. .

	Set (see page 691)	Function updates parameters describing a segment of a given index. .
-----------------------------------------------------------------------------------	--------------------	----------------------------------------------------------------------

#### II.4.2.11.3.1 Get

**C++**

```
HRESULT Get(long _idx, IRobotNonlinearLinkParamsCustomSegment** ret);
```

**C#**

```
public IRobotNonlinearLinkParamsCustomSegment Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotNonlinearLinkParamsCustomSegment
```

**Description**

Function takes a set of parameters for defining a segment of a given index. .

**Version**

Available since version 3.

#### II.4.2.11.3.2 New

**C++**

```
HRESULT New(IRobotNonlinearLinkParamsCustomSegment** ret);
```

**C#**

```
public IRobotNonlinearLinkParamsCustomSegment New();
```

**Visual Basic**

```
Public Function New() As IRobotNonlinearLinkParamsCustomSegment
```

**Description**

Function creates and returns a new curve segment. .

**Version**

Available since version 3.

#### II.4.2.11.3.3 Remove

**C++**

```
HRESULT Remove(long _idx);
```

**C#**

```
public void Remove(long _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As long)
```

**Description**

Function deletes a segment of a given index. .

**Version**

Available since version 3.

#### II.4.2.11.3.4 Set

##### C++

```
HRESULT Set(long _idx, IRobotNonlinearLinkParamsCustomSegment* _segment);
```

##### C#

```
public void Set(long _idx, IRobotNonlinearLinkParamsCustomSegment _segment);
```

##### Visual Basic

```
Public Sub Set(_idx As long, ByRef _segment As IRobotNonlinearLinkParamsCustomSegment)
```

##### Description

Function updates parameters describing a segment of a given index..

##### Version

Available since version 3.

#### II.4.2.12 IRobotNonlinearLinkParamsCustomSegment

##### Class Hierarchy

##### C++

```
interface IRobotNonlinearLinkParamsCustomSegment : IDispatch;
```

##### C#

```
public interface IRobotNonlinearLinkParamsCustomSegment;
```

##### Visual Basic

```
Public Interface IRobotNonlinearLinkParamsCustomSegment
```

##### Description

Interface defining segment of the CUSTOM type curve.

##### Version

Available since version 3.

#### II.4.2.12.1 IRobotNonlinearLinkParamsCustomSegment Members

The following tables list the members exposed by IRobotNonlinearLinkParamsCustomSegment.

##### Public Fields

	Name	Description
❖	Constant (see page 692)	Flag indicating if function expression is a constant number ( $y = a$ ).
❖	Expression (see page 692)	Expression representing the functional relationship.
❖	OriginPoint (see page 692)	Segment beginning point.

#### II.4.2.12.2 IRobotNonlinearLinkParamsCustomSegment Fields

The fields of the IRobotNonlinearLinkParamsCustomSegment class are listed here.

##### Public Fields

	Name	Description
❖	Constant (see page 692)	Flag indicating if function expression is a constant number ( $y = a$ ).
❖	Expression (see page 692)	Expression representing the functional relationship.
❖	OriginPoint (see page 692)	Segment beginning point.

#### II.4.2.12.2.1 Constant

##### C++

```
HRESULT get_Constant(VARIANT_BOOL* );
HRESULT put_Constant(VARIANT_BOOL);
```

##### C#

```
public bool Constant { get; set; }
```

##### Visual Basic

```
Public Constant As Boolean
```

##### Description

Flag indicating if function expression is a constant number ( $y = a$ ).

##### Version

Available since version 3.

#### II.4.2.12.2.2 Expression

##### C++

```
HRESULT get_Expression(BSTR* );
HRESULT put_Expression(BSTR);
```

##### C#

```
public String Expression { get; set; }
```

##### Visual Basic

```
Public Expression As String
```

##### Description

Expression representing the functional relationship.

##### Version

Available since version 3.

#### II.4.2.12.2.3 OriginPoint

##### C++

```
HRESULT get-OriginPoint(double* );
HRESULT put-OriginPoint(double);
```

##### C#

```
public double OriginPoint { get; set; }
```

##### Visual Basic

```
Public OriginPoint As double
```

##### Description

Segment beginning point.

##### Version

Available since version 3.

### II.4.2.13 IRobotNonlinearLinkModelType

#### C++

```
enum IRobotNonlinearLinkModelType;
```

#### C#

```
public enum IRobotNonlinearLinkModelType;
```

#### Visual Basic

```
Public Enum IRobotNonlinearLinkModelType
```

#### Members

Members	Description
I_NLMT_FORCE_DISPLACEMENT = 1	Force - displacement. Available since version 3.
I_NLMT_MOMENT_ROTATION = 2	Moment - rotation. Available since version 3.

#### Description

Available relation types described by a non-linear model.

#### Version

Available since version 3.

### II.4.2.14 IRobotNonlinearLinkSemiAxisType

#### C++

```
enum IRobotNonlinearLinkSemiAxisType;
```

#### C#

```
public enum IRobotNonlinearLinkSemiAxisType;
```

#### Visual Basic

```
Public Enum IRobotNonlinearLinkSemiAxisType
```

#### Members

Members	Description
I_NLSAT_POSITIVE = 1	Positive semi-axis. Available since version 3.
I_NLSAT_NEGATIVE = 2	Negative semi-axis.
I_NLSAT_ANY = 3	Arbitrary semi-axis. Available since version 3.

#### Description

Semi-axis types.

#### Version

Available since version 3.

### II.4.2.15 IRobotNonlinearLinkMngr

#### Class Hierarchy

#### C++

```
interface IRobotNonlinearLinkMngr : IDispatch;
```

**C#**

```
public interface IRobotNonlinearLinkMngr;
```

**Visual Basic**

```
Public Interface IRobotNonlinearLinkMngr
```

**Description**

Support for relations of the non-linear model - direction type.

**Version**

Available since version 3.

**II.4.2.15.1 IRobotNonlinearLinkMngr Members**

The following tables list the members exposed by IRobotNonlinearLinkMngr.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Delete ( <a href="#">see page 694</a> )	Function deletes a non-linear model from the indicated direction. .
≡	Get ( <a href="#">see page 695</a> )	Function returns the name of a non-linear model assigned to the specified direction.
≡	IsDefined ( <a href="#">see page 695</a> )	Function checks a non-zero value (True), if a non-linear model has been assigned to the indicated direction. .
≡	Set ( <a href="#">see page 695</a> )	Function determines a non-linear model for the specified drection. .

**II.4.2.15.2 IRobotNonlinearLinkMngr Methods**

The methods of the IRobotNonlinearLinkMngr class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Delete ( <a href="#">see page 694</a> )	Function deletes a non-linear model from the indicated direction. .
≡	Get ( <a href="#">see page 695</a> )	Function returns the name of a non-linear model assigned to the specified direction.
≡	IsDefined ( <a href="#">see page 695</a> )	Function checks a non-zero value (True), if a non-linear model has been assigned to the indicated direction. .
≡	Set ( <a href="#">see page 695</a> )	Function determines a non-linear model for the specified drection. .

**II.4.2.15.2.1 Delete****C++**

```
HRESULT Delete(IRobotDegreeOfFreedom _direction);
```

**C#**

```
public void Delete(IRobotDegreeOfFreedom _direction);
```

**Visual Basic**

```
Public Sub Delete(_direction As IRobotDegreeOfFreedom)
```

**Description**

Function deletes a non-linear model from the indicated direction. .

**Version**

Available since version 3.

#### II.4.2.15.2.2 Get

##### C++

```
HRESULT Get(IRobotDegreeOfFreedom _direction, BSTR* ret);
```

##### C#

```
public String Get(IRobotDegreeOfFreedom _direction);
```

##### Visual Basic

```
Public Function Get(_direction As IRobotDegreeOfFreedom) As String
```

##### Description

Function returns the name of a non-linear model assigned to the specified direction.

##### Version

Available since version 3.

#### II.4.2.15.2.3 IsDefined

##### C++

```
HRESULT IsDefined(IRobotDegreeOfFreedom _direction, VARIANT_BOOL* ret);
```

##### C#

```
public bool IsDefined(IRobotDegreeOfFreedom _direction);
```

##### Visual Basic

```
Public Function IsDefined(_direction As IRobotDegreeOfFreedom) As Boolean
```

##### Description

Function checks a non-zero value (True), if a non-linear model has been assigned to the indicated direction. .

##### Version

Available since version 3.

#### II.4.2.15.2.4 Set

##### C++

```
HRESULT Set(IRobotDegreeOfFreedom _direction, BSTR _model_name);
```

##### C#

```
public void Set(IRobotDegreeOfFreedom _direction, String _model_name);
```

##### Visual Basic

```
Public Sub Set(_direction As IRobotDegreeOfFreedom, _model_name As String)
```

##### Description

Function determines a non-linear model for the specified direction. .

##### Version

Available since version 3.

## II.5 Objects

### Enumerations

	<b>Name</b>	<b>Description</b>
	IRobotObjModificationType ( <a href="#">see page 739</a> )	Set of identifiers has been defined for modifications which may be performed on specified objects - structure components. .
	IRobotObjOperationType ( <a href="#">see page 739</a> )	Set of identifiers has been defined for operations which may be performed on specified objects - structure components. .
	IRobotObjPartType ( <a href="#">see page 773</a> )	
	IRobotObjLocalXDirDefinitionType ( <a href="#">see page 788</a> )	Type of direction definition for the local X axis of the plate. .

### Interfaces

	<b>Name</b>	<b>Description</b>
	IRobotObjModification ( <a href="#">see page 739</a> )	The basic interface describing modification of the initial object geometry .
	IRobotObjOperation ( <a href="#">see page 742</a> )	The basic interface describing transformation (translation, rotation, scaling, deformation) of the result of modification (the so called reference or image).
	IRobotObjOperationCollection ( <a href="#">see page 743</a> )	Collection of operations.
	IRobotObjModifExtrusion ( <a href="#">see page 743</a> )	The interface describing a modification of initial object geometry - extrude. .
	IRobotObjModifLathe ( <a href="#">see page 744</a> )	Interface describing modification of initial geometry of the object - revolve operation.
	IRobotObjModifPyramid ( <a href="#">see page 746</a> )	Interface describing modification of initial object geometry - extrude along polyline.
	IRobotObjOperTranslation ( <a href="#">see page 748</a> )	The interface describing the operation of translation that transforms the result of modification of the initial object geometry. .
	IRobotObjOperScaling ( <a href="#">see page 749</a> )	The interface describing the operation of scaling, transforming the result of modification of object geometry .
	IRobotObjOperRotation ( <a href="#">see page 750</a> )	Interface describing the operation of rotation that transforms the result of modification of object geometry..
	IRobotObjOperMeshing ( <a href="#">see page 752</a> )	The interface describing a deformation that transforms the result of modification of initial object geometry .
	IRobotObjEdge ( <a href="#">see page 754</a> )	Interface describing an object edge.
	IRobotObjAttributes ( <a href="#">see page 757</a> )	The interface describing additional attributes of an object or its part.
	IRobotObjPart ( <a href="#">see page 762</a> )	Interface describing an object component. .
	IRobotObjModificationCollection ( <a href="#">see page 763</a> )	Collection of modifications .
	IRobotObjEdgeCollection ( <a href="#">see page 764</a> )	Collection of edges.
	IRobotObjObject ( <a href="#">see page 764</a> )	
	IRobotObjObjectServer ( <a href="#">see page 773</a> )	Server of object that are structure components .
	IRobotObjPartMain ( <a href="#">see page 788</a> )	Main object component. It contains definition of the base geometry and geometric transformations which result in the geometry of the entire object. All object edges are also saved in the main component.
	IRobotObjPartReference ( <a href="#">see page 793</a> )	Object component definition which is the reference of the main component generated due to transformations. .
	IRobotObjPart2 ( <a href="#">see page 795</a> )	Extended definition of the object component. .
	IRobotObjMesh ( <a href="#">see page 797</a> )	Interface allowing management of the finite element mesh.

	IRobotObjEdgeSelection ( <a href="#">see page 801</a> )	An interface which describes a selection of edges of panels and other surface objects.
-----------------------------------------------------------------------------------	---------------------------------------------------------	----------------------------------------------------------------------------------------

## II.5.1 Complex attributes of an object

### Enumerations

	Name	Description
	IRobotThicknessType ( <a href="#">see page 717</a> )	
	IRobotThicknessUpliftType ( <a href="#">see page 718</a> )	
	IRobotThicknessHomoType ( <a href="#">see page 723</a> )	
	IRobotThicknessOrthoType ( <a href="#">see page 723</a> )	.
	IRobotThicknessOrthoDirType ( <a href="#">see page 724</a> )	.
	IRobotThicknessMatrixValue ( <a href="#">see page 735</a> )	Physical interpretation of individual matrix values is described in the file of the Robot program help.

### Interfaces

	Name	Description
	IRobotThicknessData ( <a href="#">see page 715</a> )	Thickness definition.
	IRobotThicknessHomoData ( <a href="#">see page 718</a> )	Definition of thickness parameters of the I_TT_HOMOGENEOUS type.
	IRobotThicknessOrthoData ( <a href="#">see page 724</a> )	.
	IRobotThicknessMatrix ( <a href="#">see page 733</a> )	Object provides access to stiffness matrix values for orthotropic thicknesses.
	IRobotSolidPropertiesData ( <a href="#">see page 735</a> )	Solid properties.

### II.5.1.1 Linear releases

Available since version 2.5.

### Enumerations

	Name	Description
	IRobotLinearReleaseDefinitionType ( <a href="#">see page 701</a> )	Available methods of release definition for one direction. .

### Interfaces

	Name	Description
	IRobotLinearReleaseData ( <a href="#">see page 698</a> )	Linear release between panels.
	IRobotLinearReleaseServer ( <a href="#">see page 702</a> )	Server allowing definition of linear releases between panels. Moreover, it provides access to the list of objects describing all the releases defined. .
	IRobotLinearReleaseDef ( <a href="#">see page 707</a> )	Object ( <a href="#">see page 709</a> ) describing definition parameters of the linear release between panels. The release is defined for the selected object component on the indicated edge. The edge may belong to object other than the released component.
	IRobotLinearReleaseDefList ( <a href="#">see page 710</a> )	List of definitions of linear releases.

## II.5.1.1.1 IRobotLinearReleaseData

### Class Hierarchy

#### C++

```
interface IRobotLinearReleaseData : IDispatch;
```

#### C#

```
public interface IRobotLinearReleaseData;
```

### Visual Basic

```
Public Interface IRobotLinearReleaseData
```

### Description

Linear release between panels.

### Version

Available since version 2.5.

## II.5.1.1.1.1 IRobotLinearReleaseData Members

The following tables list the members exposed by IRobotLinearReleaseData.

### Public Fields

	Name	Description
◆	HX (see page 699)	HX elastic coefficient for the elastic release in the RX (see page 700) direction.
◆	KX (see page 699)	KX elastic coefficient for the elastic release in the UX (see page 700) direction.
◆	KY (see page 699)	KY elastic coefficient for the elastic release in the UY (see page 701) direction.
◆	KZ (see page 700)	KZ elastic coefficient for the elastic release in the UZ (see page 701) direction.
◆	RX (see page 700)	Definition of the release (RX direction).
◆	UX (see page 700)	Definition of the release (UX direction).
◆	UY (see page 701)	Definition of the release (UY direction).
◆	UZ (see page 701)	Definition of the release (UZ direction).

## II.5.1.1.1.2 IRobotLinearReleaseData Fields

The fields of the IRobotLinearReleaseData class are listed here.

### Public Fields

	Name	Description
◆	HX (see page 699)	HX elastic coefficient for the elastic release in the RX (see page 700) direction.
◆	KX (see page 699)	KX elastic coefficient for the elastic release in the UX (see page 700) direction.
◆	KY (see page 699)	KY elastic coefficient for the elastic release in the UY (see page 701) direction.
◆	KZ (see page 700)	KZ elastic coefficient for the elastic release in the UZ (see page 701) direction.
◆	RX (see page 700)	Definition of the release (RX direction).
◆	UX (see page 700)	Definition of the release (UX direction).
◆	UY (see page 701)	Definition of the release (UY direction).

	UZ (see page 701)	Definition of the release (UZ direction).
--	-------------------	-------------------------------------------

### II.5.1.1.1.2.1 HX

#### C++

```
HRESULT get_HX(double* );
HRESULT put_HX(double);
```

#### C#

```
public double HX { get; set; }
```

#### Visual Basic

```
Public HX As Double
```

#### Description

HX elastic coefficient for the elastic release in the RX (see page 700) direction.

#### Version

Available since version 2.5.

### II.5.1.1.1.2.2 KX

#### C++

```
HRESULT get_KX(double* );
HRESULT put_KX(double);
```

#### C#

```
public double KX { get; set; }
```

#### Visual Basic

```
Public KX As Double
```

#### Description

KX elastic coefficient for the elastic release in the UX (see page 700) direction.

#### Version

Available since version 2.5.

### II.5.1.1.1.2.3 KY

#### C++

```
HRESULT get_KY(double* );
HRESULT put_KY(double);
```

#### C#

```
public double KY { get; set; }
```

#### Visual Basic

```
Public KY As Double
```

#### Description

KY elastic coefficient for the elastic release in the UY (see page 701) direction.

#### Version

Available since version 2.5.

#### II.5.1.1.2.4 KZ

##### C++

```
HRESULT get_KZ(double*);  
HRESULT put_KZ(double);
```

##### C#

```
public double KZ { get; set; }
```

##### Visual Basic

```
Public KZ As Double
```

##### Description

KZ elastic coefficient for the elastic release in the UZ (see page 701) direction.

##### Version

Available since version 2.5.

#### II.5.1.1.2.5 RX

##### C++

```
HRESULT get_RX(IRobotLinearReleaseDefinitionType*);  
HRESULT put_RX(IRobotLinearReleaseDefinitionType);
```

##### C#

```
public IRobotLinearReleaseDefinitionType RX { get; set; }
```

##### Visual Basic

```
Public RX As IRobotLinearReleaseDefinitionType
```

##### Description

Definition of the release (RX direction).

##### Version

Available since version 2.5.

#### II.5.1.1.2.6 UX

##### C++

```
HRESULT get_UX(IRobotLinearReleaseDefinitionType*);  
HRESULT put_UX(IRobotLinearReleaseDefinitionType);
```

##### C#

```
public IRobotLinearReleaseDefinitionType UX { get; set; }
```

##### Visual Basic

```
Public UX As IRobotLinearReleaseDefinitionType
```

##### Description

Definition of the release (UX direction).

##### Version

Available since version 2.5.

### II.5.1.1.2.7 UY

#### C++

```
HRESULT get_UY(IRobotLinearReleaseDefinitionType* );
HRESULT put_UY(IRobotLinearReleaseDefinitionType);
```

#### C#

```
public IRobotLinearReleaseDefinitionType UY { get; set; }
```

#### Visual Basic

```
Public UY As IRobotLinearReleaseDefinitionType
```

#### Description

Definition of the release (UY direction).

#### Version

Available since version 2.5.

### II.5.1.1.2.8 UZ

#### C++

```
HRESULT get_UZ(IRobotLinearReleaseDefinitionType* );
HRESULT put_UZ(IRobotLinearReleaseDefinitionType);
```

#### C#

```
public IRobotLinearReleaseDefinitionType UZ { get; set; }
```

#### Visual Basic

```
Public UZ As IRobotLinearReleaseDefinitionType
```

#### Description

Definition of the release (UZ direction).

#### Version

Available since version 2.5.

## II.5.1.1.2 IRobotLinearReleaseDefinitionType

#### C++

```
enum IRobotLinearReleaseDefinitionType;
```

#### C#

```
public enum IRobotLinearReleaseDefinitionType;
```

#### Visual Basic

```
Public Enum IRobotLinearReleaseDefinitionType
```

#### Members

Members	Description
I_LRDT_NONE = 0	No release for the selected direction. Available since version 2.5.
I_LRDT_RELEASED = 1	Release definition for the selected direction. Available since version 2.5.
I_LRDT_MINUS = 2	Release definition in the direction opposite to the local system axis on the panel edge. Available since version 2.5.

I_LRDT_PLUS = 3	Release definition in the direction compatible with the local system axis on the panel edge . Available since version 2.5.
-----------------	-------------------------------------------------------------------------------------------------------------------------------

**Description**

Available methods of release definition for one direction. .

**Version**

Available since version 2.5.

**II.5.1.1.3 IRobotLinearReleaseServer****Class Hierarchy****C++**

```
interface IRobotLinearReleaseServer : IDispatch;
```

**C#**

```
public interface IRobotLinearReleaseServer;
```

**Visual Basic**

```
Public Interface IRobotLinearReleaseServer
```

**Description**

Server allowing definition of linear releases between panels. Moreover, it provides access to the list of objects describing all the releases defined. .

**Version**

Available since version 2.5.

**II.5.1.1.3.1 IRobotLinearReleaseServer Members**

The following tables list the members exposed by IRobotLinearReleaseServer.

**Public Fields**

	Name	Description
◆	Count ( <a href="#">see page 703</a> )	Number of defined releases.

**Public Methods**

	Name	Description
◆	Find ( <a href="#">see page 704</a> )	Function returns index of the definition describing the release between the indicated objects. If no release has been defined between the indicated objects, then zero value is returned. .
◆	FindEdge ( <a href="#">see page 704</a> )	Function returns index of the first located linear release definition on the indicated edge. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located on the indicated edge, then zero value is returned. .
◆	FindLabel ( <a href="#">see page 705</a> )	Function returns index of the first located definition of the linear release with the indicated name. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located, then zero value is returned. .
◆	FindObject ( <a href="#">see page 705</a> )	Function returns index of the first located linear release definition for the specified object component. The list of the defined linear releases is searched beginning with the indicated start index. If the linear release is not located for the indicated object component, then zero value is returned. .

	Get ( <a href="#">see page 705</a> )	Function returns the objects containing parameters of the linear release definition. By activating this function with the successive indexes from 1 to Count ( <a href="#">see page 703</a> ), the user may obtain information about all the defined linear releases between panels. .
	GetLabel ( <a href="#">see page 706</a> )	Function returns label describing parameters of the indicated linear release.
	Remove ( <a href="#">see page 706</a> )	Function deletes the linear release whose method of definition describes the object with the indicated index on the list of all the releases defined. .
	Set ( <a href="#">see page 706</a> )	Function defines a linear release for the selected object component on the indicated edge. The edge may belong to object other than the released component. Function returns value different from zero (True) if the release is defined. .
	SetMany ( <a href="#">see page 707</a> )	Function defines linear releases according to the specified list.

### II.5.1.1.3.2 IRobotLinearReleaseServer Fields

The fields of the IRobotLinearReleaseServer class are listed here.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 703</a> )	Number of defined releases.

### II.5.1.1.3.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of defined releases.

#### Version

Available since version 2.5.

### II.5.1.1.3.3 IRobotLinearReleaseServer Methods

The methods of the IRobotLinearReleaseServer class are listed here.

#### Public Methods

	Name	Description
	Find ( <a href="#">see page 704</a> )	Function returns index of the definition describing the release between the indicated objects. If no release has been defined between the indicated objects, then zero value is returned. .
	FindEdge ( <a href="#">see page 704</a> )	Function returns index of the first located linear release definition on the indicated edge. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located on the indicated edge, then zero value is returned. .
	FindLabel ( <a href="#">see page 705</a> )	Function returns index of the first located definition of the linear release with the indicated name. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located, then zero value is returned. .

	FindObject (see page 705)	Function returns index of the first located linear release definition for the specified object component. The list of the defined linear releases is searched beginning with the indicated start index. If the linear release is not located for the indicated object component, then zero value is returned..
	Get (see page 705)	Function returns the objects containing parameters of the linear release definition. By activating this function with the successive indexes from 1 to Count (see page 703), the user may obtain information about all the defined linear releases between panels. .
	GetLabel (see page 706)	Function returns label describing parameters of the indicated linear release.
	Remove (see page 706)	Function deletes the linear release whose method of definition describes the object with the indicated index on the list of all the releases defined. .
	Set (see page 706)	Function defines a linear release for the selected object component on the indicated edge. The edge may belong to object other than the released component. Function returns value different from zero (True) if the release is defined. .
	SetMany (see page 707)	Function defines linear releases according to the specified list.

### II.5.1.1.3.3.1 Find

#### C++

```
HRESULT Find(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx, long* ret);
```

#### C#

```
public long Find(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx);
```

#### Visual Basic

```
Public Function Find(_edge_obj_num As long, _edge_idx As long, _obj_num As long, _part_idx As long) As long
```

#### Description

Function returns index of the definition describing the release between the indicated objects. If no release has been defined between the indicated objects, then zero value is returned. .

#### Version

Available since version 2.5.

### II.5.1.1.3.3.2 FindEdge

#### C++

```
HRESULT FindEdge(long _edge_obj, long _edge_idx, long _start_def_idx = 1, long* ret);
```

#### C#

```
public long FindEdge(long _edge_obj, long _edge_idx, long _start_def_idx = 1);
```

#### Visual Basic

```
Public Function FindEdge(_edge_obj As long, _edge_idx As long, Optional _start_def_idx As long = 1) As long
```

#### Description

Function returns index of the first located linear release definition on the indicated edge. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located on the indicated edge, then zero value is returned. .

#### Version

Available since version 2.5.

### II.5.1.1.3.3.3 FindLabel

#### C++

```
HRESULT FindLabel(BSTR _label_name, long _start_def_idx = 1, long* ret);
```

#### C#

```
public long FindLabel(String _label_name, long _start_def_idx = 1);
```

#### Visual Basic

```
Public Function FindLabel(_label_name As String, Optional _start_def_idx As long = 1) As long
```

#### Description

Function returns index of the first located definition of the linear release with the indicated name. The list of the defined linear releases is searched beginning with the indicated start index. If the release is not located, then zero value is returned. .

#### Version

Available since version 2.5.

### II.5.1.1.3.3.4 FindObject

#### C++

```
HRESULT FindObject(long _obj_num, long _part_idx, long _start_def_idx = 1, long* ret);
```

#### C#

```
public long FindObject(long _obj_num, long _part_idx, long _start_def_idx = 1);
```

#### Visual Basic

```
Public Function FindObject(_obj_num As long, _part_idx As long, Optional _start_def_idx As long = 1) As long
```

#### Description

Function returns index of the first located linear release definition for the specified object component. The list of the defined linear releases is searched beginning with the indicated start index. If the linear release is not located for the indicated object component, then zero value is returned. .

#### Version

Available since version 2.5.

### II.5.1.1.3.3.5 Get

#### C++

```
HRESULT Get(long _def_idx, IRobotLinearReleaseDef** ret);
```

#### C#

```
public IRobotLinearReleaseDef Get(long _def_idx);
```

#### Visual Basic

```
Public Function Get(_def_idx As long) As IRobotLinearReleaseDef
```

#### Description

Function returns the objects containing parameters of the linear release definition. By activating this function with the successive indexes from 1 to Count (see page 703), the user may obtain information about all the defined linear releases between panels. .

**Version**

Available since version 2.5.

**II.5.1.1.3.3.6 GetLabel****C++**

```
HRESULT GetLabel(long _def_idx, IRobotLabel** ret);
```

**C#**

```
public IRobotLabel GetLabel(long _def_idx);
```

**Visual Basic**

```
Public Function GetLabel(_def_idx As long) As IRobotLabel
```

**Description**

Function returns label describing parameters of the indicated linear release.

**Version**

Available since version 2.5.

**II.5.1.1.3.3.7 Remove****C++**

```
HRESULT Remove(long _def_idx);
```

**C#**

```
public void Remove(long _def_idx);
```

**Visual Basic**

```
Public Sub Remove(_def_idx As long)
```

**Description**

Function deletes the linear release whose method of definition describes the object with the indicated index on the list of all the releases defined.

**Version**

Available since version 2.5.

**II.5.1.1.3.3.8 Set****C++**

```
HRESULT Set(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx, BSTR
_label_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Set(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx, String
_label_name);
```

**Visual Basic**

```
Public Function Set(_edge_obj_num As long, _edge_idx As long, _obj_num As long, _part_idx
As long, _label_name As String) As Boolean
```

**Description**

Function defines a linear release for the selected object component on the indicated edge. The edge may belong to object other than the released component. Function returns value different from zero (True) if the release is defined.

**Version**

Available since version 2.5.

**II.5.1.1.3.9 SetMany****C++**

```
HRESULT SetMany(IRobotLinearReleaseDefList* _deflist);
```

**C#**

```
public void SetMany(IRobotLinearReleaseDefList _deflist);
```

**Visual Basic**

```
Public Sub SetMany(ByRef _deflist As IRobotLinearReleaseDefList)
```

**Description**

Function defines linear releases according to the specified list.

**Version**

Available since version 8.

**II.5.1.1.4 IRobotLinearReleaseDef****Class Hierarchy****C++**

```
interface IRobotLinearReleaseDef : IDispatch;
```

**C#**

```
public interface IRobotLinearReleaseDef;
```

**Visual Basic**

```
Public Interface IRobotLinearReleaseDef
```

**Description**

Object (see page 709) describing definition parameters of the linear release between panels. The release is defined for the selected object component on the indicated edge. The edge may belong to object other than the released component.

**Version**

Available since version 2.5.

**II.5.1.1.4.1 IRobotLinearReleaseDef Members**

The following tables list the members exposed by IRobotLinearReleaseDef.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Edgeldx (see page 708)	Index of the edge in the object.
◆	EdgeObject (see page 708)	Number of the object including the edge along which the release is defined.
◆	LabelName (see page 709)	Name of the label storing release parameters.
◆	Object (see page 709)	Number of the object for the component of which the release is defined .
◆	PartIdx (see page 709)	Index of the object component for which the release is defined .

### II.5.1.1.4.2 IRobotLinearReleaseDef Fields

The fields of the IRobotLinearReleaseDef class are listed here.

#### Public Fields

	Name	Description
◆	EdgIdx ( [ see page 708) )	Index of the edge in the object.
◆	EdgeObject ( [ see page 708) )	Number of the object including the edge along which the release is defined.
◆	LabelName ( [ see page 709) )	Name of the label storing release parameters.
◆	Object ( [ see page 709) )	Number of the object for the component of which the release is defined .
◆	PartIdx ( [ see page 709) )	Index of the object component for which the release is defined .

### II.5.1.1.4.2.1 EdgIdx

#### C++

```
HRESULT get_EdgeIdx(long* );
HRESULT put_EdgeIdx(long);
```

#### C#

```
public long EdgeIdx { get; set; }
```

#### Visual Basic

```
Public EdgeIdx As long
```

#### Description

Index of the edge in the object.

#### Version

Available since version 2.5.

### II.5.1.1.4.2.2 EdgeObject

#### C++

```
HRESULT get_EdgeObject(long* );
HRESULT put_EdgeObject(long);
```

#### C#

```
public long EdgeObject { get; set; }
```

#### Visual Basic

```
Public EdgeObject As long
```

#### Description

Number of the object including the edge along which the release is defined.

#### Version

Available since version 2.5.

### II.5.1.1.4.2.3 LabelName

#### C++

```
HRESULT get_LabelName(BSTR* );
HRESULT put_LabelName(BSTR);
```

#### C#

```
public String LabelName { get; set; }
```

**Visual Basic**

```
Public LabelName As String
```

**Description**

Name of the label storing release parameters.

**Version**

Available since version 2.5.

**II.5.1.1.4.2.4 Object****C++**

```
HRESULT get_Object(long* );
HRESULT put_Object(long);
```

**C#**

```
public long Object { get; set; }
```

**Visual Basic**

```
Public Object As long
```

**Description**

Number of the object for the component of which the release is defined .

**Version**

Available since version 2.5.

**II.5.1.1.4.2.5 PartIdx****C++**

```
HRESULT get_PartIdx(long* );
HRESULT put_PartIdx(long);
```

**C#**

```
public long PartIdx { get; set; }
```

**Visual Basic**

```
Public PartIdx As long
```

**Description**

Index of the object component for which the release is defined .

**Version**

Available since version 2.5.

**II.5.1.1.5 IRobotLinearReleaseDefList****Class Hierarchy****C++**

```
interface IRobotLinearReleaseDefList : IDispatch;
```

**C#**

```
public interface IRobotLinearReleaseDefList;
```

**Visual Basic**

```
Public Interface IRobotLinearReleaseDefList
```

## Description

List of definitions of linear releases.

## Version

Available since version 8.

### II.5.1.1.5.1 IRobotLinearReleaseDefList Members

The following tables list the members exposed by IRobotLinearReleaseDefList.

#### Public Fields

	Name	Description
◆	Count (see page 710)	

#### Public Methods

	Name	Description
◆	Add (see page 711)	Function adds a linear release definition to the list. Function returns an index of the definition added to the list.
◆	Clear (see page 711)	Function deletes all elements from the list.
◆	Get (see page 711)	Function makes available a linear release definition of the specified index.
◆	Remove (see page 712)	Function deletes a linear release definition of the specified index from the list.

### II.5.1.1.5.2 IRobotLinearReleaseDefList Fields

The fields of the IRobotLinearReleaseDefList class are listed here.

#### Public Fields

	Name	Description
◆	Count (see page 710)	

#### II.5.1.1.5.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As long
```

## Version

Available since version 8.

### II.5.1.1.5.3 IRobotLinearReleaseDefList Methods

The methods of the IRobotLinearReleaseDefList class are listed here.

#### Public Methods

	Name	Description
◆	Add (see page 711)	Function adds a linear release definition to the list. Function returns an index of the definition added to the list.
◆	Clear (see page 711)	Function deletes all elements from the list.
◆	Get (see page 711)	Function makes available a linear release definition of the specified index.

	Remove (see page 712)	Function deletes a linear release definition of the specified index from the list.
-----------------------------------------------------------------------------------	-----------------------	------------------------------------------------------------------------------------

### II.5.1.1.5.3.1 Add

#### C++

```
HRESULT Add(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx, BSTR _label_name, long* ret);
```

#### C#

```
public long Add(long _edge_obj_num, long _edge_idx, long _obj_num, long _part_idx, String _label_name);
```

#### Visual Basic

```
Public Function Add(_edge_obj_num As long, _edge_idx As long, _obj_num As long, _part_idx As long, _label_name As String) As long
```

#### Description

Function adds a linear release definition to the list. Function returns an index of the definition added to the list.

#### Version

Available since version 8.

### II.5.1.1.5.3.2 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

Function deletes all elements from the list.

#### Version

Available since version 8.

### II.5.1.1.5.3.3 Get

#### C++

```
HRESULT Get(long _idx, long* _edge_obj_num, long* _edge_idx, long* _obj_num, long* _part_idx, BSTR _label_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool Get(long _idx, long* _edge_obj_num, long* _edge_idx, long* _obj_num, long* _part_idx, String _label_name);
```

#### Visual Basic

```
Public Function Get(_idx As long, ByRef _edge_obj_num As long*, ByRef _edge_idx As long*, ByRef _obj_num As long*, ByRef _part_idx As long*, ByRef _label_name As String) As Boolean
```

#### Description

Function makes available a linear release definition of the specified index.

**Version**

Available since version 8.

**II.5.1.1.5.3.4 Remove****C++**

```
HRESULT Remove(long _idx);
```

**C#**

```
public void Remove(long _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As long)
```

**Description**

Function deletes a linear release definition of the specified index from the list.

**Version**

Available since version 8.

**II.5.1.2 Panel calculation model**

Available since version 9.7.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotPanelCalcModelFinElemType ( <a href="#">see page 712</a> )	
	IRobotPanelCalcModelLoadTransfer ( <a href="#">see page 713</a> )	
	IRobotPanelCalcModelDiaphragm ( <a href="#">see page 713</a> )	

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotPanelCalcModelData ( <a href="#">see page 714</a> )	Define a panel calculation model.

**II.5.1.2.1 IRobotPanelCalcModelFinElemType****C++**

```
enum IRobotPanelCalcModelFinElemType;
```

**C#**

```
public enum IRobotPanelCalcModelFinElemType;
```

**Visual Basic**

```
Public Enum IRobotPanelCalcModelFinElemType
```

**Members**

<b>Members</b>	<b>Description</b>
I_PCMFET_NONE = 0	Available since version 9.7.
I_PCMFET_SHELL = 1	Available since version 9.7.

**Version**

Available since version 9.7.

**II.5.1.2.2 IRobotPanelCalcModelLoadTransfer****C++**

```
enum IRobotPanelCalcModelLoadTransfer;
```

**C#**

```
public enum IRobotPanelCalcModelLoadTransfer;
```

**Visual Basic**

```
Public Enum IRobotPanelCalcModelLoadTransfer
```

**Members**

Members	Description
I_PCMLT_ANALYTICALFINITE_ELEMENTS = 1	Available since version 9.7.
I_PCMLT_SIMPLIFIEDTWO_WAY = 2	Available since version 9.7.
I_PCMLT_SIMPLIFIEDONE_WAY = 3	Available since version 9.7.

**Version**

Available since version 9.7.

**II.5.1.2.3 IRobotPanelCalcModelDiaphragm****C++**

```
enum IRobotPanelCalcModelDiaphragm;
```

**C#**

```
public enum IRobotPanelCalcModelDiaphragm;
```

**Visual Basic**

```
Public Enum IRobotPanelCalcModelDiaphragm
```

**Members**

Members	Description
I_PCMD_NONE = 0	Available since version 9.7.
I_PCMD_RIGID = 1	Available since version 9.7.
I_PCMD_FLEXIBLE = 2	Available since version 9.7.

**Version**

Available since version 9.7.

**II.5.1.2.4 IRobotPanelCalcModelData****Class Hierarchy****C++**

```
interface IRobotPanelCalcModelData : IDispatch;
```

**C#**

```
public interface IRobotPanelCalcModelData;
```

## Visual Basic

```
Public Interface IRobotPanelCalcModelData
```

### Description

Define a panel calculation model.

### Version

Available since version 9.7.

#### II.5.1.2.4.1 IRobotPanelCalcModelData Members

The following tables list the members exposed by IRobotPanelCalcModelData.

### Public Fields

	Name	Description
◆	Diaphragm (see page 714)	
◆	FinElemType (see page 715)	
◆	LoadTransfer (see page 715)	

#### II.5.1.2.4.2 IRobotPanelCalcModelData Fields

The fields of the IRobotPanelCalcModelData class are listed here.

### Public Fields

	Name	Description
◆	Diaphragm (see page 714)	
◆	FinElemType (see page 715)	
◆	LoadTransfer (see page 715)	

#### II.5.1.2.4.2.1 Diaphragm

### C++

```
HRESULT get_Diaphragm(IRobotPanelCalcModelDiaphragm* );
HRESULT put_Diaphragm(IRobotPanelCalcModelDiaphragm);
```

### C#

```
public IRobotPanelCalcModelDiaphragm Diaphragm { get; set; }
```

### Visual Basic

```
Public Diaphragm As IRobotPanelCalcModelDiaphragm
```

### Version

Available since version 9.7.

#### II.5.1.2.4.2.2 FinElemType

### C++

```
HRESULT get_FinElemType(IRobotPanelCalcModelFinElemType* );
HRESULT put_FinElemType(IRobotPanelCalcModelFinElemType);
```

### C#

```
public IRobotPanelCalcModelFinElemType FinElemType { get; set; }
```

### Visual Basic

```
Public FinElemType As IRobotPanelCalcModelFinElemType
```

**Version**

Available since version 9.7.

**II.5.1.2.4.2.3 LoadTransfer****C++**

```
HRESULT get_LoadTransfer(IRobotPanelCalcModelLoadTransfer* );
HRESULT put_LoadTransfer(IRobotPanelCalcModelLoadTransfer);
```

**C#**

```
public IRobotPanelCalcModelLoadTransfer LoadTransfer { get; set; }
```

**Visual Basic**

```
Public LoadTransfer As IRobotPanelCalcModelLoadTransfer
```

**Version**

Available since version 9.7.

**II.5.1.3 IRobotThicknessData****Class Hierarchy****C++**

```
interface IRobotThicknessData : IDispatch;
```

**C#**

```
public interface IRobotThicknessData;
```

**Visual Basic**

```
Public Interface IRobotThicknessData
```

**Description**

Thickness definition.

**II.5.1.3.1 IRobotThicknessData Members**

The following tables list the members exposed by IRobotThicknessData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Data (see page 716)	Set of data that are specific for the selected thickness type; parameters describing thickness of the I_TT_HOMOGENEOUS type are defined by means of RobotThicknessHomoData, whereas for the I_TT_ORTHOTROPIC type RobotThicknessOrthoData is defined Available since version 1.7.
❖	ElasticFoundation (see page 716)	Available since version 1.7.
❖	MaterialName (see page 717)	Name of the material connected with thickness Available since version 2.0.
❖	ThicknessType (see page 717)	Available since version 1.7.
❖	Uplift (see page 717)	Available since version 1.7.

**II.5.1.3.2 IRobotThicknessData Fields**

The fields of the IRobotThicknessData class are listed here.

## Public Fields

	Name	Description
◆	Data (see page 716)	Set of data that are specific for the selected thickness type; parameters describing thickness of the I_TT_HOMOGENEOUS type are defined by means of RobotThicknessHomoData, whereas for the I_TT_ORTHOTROPIC type RobotThicknessOrthoData is defined Available since version 1.7.
◆	ElasticFoundation (see page 716)	Available since version 1.7.
◆	MaterialName (see page 717)	Name of the material connected with thickness Available since version 2.0.
◆	ThicknessType (see page 717)	Available since version 1.7.
◆	Uplift (see page 717)	Available since version 1.7.

### II.5.1.3.2.1 Data

#### C++

```
HRESULT get_Data(Object*);
```

#### C#

```
public Object Data { get; }
```

#### Visual Basic

```
Public ReadOnly Data As Object
```

#### Description

Set of data that are specific for the selected thickness type; parameters describing thickness of the I\_TT\_HOMOGENEOUS type are defined by means of RobotThicknessHomoData, whereas for the I\_TT\_ORTHOTROPIC type RobotThicknessOrthoData is defined Available since version 1.7.

### II.5.1.3.2.2 ElasticFoundation

#### C++

```
HRESULT get_ElasticFoundation(double*);  
HRESULT put_ElasticFoundation(double);
```

#### C#

```
public double ElasticFoundation { get; set; }
```

#### Visual Basic

```
Public ElasticFoundation As Double
```

#### Description

Available since version 1.7.

### II.5.1.3.2.3 MaterialName

#### C++

```
HRESULT get_MaterialName(BSTR*);  
HRESULT put_MaterialName(BSTR);
```

#### C#

```
public String MaterialName { get; set; }
```

#### Visual Basic

```
Public MaterialName As String
```

**Description**

Name of the material connected with thickness Available since version 2.0.

**II.5.1.3.2.4 ThicknessType****C++**

```
HRESULT get_ThicknessType(IRobotThicknessType* );
HRESULT put_ThicknessType(IRobotThicknessType);
```

**C#**

```
public IRobotThicknessType ThicknessType { get; set; }
```

**Visual Basic**

```
Public ThicknessType As IRobotThicknessType
```

**Description**

Available since version 1.7.

**II.5.1.3.2.5 Uplift****C++**

```
HRESULT get_Uplift(IRobotThicknessUpliftType* );
HRESULT put_Uplift(IRobotThicknessUpliftType);
```

**C#**

```
public IRobotThicknessUpliftType Uplift { get; set; }
```

**Visual Basic**

```
Public Uplift As IRobotThicknessUpliftType
```

**Description**

Available since version 1.7.

**II.5.1.4 IRobotThicknessType****C++**

```
enum IRobotThicknessType;
```

**C#**

```
public enum IRobotThicknessType;
```

**Visual Basic**

```
Public Enum IRobotThicknessType
```

**Members**

Members	Description
I_TT_HOMOGENEOUS = 0	Available since version 1.7.
I_TT_ORTHOTROPIC = 1	Available since version 1.7.

**II.5.1.5 IRobotThicknessUpliftType****C++**

```
enum IRobotThicknessUpliftType;
```

**C#**

```
public enum IRobotThicknessUpliftType;
```

**Visual Basic**

```
Public Enum IRobotThicknessUpliftType
```

**Members**

Members	Description
I_TUT_NONE = 0	Available since version 1.7.
I_TUT_MINUS = 1	Available since version 1.7.
I_TUT_PLUS = 2	Available since version 1.7.

**II.5.1.6 IRobotThicknessHomoData****Class Hierarchy****C++**

```
interface IRobotThicknessHomoData : IDispatch;
```

**C#**

```
public interface IRobotThicknessHomoData;
```

**Visual Basic**

```
Public Interface IRobotThicknessHomoData
```

**Description**

Definition of thickness parameters of the I\_TT\_HOMOGENEOUS type.

**II.5.1.6.1 IRobotThicknessHomoData Members**

The following tables list the members exposed by IRobotThicknessHomoData.

**Public Fields**

	Name	Description
◆	Thick1 (↗ see page 719)	Thickness value in the P1 point (set only for variable thickness) Available since version 1.7.
◆	Thick2 (↗ see page 719)	Thickness value in the P2 point (set only for variable thickness) Available since version 1.7.
◆	Thick3 (↗ see page 720)	Thickness value in the P3 point (set only for variable thickness) Available since version 1.7.
◆	ThickConst (↗ see page 720)	Thickness value (if the thickness is uniform for the whole slab) Available since version 1.7.
◆	Type (↗ see page 720)	Available since version 1.7.

**Public Methods**

	Name	Description
♫	GetP1 (↗ see page 721)	Available since version 1.7.
♫	GetP2 (↗ see page 721)	Available since version 1.7.
♫	GetP3 (↗ see page 721)	Available since version 1.7.
♫	GetReduction (↗ see page 722)	Function returns True value if a reduction coefficient of the moment of inertia has been set. The coefficient value will be entered to the specified parameter.
♫	SetP1 (↗ see page 722)	Available since version 1.7.
♫	SetP2 (↗ see page 722)	Available since version 1.7.

	SetP3 (see page 722)	Available since version 1.7.
	SetReduction (see page 723)	Function sets a coefficient of reduction of the moment of inertia.

### II.5.1.6.2 IRobotThicknessHomoData Fields

The fields of the IRobotThicknessHomoData class are listed here.

#### Public Fields

	Name	Description
	Thick1 (see page 719)	Thickness value in the P1 point (set only for variable thickness) Available since version 1.7.
	Thick2 (see page 719)	Thickness value in the P2 point (set only for variable thickness) Available since version 1.7.
	Thick3 (see page 720)	Thickness value in the P3 point (set only for variable thickness) Available since version 1.7.
	ThickConst (see page 720)	Thickness value (if the thickness is uniform for the whole slab) Available since version 1.7.
	Type (see page 720)	Available since version 1.7.

### II.5.1.6.2.1 Thick1

#### C++

```
HRESULT get_Thick1(double*);  
HRESULT put_Thick1(double);
```

#### C#

```
public double Thick1 { get; set; }
```

#### Visual Basic

```
Public Thick1 As Double
```

#### Description

Thickness value in the P1 point (set only for variable thickness) Available since version 1.7.

### II.5.1.6.2.2 Thick2

#### C++

```
HRESULT get_Thick2(double*);  
HRESULT put_Thick2(double);
```

#### C#

```
public double Thick2 { get; set; }
```

#### Visual Basic

```
Public Thick2 As Double
```

#### Description

Thickness value in the P2 point (set only for variable thickness) Available since version 1.7.

### II.5.1.6.2.3 Thick3

#### C++

```
HRESULT get_Thick3(double*);  
HRESULT put_Thick3(double);
```

#### C#

```
public double Thick3 { get; set; }
```

**Visual Basic**

```
Public Thick3 As double
```

**Description**

Thickness value in the P3 point (set only for variable thickness) Available since version 1.7.

**II.5.1.6.2.4 ThickConst****C++**

```
HRESULT get_ThickConst(double* );
HRESULT put_ThickConst(double);
```

**C#**

```
public double ThickConst { get; set; }
```

**Visual Basic**

```
Public ThickConst As double
```

**Description**

Thickness value (if the thickness is uniform for the whole slab) Available since version 1.7.

**II.5.1.6.2.5 Type****C++**

```
HRESULT get_Type(IRobotThicknessHomoType* );
HRESULT put_Type(IRobotThicknessHomoType);
```

**C#**

```
public IRobotThicknessHomoType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotThicknessHomoType
```

**Description**

Available since version 1.7.

**II.5.1.6.3 IRobotThicknessHomoData Methods**

The methods of the IRobotThicknessHomoData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	GetP1 ( <a href="#">see page 721</a> )	Available since version 1.7.
≡	GetP2 ( <a href="#">see page 721</a> )	Available since version 1.7.
≡	GetP3 ( <a href="#">see page 721</a> )	Available since version 1.7.
≡	GetReduction ( <a href="#">see page 722</a> )	Function returns True value if a reduction coefficient of the moment of inertia has been set. The coefficient value will be entered to the specified parameter.
≡	SetP1 ( <a href="#">see page 722</a> )	Available since version 1.7.
≡	SetP2 ( <a href="#">see page 722</a> )	Available since version 1.7.
≡	SetP3 ( <a href="#">see page 722</a> )	Available since version 1.7.
≡	SetReduction ( <a href="#">see page 723</a> )	Function sets a coefficient of reduction of the moment of inertia.

### II.5.1.6.3.1 GetP1

**C++**

```
HRESULT GetP1(double* _x, double* _y, double* _z);
```

**C#**

```
public void GetP1(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Sub GetP1(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

**Description**

Available since version 1.7.

### II.5.1.6.3.2 GetP2

**C++**

```
HRESULT GetP2(double* _x, double* _y, double* _z);
```

**C#**

```
public void GetP2(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Sub GetP2(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

**Description**

Available since version 1.7.

### II.5.1.6.3.3 GetP3

**C++**

```
HRESULT GetP3(double* _x, double* _y, double* _z);
```

**C#**

```
public void GetP3(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Sub GetP3(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

**Description**

Available since version 1.7.

### II.5.1.6.3.4 GetReduction

**C++**

```
HRESULT GetReduction(double* _Ig_cf, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetReduction(double* _Ig_cf);
```

**Visual Basic**

```
Public Function GetReduction(ByRef _Ig_cf As double*) As Boolean
```

## Description

Function returns True value if a reduction coefficient of the moment of inertia has been set. The coefficient value will be entered to the specified parameter.

## Version

Available since version 8.

### II.5.1.6.3.5 SetP1

#### C++

```
HRESULT SetP1(double _x, double _y, double _z);
```

#### C#

```
public void SetP1(double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetP1(_x As double, _y As double, _z As double)
```

## Description

Available since version 1.7.

### II.5.1.6.3.6 SetP2

#### C++

```
HRESULT SetP2(double _x, double _y, double _z);
```

#### C#

```
public void SetP2(double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetP2(_x As double, _y As double, _z As double)
```

## Description

Available since version 1.7.

### II.5.1.6.3.7 SetP3

#### C++

```
HRESULT SetP3(double _x, double _y, double _z);
```

#### C#

```
public void SetP3(double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetP3(_x As double, _y As double, _z As double)
```

## Description

Available since version 1.7.

### II.5.1.6.3.8 SetReduction

#### C++

```
HRESULT SetReduction(VARIANT_BOOL _set, double _Ig_cf);
```

**C#**

```
public void SetReduction(bool _set, double _Ig_cf);
```

**Visual Basic**

```
Public Sub SetReduction(_set As Boolean, _Ig_cf As Double)
```

**Description**

Function sets a coefficient of reduction of the moment of inertia.

**Version**

Available since version 8.

**II.5.1.7 IRobotThicknessHomoType****C++**

```
enum IRobotThicknessHomoType;
```

**C#**

```
public enum IRobotThicknessHomoType;
```

**Visual Basic**

```
Public Enum IRobotThicknessHomoType
```

**Members**

Members	Description
I_THT_CONSTANT = 0	Available since version 1.7.
I_THT_VARIABLE_ALONG_LINE = 1	Available since version 1.7.
I_THT_VARIABLE_ON_PLANE = 2	Available since version 1.7.

**II.5.1.8 IRobotThicknessOrthoType****C++**

```
enum IRobotThicknessOrthoType;
```

**C#**

```
public enum IRobotThicknessOrthoType;
```

**Visual Basic**

```
Public Enum IRobotThicknessOrthoType
```

**Members**

Members	Description
I_TOT_ONE_SIDED_UNIDIR_RIBS = 1	Available since version 1.7.
I_TOT_DOUBLE_SIDED_UNIDIR_RIBS = 2	Available since version 1.7.
I_TOT_ONE_SIDED_BIDIR_RIBS = 3	Available since version 1.7.
I_TOT_UNIDIR_BOX_FLOOR = 4	Available since version 1.7.
I_TOT_BIDIR_BOX_FLOOR = 5	Available since version 1.7.
I_TOT_GRILLAGE = 6	Available since version 1.7.
I_TOT_SLAB_ON_TRAPEZOID_PLATE = 7	Available since version 1.7.
I_TOT_USER = 8	Geometrical thickness parameters should be described by means of the appropriate matrices Available since version 1.7.
I_TOT_MATERIAL = 9	Material orthotropy Available since version 1.7.

I_TOT_HOLLOW_CORE_SLAB = 15	Available since version 11.
I_TOT_SLAB_COMPOSED_WITH_TRAPEZOID_PLATE = 13	Available since version 16.2.

**Description****II.5.1.9 IRobotThicknessOrthoDirType****C++**

```
enum IRobotThicknessOrthoDirType;
```

**C#**

```
public enum IRobotThicknessOrthoDirType;
```

**Visual Basic**

```
Public Enum IRobotThicknessOrthoDirType
```

**Members**

Members	Description
I_TODT_DIR_X = 0	Available since version 1.7.
I_TODT_DIR_Y = 1	Available since version 1.7.
I_TODT_DIR_Z = 2	Available since version 1.7.
I_TODT_VECTOR = 3	Available since version 1.7.
I_TODT_AUTOMATIC = 4	Available since version 15.2.

**Description****II.5.1.10 IRobotThicknessOrthoData****Class Hierarchy****C++**

```
interface IRobotThicknessOrthoData : IDispatch;
```

**C#**

```
public interface IRobotThicknessOrthoData;
```

**Visual Basic**

```
Public Interface IRobotThicknessOrthoData
```

**Description****II.5.1.10.1 IRobotThicknessOrthoData Members**

The following tables list the members exposed by IRobotThicknessOrthoData.

**Public Fields**

	Name	Description
◆	A (see page 726)	Available since version 1.7.
◆	A1 (see page 726)	Available since version 1.7.

◆	A2 ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	B ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	B1 ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	DirType ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	DisregardBendStiffDirY ( <a href="#">see page 728</a> )	Disregard bending stiffness for Y direction.
◆	ES ( <a href="#">see page 728</a> )	.
◆	H ( <a href="#">see page 728</a> )	Available since version 1.7.
◆	H0 ( <a href="#">see page 729</a> )	Opening height.
◆	H1 ( <a href="#">see page 729</a> )	Available since version 1.7.
◆	H2 ( <a href="#">see page 729</a> )	Available since version 1.7.
◆	HA ( <a href="#">see page 729</a> )	Available since version 1.7.
◆	HB ( <a href="#">see page 730</a> )	Available since version 1.7.
◆	HC ( <a href="#">see page 730</a> )	Distance from lower edge to opening center.
◆	Matrix ( <a href="#">see page 730</a> )	Object providing access to stiffness matrix values Available since version 1.7.
◆	N1 ( <a href="#">see page 731</a> )	First stiffness coefficient for material orthotropy Available since version 1.7.
◆	N2 ( <a href="#">see page 731</a> )	Second stiffness coefficient for material orthotropy Available since version 1.7.
◆	T ( <a href="#">see page 731</a> )	.
◆	Thick1 ( <a href="#">see page 731</a> )	Available since version 1.7.
◆	Thick2 ( <a href="#">see page 732</a> )	Available since version 1.7.
◆	Thick3 ( <a href="#">see page 732</a> )	Available since version 1.7.
◆	Type ( <a href="#">see page 732</a> )	Available since version 1.7.
◆	VS ( <a href="#">see page 732</a> )	.

## Public Methods

	Name	Description
◆	GetVector ( <a href="#">see page 733</a> )	Available since version 1.7.
◆	SetVector ( <a href="#">see page 733</a> )	Available since version 1.7.

## II.5.1.10.2 IRobotThicknessOrthoData Fields

The fields of the IRobotThicknessOrthoData class are listed here.

## Public Fields

	Name	Description
◆	A ( <a href="#">see page 726</a> )	Available since version 1.7.
◆	A1 ( <a href="#">see page 726</a> )	Available since version 1.7.
◆	A2 ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	B ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	B1 ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	DirType ( <a href="#">see page 727</a> )	Available since version 1.7.
◆	DisregardBendStiffDirY ( <a href="#">see page 728</a> )	Disregard bending stiffness for Y direction.
◆	ES ( <a href="#">see page 728</a> )	.
◆	H ( <a href="#">see page 728</a> )	Available since version 1.7.
◆	H0 ( <a href="#">see page 729</a> )	Opening height.
◆	H1 ( <a href="#">see page 729</a> )	Available since version 1.7.
◆	H2 ( <a href="#">see page 729</a> )	Available since version 1.7.
◆	HA ( <a href="#">see page 729</a> )	Available since version 1.7.

❖	HB ( [ see page 730)	Available since version 1.7.
❖	HC ( [ see page 730)	Distance from lower edge to opening center.
❖	Matrix ( [ see page 730)	Object providing access to stiffness matrix values Available since version 1.7.
❖	N1 ( [ see page 731)	First stiffness coefficient for material orthotropy Available since version 1.7.
❖	N2 ( [ see page 731)	Second stiffness coefficient for material orthotropy Available since version 1.7.
❖	T ( [ see page 731)	.
❖	Thick1 ( [ see page 731)	Available since version 1.7.
❖	Thick2 ( [ see page 732)	Available since version 1.7.
❖	Thick3 ( [ see page 732)	Available since version 1.7.
❖	Type ( [ see page 732)	Available since version 1.7.
❖	VS ( [ see page 732)	.

### II.5.1.10.2.1 A

#### C++

```
HRESULT get_A(double* );
HRESULT put_A(double);
```

#### C#

```
public double A { get; set; }
```

#### Visual Basic

```
Public A As double
```

#### Description

Available since version 1.7.

### II.5.1.10.2.2 A1

#### C++

```
HRESULT get_A1(double* );
HRESULT put_A1(double);
```

#### C#

```
public double A1 { get; set; }
```

#### Visual Basic

```
Public A1 As double
```

#### Description

Available since version 1.7.

### II.5.1.10.2.3 A2

#### C++

```
HRESULT get_A2(double* );
HRESULT put_A2(double);
```

#### C#

```
public double A2 { get; set; }
```

#### Visual Basic

```
Public A2 As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.4 B****C++**

```
HRESULT get_B(double*);  
HRESULT put_B(double);
```

**C#**

```
public double B { get; set; }
```

**Visual Basic**

```
Public B As Double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.5 B1****C++**

```
HRESULT get_B1(double*);  
HRESULT put_B1(double);
```

**C#**

```
public double B1 { get; set; }
```

**Visual Basic**

```
Public B1 As Double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.6 DirType****C++**

```
HRESULT get_DirType(IRobotThicknessOrthoDirType*);  
HRESULT put_DirType(IRobotThicknessOrthoDirType);
```

**C#**

```
public IRobotThicknessOrthoDirType DirType { get; set; }
```

**Visual Basic**

```
Public DirType As IRobotThicknessOrthoDirType
```

**Description**

Available since version 1.7.

**II.5.1.10.2.7 DisregardBendStiffDirY****C++**

```
HRESULT get_DisregardBendStiffDirY(VARIANT_BOOL*);  
HRESULT put_DisregardBendStiffDirY(VARIANT_BOOL);
```

**C#**

```
public bool DisregardBendStiffDirY { get; set; }
```

**Visual Basic**

```
Public DisregardBendStiffDirY As Boolean
```

**Description**

Disregard bending stiffness for Y direction.

**Version**

Available since version 11.

**II.5.1.10.2.8 ES****C++**

```
HRESULT get_ES(double* );
HRESULT put_ES(double);
```

**C#**

```
public double ES { get; set; }
```

**Visual Basic**

```
Public ES As Double
```

**Description****Version**

Available since version 16.2.

**II.5.1.10.2.9 H****C++**

```
HRESULT get_H(double* );
HRESULT put_H(double);
```

**C#**

```
public double H { get; set; }
```

**Visual Basic**

```
Public H As Double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.10 H0****C++**

```
HRESULT get_H0(double* );
HRESULT put_H0(double);
```

**C#**

```
public double H0 { get; set; }
```

**Visual Basic**

```
Public H0 As Double
```

**Description**

Opening height.

**Version**

Available since version 11.

**II.5.1.10.2.11 H1****C++**

```
HRESULT get_H1(double* );
HRESULT put_H1(double);
```

**C#**

```
public double H1 { get; set; }
```

**Visual Basic**

```
Public H1 As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.12 H2****C++**

```
HRESULT get_H2(double* );
HRESULT put_H2(double);
```

**C#**

```
public double H2 { get; set; }
```

**Visual Basic**

```
Public H2 As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.13 HA****C++**

```
HRESULT get_HA(double* );
HRESULT put_HA(double);
```

**C#**

```
public double HA { get; set; }
```

**Visual Basic**

```
Public HA As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.14 HB****C++**

```
HRESULT get_HB(double* );
HRESULT put_HB(double);
```

**C#**

```
public double HB { get; set; }
```

**Visual Basic**

```
Public HB As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.15 HC****C++**

```
HRESULT get_HC(double* );
HRESULT put_HC(double);
```

**C#**

```
public double HC { get; set; }
```

**Visual Basic**

```
Public HC As double
```

**Description**

Distance from lower edge to opening center.

**Version**

Available since version 11.

**II.5.1.10.2.16 Matrix****C++**

```
HRESULT get_Matrix(IRobotThicknessMatrix** );
```

**C#**

```
public IRobotThicknessMatrix Matrix { get; }
```

**Visual Basic**

```
Public ReadOnly Matrix As IRobotThicknessMatrix
```

**Description**

Object providing access to stiffness matrix values Available since version 1.7.

**II.5.1.10.2.17 N1****C++**

```
HRESULT get_N1(double* );
HRESULT put_N1(double);
```

**C#**

```
public double N1 { get; set; }
```

**Visual Basic**

```
Public N1 As double
```

**Description**

First stiffness coefficient for material orthotropy Available since version 1.7.

**II.5.1.10.2.18 N2****C++**

```
HRESULT get_N2(double* );
```

```
HRESULT put_N2(double);  
  
C#  
public double N2 { get; set; }
```

**Visual Basic**

```
Public N2 As double
```

**Description**

Second stiffness coefficient for material orthotropy Available since version 1.7.

**II.5.1.10.2.19 T****C++**

```
HRESULT get_T(double*);  
HRESULT put_T(double);
```

**C#**

```
public double T { get; set; }
```

**Visual Basic**

```
Public T As double
```

**Description****Version**

Available since version 16.2.

**II.5.1.10.2.20 Thick1****C++**

```
HRESULT get_Thick1(double*);  
HRESULT put_Thick1(double);
```

**C#**

```
public double Thick1 { get; set; }
```

**Visual Basic**

```
Public Thick1 As double
```

**Description**

Available since version 1.7.

**II.5.1.10.2.21 Thick2****C++**

```
HRESULT get_Thick2(double*);  
HRESULT put_Thick2(double);
```

**C#**

```
public double Thick2 { get; set; }
```

**Visual Basic**

```
Public Thick2 As double
```

**Description**

Available since version 1.7.

### II.5.1.10.2.22 Thick3

#### C++

```
HRESULT get_Thickness3(double* );
HRESULT put_Thickness3(double);
```

#### C#

```
public double Thickness3 { get; set; }
```

#### Visual Basic

```
Public Thickness3 As Double
```

#### Description

Available since version 1.7.

### II.5.1.10.2.23 Type

#### C++

```
HRESULT get_Type(IRobotThicknessOrthoType* );
HRESULT put_Type(IRobotThicknessOrthoType);
```

#### C#

```
public IRobotThicknessOrthoType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotThicknessOrthoType
```

#### Description

Available since version 1.7.

### II.5.1.10.2.24 VS

#### C++

```
HRESULT get_VS(double* );
HRESULT put_VS(double);
```

#### C#

```
public double VS { get; set; }
```

#### Visual Basic

```
Public VS As Double
```

#### Description

#### Version

Available since version 16.2.

### II.5.1.10.3 IRobotThicknessOrthoData Methods

The methods of the IRobotThicknessOrthoData class are listed here.

#### Public Methods

	Name	Description
	GetVector (see page 733)	Available since version 1.7.
	SetVector (see page 733)	Available since version 1.7.

### II.5.1.10.3.1 GetVector

**C++**

```
HRESULT GetVector(double* _x, double* _y, double* _z);
```

**C#**

```
public void GetVector(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Sub GetVector(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

**Description**

Available since version 1.7.

### II.5.1.10.3.2 SetVector

**C++**

```
HRESULT SetVector(double _x, double _y, double _z);
```

**C#**

```
public void SetVector(double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetVector(_x As double, _y As double, _z As double)
```

**Description**

Available since version 1.7.

## II.5.1.11 IRobotThicknessMatrix

**Class Hierarchy**

**C++**

```
interface IRobotThicknessMatrix : IDispatch;
```

**C#**

```
public interface IRobotThicknessMatrix;
```

**Visual Basic**

```
Public Interface IRobotThicknessMatrix
```

**Description**

Object provides access to stiffness matrix values for orthotropic thicknesses.

### II.5.1.11.1 IRobotThicknessMatrix Members

The following tables list the members exposed by IRobotThicknessMatrix.

**Public Methods**

	Name	Description
⊕	GetValue ( see page 734)	Function returns a value of the indicated matrix element. Available since version 1.7.
⊕	SetValue ( see page 734)	Function sets the value of the indicated matrix element. Available since version 1.7.

## II.5.1.11.2 IRobotThicknessMatrix Methods

The methods of the IRobotThicknessMatrix class are listed here.

### Public Methods

	Name	Description
💡	GetValue (see page 734)	Function returns a value of the indicated matrix element. Available since version 1.7.
💡	SetValue (see page 734)	Function sets the value of the indicated matrix element. Available since version 1.7.

### II.5.1.11.2.1 GetValue

#### C++

```
HRESULT GetValue(IRobotThicknessMatrixValue _val_id, double* ret);
```

#### C#

```
public double GetValue(IRobotThicknessMatrixValue _val_id);
```

#### Visual Basic

```
Public Function GetValue(_val_id As IRobotThicknessMatrixValue) As double
```

#### Description

Function returns a value of the indicated matrix element. Available since version 1.7.

### II.5.1.11.2.2 SetValue

#### C++

```
HRESULT SetValue(IRobotThicknessMatrixValue _val_id, double _val);
```

#### C#

```
public void SetValue(IRobotThicknessMatrixValue _val_id, double _val);
```

#### Visual Basic

```
Public Sub SetValue(_val_id As IRobotThicknessMatrixValue, _val As double)
```

#### Description

Function sets the value of the indicated matrix element. Available since version 1.7.

## II.5.1.12 IRobotThicknessMatrixValue

#### C++

```
enum IRobotThicknessMatrixValue;
```

#### C#

```
public enum IRobotThicknessMatrixValue;
```

#### Visual Basic

```
Public Enum IRobotThicknessMatrixValue
```

#### Members

Members	Description
I_TMV_D_XXXX = 0	Defining this quantity makes sense only in the case of shell structures (it is ignored for plates) Available since version 1.7.

I_TMV_D_XXYY = 1	Defining this quantity makes sense only in the case of shell structures (it is ignored for plates) Available since version 1.7.
I_TMV_D_YYYY = 2	Defining this quantity makes sense only in the case of shell structures (it is ignored for plates) Available since version 1.7.
I_TMV_D_XYXY = 3	Defining this quantity makes sense only in the case of shell structures (it is ignored for plates) Available since version 1.7.
I_TMV_K_XXXX = 4	Available since version 1.7.
I_TMV_K_XXYY = 5	Available since version 1.7.
I_TMV_K_YYYY = 6	Available since version 1.7.
I_TMV_K_XYXY = 7	Available since version 1.7.
I_TMV_H_AX = 8	Available since version 1.7.
I_TMV_H_AY = 9	Available since version 1.7.

**Description**

Physical interpretation of individual matrix values is described in the file of the Robot program help.

**II.5.1.13 IRobotSolidPropertiesData****Class Hierarchy****C++**

```
interface IRobotSolidPropertiesData : IDispatch;
```

**C#**

```
public interface IRobotSolidPropertiesData;
```

**Visual Basic**

```
Public Interface IRobotSolidPropertiesData
```

**Description**

Solid properties.

**Version**

Available since version 3.

**II.5.1.13.1 IRobotSolidPropertiesData Members**

The following tables list the members exposed by IRobotSolidPropertiesData.

**Public Fields**

	Name	Description
◆	DampCoef (see page 736)	Damping coefficient.
◆	E (see page 737)	Young's modulus.
◆	LX (see page 737)	Thermal expansion coefficient.
◆	MaterialModel (see page 737)	Material model.
◆	NU (see page 738)	Poisson ratio.
◆	RO (see page 738)	Density.

**Public Methods**

	Name	Description
◆	LoadFromDBase (see page 738)	Function reads data characteristic of a material of the indicated name from the material database. If the material of the indicated name is not found in the current material database, then zero value is returned (False). .

## II.5.1.13.2 IRobotSolidPropertiesData Fields

The fields of the IRobotSolidPropertiesData class are listed here.

### Public Fields

	Name	Description
◆	DampCoef (see page 736)	Damping coefficient.
◆	E (see page 737)	Young's modulus.
◆	LX (see page 737)	Thermal expansion coefficient.
◆	MaterialModel (see page 737)	Material model.
◆	NU (see page 738)	Poisson ratio.
◆	RO (see page 738)	Density.

### II.5.1.13.2.1 DampCoef

#### C++

```
HRESULT get_DampCoef(double* );
HRESULT put_DampCoef(double);
```

#### C#

```
public double DampCoef { get; set; }
```

#### Visual Basic

```
Public DampCoef As Double
```

#### Description

Damping coefficient.

#### Version

Available since version 3.

### II.5.1.13.2.2 E

#### C++

```
HRESULT get_E(double* );
HRESULT put_E(double);
```

#### C#

```
public double E { get; set; }
```

#### Visual Basic

```
Public E As Double
```

#### Description

Young's modulus.

#### Version

Available since version 3.

### II.5.1.13.2.3 LX

#### C++

```
HRESULT get_LX(double* );
HRESULT put_LX(double);
```

**C#**

```
public double LX { get; set; }
```

**Visual Basic**

```
Public LX As Double
```

**Description**

Thermal expansion coefficient.

**Version**

Available since version 3.

**II.5.1.13.2.4 MaterialModel****C++**

```
HRESULT get_MaterialModel(IRobotMaterialModel* );
HRESULT put_MaterialModel(IRobotMaterialModel* );
```

**C#**

```
public IRobotMaterialModel MaterialModel { get; set; }
```

**Visual Basic**

```
Public MaterialModel As IRobotMaterialModel
```

**Description**

Material model.

**Version**

Available since version 3.

**II.5.1.13.2.5 NU****C++**

```
HRESULT get_NU(double* );
HRESULT put_NU(double* );
```

**C#**

```
public double NU { get; set; }
```

**Visual Basic**

```
Public NU As Double
```

**Description**

Poisson ratio.

**Version**

Available since version 3.

**II.5.1.13.2.6 RO****C++**

```
HRESULT get_RO(double* );
HRESULT put_RO(double* );
```

**C#**

```
public double RO { get; set; }
```

**Visual Basic**

```
Public RO As double
```

**Description**

Density.

**Version**

Available since version 3.

**II.5.1.13.3 IRobotSolidPropertiesData Methods**

The methods of the IRobotSolidPropertiesData class are listed here.

**Public Methods**

	Name	Description
✳	LoadFromDBase (see page 738)	Function reads data characteristic of a material of the indicated name from the material database. If the material of the indicated name is not found in the current material database, then zero value is returned (False). .

**II.5.1.13.3.1 LoadFromDBase****C++**

```
HRESULT LoadFromDBase(BSTR _material_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromDBase(String _material_name);
```

**Visual Basic**

```
Public Function LoadFromDBase(_material_name As String) As Boolean
```

**Description**

Function reads data characteristic of a material of the indicated name from the material database. If the material of the indicated name is not found in the current material database, then zero value is returned (False). .

**Version**

Available since version 3.

**II.5.2 IRobotObjModificationType****C++**

```
enum IRobotObjModificationType;
```

**C#**

```
public enum IRobotObjModificationType;
```

**Visual Basic**

```
Public Enum IRobotObjModificationType
```

**Members**

Members	Description
I_OMT_NONE = 17	Undefined.
I_OMT_EXTRUSION = 7	Extrude.
I_OMT_LATHE = 8	Revolve.
I_OMT_PYRAMID = 9	Extrude along polyline.

## Description

Set of identifiers has been defined for modifications which may be performed on specified objects - structure components. .

### II.5.3 IRobotObjOperationType

#### C++

```
enum IRobotObjOperationType;
```

#### C#

```
public enum IRobotObjOperationType;
```

#### Visual Basic

```
Public Enum IRobotObjOperationType
```

#### Members

Members	Description
I_OOT_NONE = 16	Undefined.
I_OOT_MESH = 1	Deformation.
I_OOT_TRANSLATE = 2	Translation.
I_OOT_SCALE = 3	Scaling.
I_OOT_ROTATE = 18	Rotation.

## Description

Set of identifiers has been defined for operations which may be performed on specified objects - structure components. .

### II.5.4 IRobotObjModification

#### Class Hierarchy

#### C++

```
interface IRobotObjModification : IDispatch;
```

#### C#

```
public interface IRobotObjModification;
```

#### Visual Basic

```
Public Interface IRobotObjModification
```

#### Description

The basic interface describing modification of the initial object geometry .

#### II.5.4.1 IRobotObjModification Members

The following tables list the members exposed by IRobotObjModification.

#### Public Fields

	Name	Description
◆	Filled ( [ see page 740 )	The flag that indicates if the modification is filled with volumetric elements .
◆	NDiv ( [ see page 741 )	Number of divisions of lateral walls.
◆	Operations ( [ see page 741 )	Collection of operations describing transformation of the modification result (so called reference or image).
◆	Type ( [ see page 741 )	Modification type.

## Public Methods

	Name	Description
⊕	Add (see page 742)	The function adds a new operation.
⊖	Clear (see page 742)	The function clears the table of operations.

### II.5.4.2 IRobotObjModification Fields

The fields of the IRobotObjModification class are listed here.

#### Public Fields

	Name	Description
◆	Filled (see page 740)	The flag that indicates if the modification is filled with volumetric elements .
◆	NDiv (see page 741)	Number of divisions of lateral walls.
◆	Operations (see page 741)	Collection of operations describing transformation of the modification result (so called reference or image).
◆	Type (see page 741)	Modification type.

#### II.5.4.2.1 Filled

##### C++

```
HRESULT get_Filled(VARIANT_BOOL* );
HRESULT put_Filled(VARIANT_BOOL);
```

##### C#

```
public bool Filled { get; set; }
```

##### Visual Basic

```
Public Filled As Boolean
```

##### Description

The flag that indicates if the modification is filled with volumetric elements .

#### II.5.4.2.2 NDiv

##### C++

```
HRESULT get_NDiv(long* );
HRESULT put_NDiv(long);
```

##### C#

```
public long NDiv { get; set; }
```

##### Visual Basic

```
Public NDiv As long
```

##### Description

Number of divisions of lateral walls.

#### II.5.4.2.3 Operations

##### C++

```
HRESULT get_Operations(IRobotObjOperationCollection** );
```

##### C#

```
public IRobotObjOperationCollection Operations { get; }
```

**Visual Basic**

```
Public ReadOnly Operations As IRobotObjOperationCollection
```

**Description**

Collection of operations describing transformation of the modification result (so called reference or image).

**II.5.4.2.4 Type****C++**

```
HRESULT get_Type(IRobotObjModificationType* );
```

**C#**

```
public IRobotObjModificationType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotObjModificationType
```

**Description**

Modification type.

**II.5.4.3 IRobotObjModification Methods**

The methods of the IRobotObjModification class are listed here.

**Public Methods**

	Name	Description
>Add (see page 742)		The function adds a new operation.
Clear (see page 742)		The function clears the table of operations.

**II.5.4.3.1 Add****C++**

```
HRESULT Add( IRobotObjOperation* _operation);
```

**C#**

```
public void Add(IRobotObjOperation _operation);
```

**Visual Basic**

```
Public Sub Add(ByRef _operation As IRobotObjOperation)
```

**Description**

The function adds a new operation.

**II.5.4.3.2 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

The function clears the table of operations.

**II.5.5 IRobotObjOperation****Class Hierarchy****C++**

```
interface IRobotObjOperation : IDispatch;
```

**C#**

```
public interface IRobotObjOperation;
```

**Visual Basic**

```
Public Interface IRobotObjOperation
```

**Description**

The basic interface describing transformation (translation, rotation, scaling, deformation) of the result of modification (the so called reference or image).

**II.5.5.1 IRobotObjOperation Members**

The following tables list the members exposed by IRobotObjOperation.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Type (see page 743)	Operation type.

**II.5.5.2 IRobotObjOperation Fields**

The fields of the IRobotObjOperation class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Type (see page 743)	Operation type.

**II.5.5.2.1 Type****C++**

```
HRESULT get_Type(IRobotObjOperationType* );
```

**C#**

```
public IRobotObjOperationType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotObjOperationType
```

**Description**

Operation type.

**II.5.6 IRobotObjOperationCollection****Class Hierarchy****C++**

```
interface IRobotObjOperationCollection : IRobotCollection;
```

**C#**

```
public interface IRobotObjOperationCollection : IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotObjOperationCollection
```

**Description**

Collection of operations.

**II.5.7 IRobotObjModifExtrusion****Class Hierarchy****C++**

```
interface IRobotObjModifExtrusion : IRobotObjModification;
```

**C#**

```
public interface IRobotObjModifExtrusion : IRobotObjModification;
```

**Visual Basic**

```
Public Interface IRobotObjModifExtrusion
```

**Description**

The interface describing a modification of initial object geometry - extrude. .

**II.5.7.1 IRobotObjModifExtrusion Members**

The following tables list the members exposed by IRobotObjModifExtrusion.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Filled (see page 740)	The flag that indicates if the modification is filled with volumetric elements .
◆	NDiv (see page 741)	Number of divisions of lateral walls.
◆	Operations (see page 741)	Collection of operations describing transformation of the modification result (so called reference or image).
◆	Type (see page 741)	Modification type.
◆	Vector (see page 744)	Direction vector for extrusion.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡◆	Add (see page 742)	The function adds a new operation.
≡◆	Clear (see page 742)	The function clears the table of operations.

**II.5.7.2 IRobotObjModifExtrusion Fields**

The fields of the IRobotObjModifExtrusion class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Vector (see page 744)	Direction vector for extrusion.

### II.5.7.2.1 Vector

#### C++

```
HRESULT get_Vector(IRobotGeoPoint3D**);
HRESULT put_Vector(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D Vector { get; set; }
```

#### Visual Basic

```
Public Vector As IRobotGeoPoint3D
```

#### Description

Direction vector for extrusion.

## II.5.8 IRobotObjModifLathe

#### Class Hierarchy

#### C++

```
interface IRobotObjModifLathe : IRobotObjModification;
```

#### C#

```
public interface IRobotObjModifLathe : IRobotObjModification;
```

#### Visual Basic

```
Public Interface IRobotObjModifLathe
```

#### Description

Interface describing modification of initial geometry of the object - revolve operation.

### II.5.8.1 IRobotObjModifLathe Members

The following tables list the members exposed by IRobotObjModifLathe.

#### Public Fields

	Name	Description
◆	Angle (see page 745)	Revolution angle.
◆	AxsP1 (see page 745)	The first point of revolution axis.
◆	AxsP2 (see page 746)	The second point of revolution axis.
◆	Filled (see page 740)	The flag that indicates if the modification is filled with volumetric elements .
◆	NDiv (see page 741)	Number of divisions of lateral walls.
◆	Operations (see page 741)	Collection of operations describing transformation of the modification result (so called reference or image).
◆	Type (see page 741)	Modification type.

#### Public Methods

	Name	Description
◆	Add (see page 742)	The function adds a new operation.
◆	Clear (see page 742)	The function clears the table of operations.

### II.5.8.2 IRobotObjModifLathe Fields

The fields of the IRobotObjModifLathe class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Angle (see page 745)	Revolution angle.
◆	AxsP1 (see page 745)	The first point of revolution axis.
◆	AxsP2 (see page 746)	The second point of revolution axis.

**II.5.8.2.1 Angle****C++**

```
HRESULT get_Angle(double* );
HRESULT put_Angle(double);
```

**C#**

```
public double Angle { get; set; }
```

**Visual Basic**

```
Public AxsP1 As IRobotGeoPoint3D
```

**Description**

Revolution angle.

**II.5.8.2.2 AxsP1****C++**

```
HRESULT get_AxsP1(IRobotGeoPoint3D** );
HRESULT put_AxsP1(IRobotGeoPoint3D* );
```

**C#**

```
public IRobotGeoPoint3D AxsP1 { get; set; }
```

**Visual Basic**

```
Public AxsP1 As IRobotGeoPoint3D
```

**Description**

The first point of revolution axis.

**II.5.8.2.3 AxsP2****C++**

```
HRESULT get_AxsP2(IRobotGeoPoint3D** );
HRESULT put_AxsP2(IRobotGeoPoint3D* );
```

**C#**

```
public IRobotGeoPoint3D AxsP2 { get; set; }
```

**Visual Basic**

```
Public AxsP2 As IRobotGeoPoint3D
```

**Description**

The second point of revolution axis.

**II.5.9 IRobotObjModifPyramid****Class Hierarchy****C++**

```
interface IRobotObjModifPyramid : IRobotObjModification;
```

**C#**

```
public interface IRobotObjModifPyramid : IRobotObjModification;
```

**Visual Basic**

```
Public Interface IRobotObjModifPyramid
```

**Description**

Interface describing modification of initial object geometry - extrude along polyline.

**II.5.9.1 IRobotObjModifPyramid Members**

The following tables list the members exposed by IRobotObjModifPyramid.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Factors (see page 747)	Direction (XYZ) scaling factors .
◆	Filled (see page 740)	The flag that indicates if the modification is filled with volumetric elements .
◆	NDiv (see page 741)	Number of divisions of lateral walls.
◆	Operations (see page 741)	Collection of operations describing transformation of the modification result (so called reference or image).
◆	Points (see page 747)	Polyline points.
◆	Type (see page 741)	Modification type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add (see page 742)	The function adds a new operation.
◆	AddPoint (see page 748)	Function adds a new point of the polyline with a scaling factor.
◆	Clear (see page 742)	The function clears the table of operations.
◆	ClearPoints (see page 748)	Available since version 1.7.

**II.5.9.2 IRobotObjModifPyramid Fields**

The fields of the IRobotObjModifPyramid class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Factors (see page 747)	Direction (XYZ) scaling factors .
◆	Points (see page 747)	Polyline points.

**II.5.9.2.1 Factors****C++**

```
HRESULT get_Factors(IRobotGeoPoint3DCollection**);
```

**C#**

```
public IRobotGeoPoint3DCollection Factors { get; }
```

**Visual Basic**

```
Public ReadOnly Factors As IRobotGeoPoint3DCollection
```

**Description**

Direction (XYZ) scaling factors .

## II.5.9.2.2 Points

### C++

```
HRESULT get_Points(IRobotGeoPoint3DCollection**);
```

### C#

```
public IRobotGeoPoint3DCollection Points { get; }
```

### Visual Basic

```
Public ReadOnly Points As IRobotGeoPoint3DCollection
```

### Description

Polyline points.

## II.5.9.3 IRobotObjModifPyramid Methods

The methods of the IRobotObjModifPyramid class are listed here.

### Public Methods

	Name	Description
新加	AddPoint (see page 748)	Function adds a new point of the polyline with a scaling factor.
新加	ClearPoints (see page 748)	Available since version 1.7.

## II.5.9.3.1 AddPoint

### C++

```
HRESULT AddPoint(IRobotGeoPoint3D* _point, IRobotGeoPoint3D* _factor);
```

### C#

```
public void AddPoint(IRobotGeoPoint3D _point, IRobotGeoPoint3D _factor);
```

### Visual Basic

```
Public Sub AddPoint(ByRef _point As IRobotGeoPoint3D, ByRef _factor As IRobotGeoPoint3D)
```

### Description

Function adds a new point of the polyline with a scaling factor.

## II.5.9.3.2 ClearPoints

### C++

```
HRESULT ClearPoints();
```

### C#

```
public void ClearPoints();
```

### Visual Basic

```
Public Sub ClearPoints()
```

### Description

Available since version 1.7.

## II.5.10 IRobotObjOperTranslation

### Class Hierarchy

**C++**

```
interface IRobotObjOperTranslation : IRobotObjOperation;
```

**C#**

```
public interface IRobotObjOperTranslation : IRobotObjOperation;
```

**Visual Basic**

```
Public Interface IRobotObjOperTranslation
```

**Description**

The interface describing the operation of translation that transforms the result of modification of the initial object geometry.

**II.5.10.1 IRobotObjOperTranslation Members**

The following tables list the members exposed by IRobotObjOperTranslation.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Type (see page 743)	Operation type.
◆	Vector (see page 749)	Translation vector.

**II.5.10.2 IRobotObjOperTranslation Fields**

The fields of the IRobotObjOperTranslation class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Vector (see page 749)	Translation vector.

**II.5.10.2.1 Vector****C++**

```
HRESULT get_Vector(IRobotGeoPoint3D**);
HRESULT put_Vector(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Vector { get; set; }
```

**Visual Basic**

```
Public Vector As IRobotGeoPoint3D
```

**Description**

Translation vector.

**II.5.11 IRobotObjOperScaling****Class Hierarchy****C++**

```
interface IRobotObjOperScaling : IRobotObjOperation;
```

**C#**

```
public interface IRobotObjOperScaling : IRobotObjOperation;
```

**Visual Basic**

```
Public Interface IRobotObjOperScaling
```

## Description

The interface describing the operation of scaling, transforming the result of modification of object geometry .

### II.5.11.1 IRobotObjOperScaling Members

The following tables list the members exposed by IRobotObjOperScaling.

#### Public Fields

	Name	Description
◆	Center ( <a href="#">see page 750</a> )	Scaling center.
◆	Factor ( <a href="#">see page 750</a> )	Scaling coefficient in the three main directions.
◆	Type ( <a href="#">see page 743</a> )	Operation type.

### II.5.11.2 IRobotObjOperScaling Fields

The fields of the IRobotObjOperScaling class are listed here.

#### Public Fields

	Name	Description
◆	Center ( <a href="#">see page 750</a> )	Scaling center.
◆	Factor ( <a href="#">see page 750</a> )	Scaling coefficient in the three main directions.

#### II.5.11.2.1 Center

##### C++

```
HRESULT get_Center(IRobotGeoPoint3D**);
HRESULT put_Center(IRobotGeoPoint3D*);
```

##### C#

```
public IRobotGeoPoint3D Center { get; set; }
```

##### Visual Basic

```
Public Center As IRobotGeoPoint3D
```

#### Description

Scaling center.

#### II.5.11.2.2 Factor

##### C++

```
HRESULT get_Factor(IRobotGeoPoint3D**);
```

##### C#

```
public IRobotGeoPoint3D Factor { get; }
```

##### Visual Basic

```
Public ReadOnly Factor As IRobotGeoPoint3D
```

#### Description

Scaling coefficient in the three main directions.

### II.5.12 IRobotObjOperRotation

#### Class Hierarchy

##### C++

```
interface IRobotObjOperRotation : IRobotObjOperation;
```

**C#**

```
public interface IRobotObjOperRotation : IRobotObjOperation;
```

**Visual Basic**

```
Public Interface IRobotObjOperRotation
```

**Description**

Interface describing the operation of rotation that transforms the result of modification of object geometry. .

**II.5.12.1 IRobotObjOperRotation Members**

The following tables list the members exposed by IRobotObjOperRotation.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Angle (see page 751)	Rotation angle.
◆	AxsP1 (see page 751)	The first point of rotation axis.
◆	AxsP2 (see page 751)	The second point of rotation axis.
◆	Type (see page 743)	Operation type.

**II.5.12.2 IRobotObjOperRotation Fields**

The fields of the IRobotObjOperRotation class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Angle (see page 751)	Rotation angle.
◆	AxsP1 (see page 751)	The first point of rotation axis.
◆	AxsP2 (see page 751)	The second point of rotation axis.

**II.5.12.2.1 Angle****C++**

```
HRESULT get_Angle(double* );
HRESULT put_Angle(double);
```

**C#**

```
public double Angle { get; set; }
```

**Visual Basic**

```
Public Angle As Double
```

**Description**

Rotation angle.

**II.5.12.2.2 AxsP1****C++**

```
HRESULT get_AxsP1(IRobotGeoPoint3D** );
HRESULT put_AxsP1(IRobotGeoPoint3D* );
```

**C#**

```
public IRobotGeoPoint3D AxsP1 { get; set; }
```

**Visual Basic**

```
Public Axsp1 As IRobotGeoPoint3D
```

**Description**

The first point of rotation axis.

**II.5.12.2.3 Axsp2****C++**

```
HRESULT get_Axsp2(IRobotGeoPoint3D**);
HRESULT put_Axsp2(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D Axsp2 { get; set; }
```

**Visual Basic**

```
Public Axsp2 As IRobotGeoPoint3D
```

**Description**

The second point of rotation axis.

**II.5.13 IRobotObjOperMeshing****Class Hierarchy****C++**

```
interface IRobotObjOperMeshing : IRobotObjOperation;
```

**C#**

```
public interface IRobotObjOperMeshing : IRobotObjOperation;
```

**Visual Basic**

```
Public Interface IRobotObjOperMeshing
```

**Description**

The interface describing a deformation that transforms the result of modification of initial object geometry .

**II.5.13.1 IRobotObjOperMeshing Members**

The following tables list the members exposed by IRobotObjOperMeshing.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Points (see page 752)	Collection of indexes (numbers) of points that undergo displacement .
❖	Type (see page 743)	Operation type.
❖	Vectors (see page 753)	Collection of displacement vectors.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Add (see page 753)	The function adds a displacement vector for an indicated point that defines initial object geometry. The initial geometry of each object is defined through a set of points.
❖	Clear (see page 753)	The function clears the collection of vectors and point numbers.

## II.5.13.2 IRobotObjOperMeshing Fields

The fields of the IRobotObjOperMeshing class are listed here.

### Public Fields

	Name	Description
◆	Points (see page 752)	Collection of indexes (numbers) of points that undergo displacement .
◆	Vectors (see page 753)	Collection of displacement vectors.

### II.5.13.2.1 Points

#### C++

```
HRESULT get_Points(IRobotNumbersCollection**);
```

#### C#

```
public IRobotNumbersCollection Points { get; }
```

#### Visual Basic

```
Public ReadOnly Points As IRobotNumbersCollection
```

#### Description

Collection of indexes (numbers) of points that undergo displacement .

### II.5.13.2.2 Vectors

#### C++

```
HRESULT get_Vectors(IRobotGeoPoint3DCollection**);
```

#### C#

```
public IRobotGeoPoint3DCollection Vectors { get; }
```

#### Visual Basic

```
Public ReadOnly Vectors As IRobotGeoPoint3DCollection
```

#### Description

Collection of displacement vectors.

## II.5.13.3 IRobotObjOperMeshing Methods

The methods of the IRobotObjOperMeshing class are listed here.

### Public Methods

	Name	Description
◆	Add (see page 753)	The function adds a displacement vector for an indicated point that defines initial object geometry. The initial geometry of each object is defined through a set of points.
◆	Clear (see page 753)	The function clears the collection of vectors and point numbers.

### II.5.13.3.1 Add

#### C++

```
HRESULT Add(long _point, IRobotGeoPoint3D* _vector);
```

#### C#

```
public void Add(long _point, IRobotGeoPoint3D _vector);
```

**Visual Basic**

```
Public Sub Add(_point As long, ByRef _vector As IRobotGeoPoint3D)
```

**Description**

The function adds a displacement vector for an indicated point that defines initial object geometry. The initial geometry of each object is defined through a set of points.

**II.5.13.3.2 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

The function clears the collection of vectors and point numbers.

**II.5.14 IRobotObjEdge****Class Hierarchy****C++**

```
interface IRobotObjEdge : IDispatch;
```

**C#**

```
public interface IRobotObjEdge;
```

**Visual Basic**

```
Public Interface IRobotObjEdge
```

**Description**

Interface describing an object edge.

**II.5.14.1 IRobotObjEdge Members**

The following tables list the members exposed by IRobotObjEdge.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Path (see page 754)	Edge geometry .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Divide (see page 755)	Fuction divides the edge at the indicated point. If the _relative flag is set as True, then the _x parameter determines a relative distance between the division point and edge beginning - a value contained in the interval (0, 1). Otherwise, _x determines a distance between the division point and edge beginning expressed in meters. .
◆	DivideByPlane (see page 756)	Function divides the edge by cutting it with the indicated plane of the defined coordinate on the normal axis.
◆	DivideN (see page 756)	Function divides the edge into N equal parts.

	GetLabel (see page 756)	
	GetLabelName (see page 756)	
	GetLabels (see page 757)	
	HasLabel (see page 757)	
	RemoveLabel (see page 757)	
	SetLabel (see page 757)	

## II.5.14.2 IRobotObjEdge Fields

The fields of the IRobotObjEdge class are listed here.

### Public Fields

	Name	Description
	Path (see page 754)	Edge geometry .

### II.5.14.2.1 Path

#### C++

```
HRESULT get_Path(IRobotGeoPoint3DCollection**);
```

#### C#

```
public IRobotGeoPoint3DCollection Path { get; }
```

#### Visual Basic

```
Public ReadOnly Path As IRobotGeoPoint3DCollection
```

#### Description

Edge geometry .

## II.5.14.3 IRobotObjEdge Methods

The methods of the IRobotObjEdge class are listed here.

### Public Methods

	Name	Description
	Divide (see page 755)	Fuction divides the edge at the indicated point. If the _relative flag is set as True, then the _x parameter determines a relative distance between the division point and edge beginning - a value contained in the interval (0, 1). Otherwise, _x determines a distance between the division point and edge beginning expressed in meters. .
	DivideByPlane (see page 756)	Function divides the edge by cutting it with the indicated plane of the defined coordinate on the normal axis.
	DivideN (see page 756)	Function divides the edge into N equal parts.
	GetLabel (see page 756)	
	GetLabelName (see page 756)	
	GetLabels (see page 757)	
	HasLabel (see page 757)	
	RemoveLabel (see page 757)	
	SetLabel (see page 757)	

### II.5.14.3.1 Divide

#### C++

```
HRESULT Divide(double _x, VARIANT_BOOL _relative = false, VARIANT_BOOL _only_gen_node = false);
```

**C#**

```
public void Divide(double _x, bool _relative = false, bool _only_gen_node = false);
```

**Visual Basic**

```
Public Sub Divide(_x As Double, Optional _relative As Boolean = False, Optional  
_only_gen_node As Boolean = False)
```

**Description**

Fuction divides the edge at the indicated point. If the \_relative flag is set as True, then the \_x parameter determines a relative distance between the division point and edge beginning - a value contained in the interval (0, 1). Otherwise, \_x determines a distance between the division point and edge beginning expressed in meters..

**Version**

Available since version 4.5.

**II.5.14.3.2 DivideByPlane****C++**

```
HRESULT DivideByPlane(double _val, IRobotGeoCoordinateAxis _axis, VARIANT_BOOL  
_only_gen_node = false);
```

**C#**

```
public void DivideByPlane(double _val, IRobotGeoCoordinateAxis _axis, bool _only_gen_node =  
false);
```

**Visual Basic**

```
Public Sub DivideByPlane(_val As Double, _axis As IRobotGeoCoordinateAxis, Optional  
_only_gen_node As Boolean = False)
```

**Description**

Function divides the edge by cutting it with the indicated plane of the defined coordinate on the normal axis.

**Version**

Available since version 4.5.

**II.5.14.3.3 DivideN****C++**

```
HRESULT DivideN(long _n, VARIANT_BOOL _only_gen_nodes = false);
```

**C#**

```
public void DivideN(long _n, bool _only_gen_nodes = false);
```

**Visual Basic**

```
Public Sub DivideN(_n As Long, Optional _only_gen_nodes As Boolean = False)
```

**Description**

Function divides the edge into N equal parts.

**Version**

Available since version 4.5.

#### II.5.14.3.4 GetLabel

C++

```
HRESULT GetLabel(IRobotLabelType _lab_type, IRobotLabel** ret);
```

C#

```
public IRobotLabel GetLabel(IRobotLabelType _lab_type);
```

Visual Basic

```
Public Function GetLabel(_lab_type As IRobotLabelType) As IRobotLabel
```

#### II.5.14.3.5 GetLabelName

C++

```
HRESULT GetLabelName(IRobotLabelType _lab_type, BSTR* ret);
```

C#

```
public String GetLabelName(IRobotLabelType _lab_type);
```

Visual Basic

```
Public Function GetLabelName(_lab_type As IRobotLabelType) As String
```

#### II.5.14.3.6 GetLabels

C++

```
HRESULT GetLabels(IRobotCollection** ret);
```

C#

```
public IRobotCollection GetLabels();
```

Visual Basic

```
Public Function GetLabels() As IRobotCollection
```

#### II.5.14.3.7 HasLabel

C++

```
HRESULT HasLabel(IRobotLabelType _lab_type, VARIANT_BOOL* ret);
```

C#

```
public bool HasLabel(IRobotLabelType _lab_type);
```

Visual Basic

```
Public Function HasLabel(_lab_type As IRobotLabelType) As Boolean
```

#### II.5.14.3.8 RemoveLabel

C++

```
HRESULT RemoveLabel(IRobotLabelType _lab_type);
```

C#

```
public void RemoveLabel(IRobotLabelType _lab_type);
```

Visual Basic

```
Public Sub RemoveLabel(_lab_type As IRobotLabelType)
```

### II.5.14.3.9 SetLabel

**C++**

```
HRESULT SetLabel(IRobotLabelType _lab_type, BSTR _lab_name);
```

**C#**

```
public void SetLabel(IRobotLabelType _lab_type, String _lab_name);
```

**Visual Basic**

```
Public Sub SetLabel(_lab_type As IRobotLabelType, _lab_name As String)
```

## II.5.15 IRobotObjAttributes

**Class Hierarchy**

**C++**

```
interface IRobotObjAttributes : IDispatch;
```

**C#**

```
public interface IRobotObjAttributes;
```

**Visual Basic**

```
Public Interface IRobotObjAttributes
```

**Description**

The interface describing additional attributes of an object or its part.

### II.5.15.1 IRobotObjAttributes Members

The following tables list the members exposed by IRobotObjAttributes.

**Public Fields**

	Name	Description
❖	DirZ (see page 759)	The flag allowing change of the object component orientation to the opposite (change of the local Z axis sense).
❖	Meshed (see page 759)	Flag determining if mesh is to be generated on an object component.

**Public Methods**

	Name	Description
❖	GetDirX (see page 760)	Function reads the local X axis direction. Read values are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Additionally, the definition method is returned. Available since version 1.7.
❖	GetLabelName (see page 760)	
❖	GetLabels (see page 760)	
❖	GetLCS (see page 760)	Return Local Coordinate System of the panel.
❖	GetLCSDisplayPosition (see page 761)	
❖	HasLabel (see page 761)	
❖	RemoveLabel (see page 761)	

	SetDirX ( <a href="#">see page 761</a> )	Function defines the local X axis direction in the defined coordinate system. Additional function parameters determine the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Available since version 1.7.
	SetLabel ( <a href="#">see page 762</a> )	

## II.5.15.2 IRobotObjAttributes Fields

The fields of the IRobotObjAttributes class are listed here.

### Public Fields

	Name	Description
	DirZ ( <a href="#">see page 759</a> )	The flag allowing change of the object component orientation to the opposite (change of the local Z axis sense) .
	Meshed ( <a href="#">see page 759</a> )	Flag determining if mesh is to be generated on an object component.

### II.5.15.2.1 DirZ

#### C++

```
HRESULT get_DirZ(VARIANT_BOOL* );
HRESULT put_DirZ(VARIANT_BOOL);
```

#### C#

```
public bool DirZ { get; set; }
```

#### Visual Basic

```
Public DirZ As Boolean
```

#### Description

The flag allowing change of the object component orientation to the opposite (change of the local Z axis sense) .

### II.5.15.2.2 Meshed

#### C++

```
HRESULT get_Meshed(VARIANT_BOOL* );
HRESULT put_Meshed(VARIANT_BOOL);
```

#### C#

```
public bool Meshed { get; set; }
```

#### Visual Basic

```
Public Meshed As Boolean
```

#### Description

Flag determining if mesh is to be generated on an object component.

### II.5.15.3 IRobotObjAttributes Methods

The methods of the IRobotObjAttributes class are listed here.

### Public Methods

	Name	Description
	GetDirX ( <a href="#">see page 760</a> )	Function reads the local X axis direction. Read values are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Additionally, the definition method is returned. Available since version 1.7.

	GetLabelName (see page 760)	
	GetLabels (see page 760)	
	GetLCS (see page 760)	Return Local Coordinate System of the panel.
	GetLCSDisplayPosition (see page 761)	
	HasLabel (see page 761)	
	RemoveLabel (see page 761)	
	SetDirX (see page 761)	Function defines the local X axis direction in the defined coordinate system. Additional function parameters determine the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Available since version 1.7.
	SetLabel (see page 762)	

### II.5.15.3.1 GetDirX

#### C++

```
HRESULT GetDirX(double* _x, double* _y, double* _z, IRobotObjLocalXDirDefinitionType* ret);
```

#### C#

```
public IRobotObjLocalXDirDefinitionType GetDirX(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetDirX(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As IRobotObjLocalXDirDefinitionType
```

#### Description

Function reads the local X axis direction. Read values are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Additionally, the definition method is returned. Available since version 1.7.

### II.5.15.3.2 GetLabelName

#### C++

```
HRESULT GetLabelName(IRobotLabelType _lab_type, BSTR* ret);
```

#### C#

```
public String GetLabelName(IRobotLabelType _lab_type);
```

#### Visual Basic

```
Public Function GetLabelName(_lab_type As IRobotLabelType) As String
```

### II.5.15.3.3 GetLabels

#### C++

```
HRESULT GetLabels(IRobotCollection** ret);
```

#### C#

```
public IRobotCollection GetLabels();
```

#### Visual Basic

```
Public Function GetLabels() As IRobotCollection
```

#### II.5.15.3.4 GetLCS

##### C++

```
HRESULT GetLCS(IRobotGeoPoint3D* _x, IRobotGeoPoint3D* _y, IRobotGeoPoint3D* _z);
```

##### C#

```
public void GetLCS(IRobotGeoPoint3D _x, IRobotGeoPoint3D _y, IRobotGeoPoint3D _z);
```

##### Visual Basic

```
Public Sub GetLCS(ByRef _x As IRobotGeoPoint3D, ByRef _y As IRobotGeoPoint3D, ByRef _z As IRobotGeoPoint3D)
```

##### Description

Return Local Coordinate System of the panel.

##### Version

Available since version 14.7.

#### II.5.15.3.5 GetLCSDisplayPosition

##### C++

```
HRESULT GetLCSDisplayPosition(IRobotGeoPoint3D* _pt);
```

##### C#

```
public void GetLCSDisplayPosition(IRobotGeoPoint3D _pt);
```

##### Visual Basic

```
Public Sub GetLCSDisplayPosition(ByRef _pt As IRobotGeoPoint3D)
```

##### Version

Available since version 14.7.

#### II.5.15.3.6 HasLabel

##### C++

```
HRESULT HasLabel(IRobotLabelText _lab_type, VARIANT_BOOL* ret);
```

##### C#

```
public bool HasLabel(IRobotLabelText _lab_type);
```

##### Visual Basic

```
Public Function HasLabel(_lab_type As IRobotLabelText) As Boolean
```

#### II.5.15.3.7 RemoveLabel

##### C++

```
HRESULT RemoveLabel(IRobotLabelText _lab_type);
```

##### C#

```
public void RemoveLabel(IRobotLabelText _lab_type);
```

##### Visual Basic

```
Public Sub RemoveLabel(_lab_type As IRobotLabelText)
```

### II.5.15.3.8 SetDirX

**C++**

```
HRESULT SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

**C#**

```
public void SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetDirX(_def_type As IRobotObjLocalXDirDefinitionType, _x As Double, _y As Double, _z As Double)
```

**Description**

Function defines the local X axis direction in the defined coordinate system. Additional function parameters determine the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the system beginning for definition in the polar system. Available since version 1.7.

### II.5.15.3.9 SetLabel

**C++**

```
HRESULT SetLabel(IRobotLabelType _lab_type, BSTR _lab_name);
```

**C#**

```
public void SetLabel(IRobotLabelType _lab_type, String _lab_name);
```

**Visual Basic**

```
Public Sub SetLabel(_lab_type As IRobotLabelType, _lab_name As String)
```

## II.5.16 IRobotObjPart

**Class Hierarchy**

**C++**

```
interface IRobotObjPart : IDispatch;
```

**C#**

```
public interface IRobotObjPart;
```

**Visual Basic**

```
Public Interface IRobotObjPart
```

**Description**

Interface describing an object component. .

### II.5.16.1 IRobotObjPart Members

The following tables list the members exposed by IRobotObjPart.

**Public Fields**

	Name	Description
❖	Attribs (see page 763)	Attributes of the object component .
❖	Type (see page 763)	Object component type Available since version 2.0.

## Public Methods

	Name	Description
💡	GetGeometry (see page 763)	Function returns definition of the object component geometry. Available since version 2.0.

## II.5.16.2 IRobotObjPart Fields

The fields of the IRobotObjPart class are listed here.

### Public Fields

	Name	Description
💡	Attribs (see page 763)	Attributes of the object component .
💡	Type (see page 763)	Object component type Available since version 2.0.

### II.5.16.2.1 Attribs

#### C++

```
HRESULT get_Attribs(IRobotObjAttributes**);
```

#### C#

```
public IRobotObjAttributes Attribs { get; }
```

#### Visual Basic

```
Public ReadOnly Attribs As IRobotObjAttributes
```

#### Description

Attributes of the object component .

### II.5.16.2.2 Type

#### C++

```
HRESULT get_Type(IRobotObjPartType*);
```

#### C#

```
public IRobotObjPartType Type { get; }
```

#### Visual Basic

```
Public ReadOnly Type As IRobotObjPartType
```

#### Description

Object component type Available since version 2.0.

## II.5.16.3 IRobotObjPart Methods

The methods of the IRobotObjPart class are listed here.

### Public Methods

	Name	Description
💡	GetGeometry (see page 763)	Function returns definition of the object component geometry. Available since version 2.0.

### II.5.16.3.1 GetGeometry

#### C++

```
HRESULT GetGeometry(IRobotGeoObject** ret);
```

**C#**

```
public IRobotGeoObject GetGeometry();
```

**Visual Basic**

```
Public Function GetGeometry() As IRobotGeoObject
```

**Description**

Function returns definition of the object component geometry. Available since version 2.0.

## II.5.17 IRobotObjModificationCollection

**Class Hierarchy****C++**

```
interface IRobotObjModificationCollection : IRobotCollection;
```

**C#**

```
public interface IRobotObjModificationCollection : IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotObjModificationCollection
```

**Description**

Collection of modifications .

## II.5.18 IRobotObjEdgeCollection

**Class Hierarchy****C++**

```
interface IRobotObjEdgeCollection : IRobotCollection;
```

**C#**

```
public interface IRobotObjEdgeCollection : IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotObjEdgeCollection
```

**Description**

Collection of edges.

## II.5.19 IRobotObjObject

**Class Hierarchy****C++**

```
interface IRobotObjObject : IRobotDataObject;
```

**C#**

```
public interface IRobotObjObject : IRobotDataObject;
```

**Visual Basic**

```
Public Interface IRobotObjObject
```

## II.5.19.1 IRobotObjObject Members

The following tables list the members exposed by IRobotObjObject.

### Public Fields

	<b>Name</b>	<b>Description</b>
◆	AnalyzeTTMethod ( <a href="#">see page 766</a> )	A flag indicating if the object is taken into consideration in the triangular / trapezoidal method.
◆	FiniteElems ( <a href="#">see page 766</a> )	List of all finite elements assigned to the object.
◆	Host ( <a href="#">see page 767</a> )	Number ( <a href="#">see page 30</a> ) of the object which is a 'host' for this object.
◆	IsVolume ( <a href="#">see page 767</a> )	Flag (read-only) indicating if the object is a solid.
◆	Main ( <a href="#">see page 767</a> )	Main object component.
◆	Mesh ( <a href="#">see page 767</a> )	Component managing the finite element mesh for this object.
◆	Name ( <a href="#">see page 768</a> )	Object name.
◆	NameTemplate ( <a href="#">see page 768</a> )	Object name pattern.
◆	Nodes ( <a href="#">see page 768</a> )	List of nodes belonging to the object.
◆	Number ( <a href="#">see page 30</a> )	Object number is assigned by the user; it is unique among all the components of the same type..
◆	PartsCount ( <a href="#">see page 769</a> )	Number ( <a href="#">see page 30</a> ) of object components .
◆	Reference ( <a href="#">see page 769</a> )	Main ( <a href="#">see page 767</a> ) component reference Available since version 2.0.
◆	StructuralType ( <a href="#">see page 769</a> )	Type of structure object.
◆	UniqueId ( <a href="#">see page 769</a> )	Unique object identifier Available since version 1.7.

### Public Methods

	<b>Name</b>	<b>Description</b>
◆	CalcArea ( <a href="#">see page 770</a> )	Function calculates the object area.
◆	CalcVol ( <a href="#">see page 770</a> )	Function calculates the solid volume. .
◆	GetFiniteElemsData ( <a href="#">see page 771</a> )	Function gives access to data of finite elements belonging to the object.
◆	GetHostedObjects ( <a href="#">see page 771</a> )	Function returns selection of objects for which this object is a host.
◆	GetLabel ( <a href="#">see page 31</a> )	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabelName ( <a href="#">see page 31</a> )	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
◆	GetLabels ( <a href="#">see page 31</a> )	The function returns a collection containing all labels (of different types) defined for a given object. .
◆	GetPart ( <a href="#">see page 771</a> )	Function returns the indicated object component. Components are indexed from 1 to their number.
◆	GetPartType ( <a href="#">see page 772</a> )	The function returns the type of the indicated object part.
◆	HasLabel ( <a href="#">see page 32</a> )	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
◆	Initialize ( <a href="#">see page 772</a> )	The function initializes or rebuilds the object on the basis of the base geometry and transformations defined in the main component. .
◆	RemoveLabel ( <a href="#">see page 32</a> )	The function deletes a label of the indicated type. .
◆	SetHostedObjects ( <a href="#">see page 772</a> )	Function sets the object as a host for objects described by the given selection.
◆	SetLabel ( <a href="#">see page 32</a> )	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .
◆	Update ( <a href="#">see page 772</a> )	The function updates definition of the object once its attributes are changed..

## II.5.19.2 IRobotObjObject Fields

The fields of the IRobotObjObject class are listed here.

### Public Fields

	Name	Description
◆	AnalyzeTTMethod ( [ see page 766 )	A flag indicating if the object is taken into consideration in the triangular / trapezoidal method.
◆	FiniteElems ( [ see page 766 )	List of all finite elements assigned to the object.
◆	Host ( [ see page 767 )	Number ( [ see page 30 ) of the object which is a 'host' for this object.
◆	IsVolume ( [ see page 767 )	Flag (read-only) indicating if the object is a solid.
◆	Main ( [ see page 767 )	Main object component.
◆	Mesh ( [ see page 767 )	Component managing the finite element mesh for this object.
◆	Name ( [ see page 768 )	Object name.
◆	NameTemplate ( [ see page 768 )	Object name pattern.
◆	Nodes ( [ see page 768 )	List of nodes belonging to the object.
◆	PartsCount ( [ see page 769 )	Number ( [ see page 30 ) of object components .
◆	Reference ( [ see page 769 )	Main ( [ see page 767 ) component reference Available since version 2.0.
◆	StructuralType ( [ see page 769 )	Type of structure object.
◆	UniqueId ( [ see page 769 )	Unique object identifier Available since version 1.7.

## II.5.19.2.1 AnalyzeTTMethod

### C++

```
HRESULT get_AnalyzeTTMethod(VARIANT_BOOL* );
HRESULT put_AnalyzeTTMethod(VARIANT_BOOL );
```

### C#

```
public bool AnalyzeTTMethod { get; set; }
```

### Visual Basic

```
Public AnalyzeTTMethod As Boolean
```

### Description

A flag indicating if the object is taken into consideration in the triangular / trapezoidal method.

### Version

Available since version 11.

## II.5.19.2.2 FiniteElems

### C++

```
HRESULT get_FiniteElems(BSTR* );
```

### C#

```
public String FiniteElems { get; }
```

### Visual Basic

```
Public ReadOnly FiniteElems As String
```

### Description

List of all finite elements assigned to the object.

### Version

Available since version 2.5.

### II.5.19.2.3 Host

#### C++

```
HRESULT get_Host(long* );
HRESULT put_Host(long );
```

#### C#

```
public long Host { get; set; }
```

#### Visual Basic

```
Public Host As long
```

#### Description

Number (see page 30) of the object which is a 'host' for this object.

#### Version

Available since version 11.

### II.5.19.2.4 IsVolume

#### C++

```
HRESULT get_IsVolume(VARIANT_BOOL* );
```

#### C#

```
public bool IsVolume { get; }
```

#### Visual Basic

```
Public ReadOnly IsVolume As Boolean
```

#### Description

Flag (read-only) indicating if the object is a solid.

#### Version

Available since version 4.5.

### II.5.19.2.5 Main

#### C++

```
HRESULT get_Main(IRobotObjPartMain** );
```

#### C#

```
public IRobotObjPartMain Main { get; }
```

#### Visual Basic

```
Public ReadOnly Main As IRobotObjPartMain
```

#### Description

Main object component.

### II.5.19.2.6 Mesh

#### C++

```
HRESULT get_Mesh(IRobotObjMesh** );
```

#### C#

```
public IRobotObjMesh Mesh { get; }
```

**Visual Basic**

```
Public ReadOnly Mesh As IRobotObjMesh
```

**Description**

Component managing the finite element mesh for this object.

**Version**

Available since version 3.

**II.5.19.2.7 Name****C++**

```
HRESULT get_Name(BSTR* );
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Description**

Object name.

**Version**

Available since version 7.5.

**II.5.19.2.8 NameTemplate****C++**

```
HRESULT get_NameTemplate(BSTR* );
HRESULT put_NameTemplate(BSTR);
```

**C#**

```
public String NameTemplate { get; set; }
```

**Visual Basic**

```
Public NameTemplate As String
```

**Description**

Object name pattern.

**Version**

Available since version 7.5.

**II.5.19.2.9 Nodes****C++**

```
HRESULT get_Nodes(BSTR* );
```

**C#**

```
public String Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As String
```

**Description**

List of nodes belonging to the object.

**Version**

Available since version 4.

**II.5.19.2.10 PartsCount****C++**

```
HRESULT get_PartsCount(long*);
```

**C#**

```
public long PartsCount { get; }
```

**Visual Basic**

```
Public ReadOnly PartsCount As long
```

**Description**

Number (see page 30) of object components .

**II.5.19.2.11 Reference****C++**

```
HRESULT get_Reference(IRobotObjPartReference**);
```

**C#**

```
public IRobotObjPartReference Reference { get; }
```

**Visual Basic**

```
Public ReadOnly Reference As IRobotObjPartReference
```

**Description**

Main (see page 767) component reference Available since version 2.0.

**II.5.19.2.12 StructuralType****C++**

```
HRESULT get_StructuralType(IRobotObjectStructuralType*);  
HRESULT put_StructuralType(IRobotObjectStructuralType);
```

**C#**

```
public IRobotObjectStructuralType StructuralType { get; set; }
```

**Visual Basic**

```
Public StructuralType As IRobotObjectStructuralType
```

**Description**

Type of structure object.

**Version**

Available since version 9.7.

**II.5.19.2.13 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

## Visual Basic

```
Public ReadOnly UniqueId As long
```

### Description

Unique object identifier Available since version 1.7.

## II.5.19.3 IRobotObjObject Methods

The methods of the IRobotObjObject class are listed here.

### Public Methods

	Name	Description
💡	CalcArea (see page 770)	Function calculates the object area.
💡	CalcVol (see page 770)	Function calculates the solid volume. .
💡	GetFiniteElemsData (see page 771)	Function gives access to data of finite elements belonging to the object.
💡	GetHostedObjects (see page 771)	Function returns selection of objects for which this object is a host.
💡	GetPart (see page 771)	Function returns the indicated object component. Components are indexed from 1 to their number.
💡	GetPartType (see page 772)	The function returns the type of the indicated object part.
💡	Initialize (see page 772)	The function initializes or rebuilds the object on the basis of the base geometry and transformations defined in the main component. .
💡	SetHostedObjects (see page 772)	Function sets the object as a host for objects described by the given selection.
💡	Update (see page 772)	The function updates definition of the object once its attributes are changed. .

### II.5.19.3.1 CalcArea

#### C++

```
HRESULT CalcArea(double* ret);
```

#### C#

```
public double CalcArea();
```

## Visual Basic

```
Public Function CalcArea() As double
```

### Description

Function calculates the object area.

### Version

Available since version 4.5.

### II.5.19.3.2 CalcVol

#### C++

```
HRESULT CalcVol(double* ret);
```

#### C#

```
public double CalcVol();
```

## Visual Basic

```
Public Function CalcVol() As double
```

**Description**

Function calculates the solid volume. .

**Version**

Available since version 4.5.

**II.5.19.3.3 GetFiniteElemsData****C++**

```
HRESULT GetFiniteElemsData(IRobotFiniteElementDataSet* __ret_data);
```

**C#**

```
public void GetFiniteElemsData(IRobotFiniteElementDataSet __ret_data);
```

**Visual Basic**

```
Public Sub GetFiniteElemsData(ByRef __ret_data As IRobotFiniteElementDataSet)
```

**Description**

Function gives access to data of finite elements belonging to the object.

**Version**

Available since version 9.

**II.5.19.3.4 GetHostedObjects****C++**

```
HRESULT GetHostedObjects(IRobotSelection** ret);
```

**C#**

```
public IRobotSelection GetHostedObjects();
```

**Visual Basic**

```
Public Function GetHostedObjects() As IRobotSelection
```

**Description**

Function returns selection of objects for which this object is a host.

**Version**

Available since version 11.

**II.5.19.3.5 GetPart****C++**

```
HRESULT GetPart(long __part_num, IRobotObjPart** ret);
```

**C#**

```
public IRobotObjPart GetPart(long __part_num);
```

**Visual Basic**

```
Public Function GetPart(__part_num As long) As IRobotObjPart
```

**Description**

Function returns the indicated object component. Components are indexed from 1 to their number.

### II.5.19.3.6 GetPartType

**C++**

```
HRESULT GetPartType(long _part_num, IRobotObjPartType* ret);
```

**C#**

```
public IRobotObjPartType GetPartType(long _part_num);
```

**Visual Basic**

```
Public Function GetPartType(_part_num As long) As IRobotObjPartType
```

**Description**

The function returns the type of the indicated object part.

### II.5.19.3.7 Initialize

**C++**

```
HRESULT Initialize();
```

**C#**

```
public void Initialize();
```

**Visual Basic**

```
Public Sub Initialize()
```

**Description**

The function initializes or rebuilds the object on the basis of the base geometry and transformations defined in the main component..

### II.5.19.3.8 SetHostedObjects

**C++**

```
HRESULT SetHostedObjects(IRobotSelection* _obj_sel);
```

**C#**

```
public void SetHostedObjects(IRobotSelection _obj_sel);
```

**Visual Basic**

```
Public Sub SetHostedObjects(ByRef _obj_sel As IRobotSelection)
```

**Description**

Function sets the object as a host for objects described by the given selection.

**Version**

Available since version 11.

### II.5.19.3.9 Update

**C++**

```
HRESULT Update();
```

**C#**

```
public void Update();
```

**Visual Basic**

```
Public Sub Update()
```

**Description**

The function updates definition of the object once its attributes are changed. .

**II.5.20 IRobotObjPartType****C++**

```
enum IRobotObjPartType;
```

**C#**

```
public enum IRobotObjPartType;
```

**Visual Basic**

```
Public Enum IRobotObjPartType
```

**Members**

Members	Description
I_OPT_MAIN = 0	The main part of the object .
I_OPT_REFERENCE = 1	Reference - the image of the main part of the object after transformation.
I_OPT_SIDE = 2	Side - side object component.

**II.5.21 IRobotObjObjectServer****Class Hierarchy****C++**

```
interface IRobotObjObjectServer : IRobotDataObjectServer;
```

**C#**

```
public interface IRobotObjObjectServer : IRobotDataObjectServer;
```

**Visual Basic**

```
Public Interface IRobotObjObjectServer
```

**Description**

Server of object that are structure components .

**II.5.21.1 IRobotObjObjectServer Members**

The following tables list the members exposed by IRobotObjObjectServer.

**Public Fields**

	Name	Description
❖	AutoRecalcHoles ( [ see page 775 ] )	Flag switching on/off automatic detection of openings.
❖	FreeNumber ( [ see page 776 ] )	First free user number available for the object Available since version 1.7.
❖	LinearReleases ( [ see page 776 ] )	Server of linear releases defined for objects.
❖	Mesh ( [ see page 776 ] )	Component managing the finite element mesh on the level of the entire structure model (for all the objects defined in the structure) .

## Public Methods

	Name	Description
✳️	BeginMultiOperation ( <a href="#">see page 778</a> )	Function enables speed-up of generation or modification of a whole object group through appropriate optimization of some operations. After completing object generation, EndMultiOperation ( <a href="#">see page 782</a> ()) function should be called up to update definitions of objects in a structure. .
✳️	CalcArea ( <a href="#">see page 778</a> )	Function calculates area of the indicated object component. If the object component index equals 0, function returns area of the whole object. Attention: Before calling the function, mesh of surface finite elements should be generated. .
✳️	CalcVol ( <a href="#">see page 779</a> )	Function calculates volume of the indicated solid or its component. If the object component index equals 0, function returns volume of the whole solid. Attention: Before calling the function, mesh of volumetric finite elements should be generated.
✳️	Create ( <a href="#">see page 779</a> )	The function creates and returns a new structure object with the user-defined number .
✳️	CreateArc ( <a href="#">see page 779</a> )	Function creates an arc based on the indicated points. .
✳️	CreateCircle ( <a href="#">see page 780</a> )	Function creates a circle of the specified user-defined number based on three points. .
✳️	CreateCone ( <a href="#">see page 780</a> )	Function creates a cone based on the indicated four points. First three points define the cone base and the fourth point defines position of the vertex. To create a truncated cone the top radius should be provided as a fifth defining point .
✳️	CreateContour ( <a href="#">see page 780</a> )	Function creates a contour based on the specified points. .
✳️	CreateCube ( <a href="#">see page 781</a> )	Function creates a cube based on the indicated four points - first three points (P1, P2, P3) define the cube base and the fourth point (PH) defines the height. The cube will be created in such a manner so that the PH point is connected by an edge with the P3 point. .
✳️	CreateCylinder ( <a href="#">see page 781</a> )	Function creates a cylinder based on the indicated four points. First three points define the cylinder base and the fourth point (PH) defines its height. The cylinder will be created in such a manner so that the fourth point (PH) is a center of the second base of the cylinder. .
✳️	CreateOnFiniteElems ( <a href="#">see page 781</a> )	Function creates one or more objects in such a manner so that they cover the indicated list of finite elements. Function returns the collection of the created object numbers. .
✳️	CreatePolyline ( <a href="#">see page 782</a> )	Function creates a polyline of the specified user-defined number based on the indicated points. .
✳️	CreateSolid ( <a href="#">see page 782</a> )	Function creates a solid based on the specified list of surface objects.
✳️	Delete ( <a href="#">see page 34</a> )	The function deletes the object of the indicated number. .
✳️	DeleteMany ( <a href="#">see page 34</a> )	The function deletes all objects that meet the criteria of the indicated selection. .
✳️	EndMultiOperation ( <a href="#">see page 782</a> )	Function updates definitions of objects in a structure. It should be called up only if BeginMultiOperation ( <a href="#">see page 778</a> ()) function has been called up earlier. .
✳️	Exist ( <a href="#">see page 34</a> )	The function returns True (non-zero value) if the object of the indicated name already exists. .
✳️	FindWithId ( <a href="#">see page 783</a> )	Function returns a number of the object with the specified unique identifier. If the object is not found, then, zero value is returned (0). Available since version 1.7.
✳️	Get ( <a href="#">see page 34</a> )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. .
✳️	GetAll ( <a href="#">see page 35</a> )	The function returns a collection containing all objects managed by the server. .

	GetAnalyzeTTMethodEnabled ( <a href="#">see page 783</a> )	Function returns selection of objects taking into consideration in the triangular / trapezoidal distribution method.
	GetFiniteElemsData ( <a href="#">see page 783</a> )	Function returns finite element data for the determined object selection.
	GetHost ( <a href="#">see page 784</a> )	Function returns the number of host object for the given object.
	GetHostedObjects ( <a href="#">see page 784</a> )	Function returns a list of objects for which the given object is a host.
	GetMany ( <a href="#">see page 35</a> )	The function returns a collection of objects that meet the criteria of the indicated selection. .
	GetName ( <a href="#">see page 784</a> )	Function returns a name of the object of the given number.
	GetNameTemplate ( <a href="#">see page 785</a> )	Function returns the name pattern of an object of the specified number.
	GetStructuralType ( <a href="#">see page 785</a> )	Function returns the structure object type for an object with the specified number.
	GetUniqueld ( <a href="#">see page 785</a> )	Function returns a unique identifier for an object with the user-specified number.
	IsVolume ( <a href="#">see page 786</a> )	Function returns the True value, if the indicated object is a solid. .
	RemoveLabel ( <a href="#">see page 35</a> )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. .
	SetAnalyzeTTMethod ( <a href="#">see page 786</a> )	Function allows for adding/removing objects to/from list of objects taken into consideration in triangular/trapezoidal distribution method.
	SetHost ( <a href="#">see page 786</a> )	Function sets the number of host object for the given object.
	SetHostedObjects ( <a href="#">see page 787</a> )	Function sets a list of objects for which the given object is to be a host.
	SetLabel ( <a href="#">see page 36</a> )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. .
	SetNameTemplate ( <a href="#">see page 787</a> )	Function sets a name pattern for the specified object.
	SetStructuralType ( <a href="#">see page 787</a> )	Function sets a structure object type for the specified object list.

## II.5.21.2 IRobotObjObjectServer Fields

The fields of the IRobotObjObjectServer class are listed here.

### Public Fields

	Name	Description
	AutoRecalcHoles ( <a href="#">see page 775</a> )	Flag switching on/off automatic detection of openings.
	FreeNumber ( <a href="#">see page 776</a> )	First free user number available for the object Available since version 1.7.
	LinearReleases ( <a href="#">see page 776</a> )	Server of linear releases defined for objects.
	Mesh ( <a href="#">see page 776</a> )	Component managing the finite element mesh on the level of the entire structure model (for all the objects defined in the structure) .

### II.5.21.2.1 AutoRecalcHoles

#### C++

```
HRESULT get_AutoRecalcHoles(VARIANT_BOOL* );
HRESULT put_AutoRecalcHoles(VARIANT_BOOL);
```

#### C#

```
public bool AutoRecalcHoles { get; set; }
```

#### Visual Basic

```
Public AutoRecalcHoles As Boolean
```

#### Description

Flag switching on/off automatic detection of openings.

**Version**

Available since version 3.5.

**II.5.21.2.2 FreeNumber****C++**

```
HRESULT get_FreeNumber(long*);
```

**C#**

```
public long FreeNumber { get; }
```

**Visual Basic**

```
Public ReadOnly FreeNumber As long
```

**Description**

First free user number available for the object Available since version 1.7.

**II.5.21.2.3 LinearReleases****C++**

```
HRESULT get_LinearReleases(IRobotLinearReleaseServer**);
```

**C#**

```
public IRobotLinearReleaseServer LinearReleases { get; }
```

**Visual Basic**

```
Public ReadOnly LinearReleases As IRobotLinearReleaseServer
```

**Description**

Server of linear releases defined for objects.

**Version**

Available since version 2.5.

**II.5.21.2.4 Mesh****C++**

```
HRESULT get_Mesh(IRobotObjMesh**);
```

**C#**

```
public IRobotObjMesh Mesh { get; }
```

**Visual Basic**

```
Public ReadOnly Mesh As IRobotObjMesh
```

**Description**

Component managing the finite element mesh on the level of the entire structure model (for all the objects defined in the structure).

**Version**

Available since version 3.

**II.5.21.3 IRobotObjObjectServer Methods**

The methods of the IRobotObjObjectServer class are listed here.

## Public Methods

	Name	Description
✳️	BeginMultiOperation ( <a href="#">see page 778</a> )	Function enables speed-up of generation or modification of a whole object group through appropriate optimization of some operations. After completing object generation, EndMultiOperation ( <a href="#">see page 782</a> ()) function should be called up to update definitions of objects in a structure. .
✳️	CalcArea ( <a href="#">see page 778</a> )	Function calculates area of the indicated object component. If the object component index equals 0, function returns area of the whole object. Attention: Before calling the function, mesh of surface finite elements should be generated. .
✳️	CalcVol ( <a href="#">see page 779</a> )	Function calculates volume of the indicated solid or its component. If the object component index equals 0, function returns volume of the whole solid. Attention: Before calling the function, mesh of volumetric finite elements should be generated.
✳️	Create ( <a href="#">see page 779</a> )	The function creates and returns a new structure object with the user-defined number .
✳️	CreateArc ( <a href="#">see page 779</a> )	Function creates an arc based on the indicated points. .
✳️	CreateCircle ( <a href="#">see page 780</a> )	Function creates a circle of the specified user-defined number based on three points. .
✳️	CreateCone ( <a href="#">see page 780</a> )	Function creates a cone based on the indicated four points. First three points define the cone base and the fourth point defines position of the vertex. To create a truncated cone the top radius should be provided as a fifth defining point.
✳️	CreateContour ( <a href="#">see page 780</a> )	Function creates a contour based on the specified points. .
✳️	CreateCube ( <a href="#">see page 781</a> )	Function creates a cube based on the indicated four points - first three points (P1, P2, P3) define the cube base and the fourth point (PH) defines the height. The cube will be created in such a manner so that the PH point is connected by an edge with the P3 point. .
✳️	CreateCylinder ( <a href="#">see page 781</a> )	Function creates a cylinder based on the indicated four points. First three points define the cylinder base and the fourth point (PH) defines its height. The cylinder will be created in such a manner so that the fourth point (PH) is a center of the second base of the cylinder. .
✳️	CreateOnFiniteElems ( <a href="#">see page 781</a> )	Function creates one or more objects in such a manner so that they cover the indicated list of finite elements. Function returns the collection of the created object numbers. .
✳️	CreatePolyline ( <a href="#">see page 782</a> )	Function creates a polyline of the specified user-defined number based on the indicated points. .
✳️	CreateSolid ( <a href="#">see page 782</a> )	Function creates a solid based on the specified list of surface objects.
✳️	EndMultiOperation ( <a href="#">see page 782</a> )	Function updates definitions of objects in a structure. It should be called up only if BeginMultiOperation ( <a href="#">see page 778</a> ()) function has been called up earlier. .
✳️	FindWithId ( <a href="#">see page 783</a> )	Function returns a number of the object with the specified unique identifier. If the object is not found, then, zero value is returned (0). Available since version 1.7.
✳️	GetAnalyzeTTMethodEnabled ( <a href="#">see page 783</a> )	Function returns selection of objects taking into consideration in the triangular / trapezoidal distribution method.
✳️	GetFiniteElemsData ( <a href="#">see page 783</a> )	Function returns finite element data for the determined object selection.
✳️	GetHost ( <a href="#">see page 784</a> )	Function returns the number of host object for the given object.
✳️	GetHostedObjects ( <a href="#">see page 784</a> )	Function returns a list of objects for which the given object is a host.
✳️	GetName ( <a href="#">see page 784</a> )	Function returns a name of the object of the given number.
✳️	GetNameTemplate ( <a href="#">see page 785</a> )	Function returns the name pattern of an object of the specified number.
✳️	GetStructuralType ( <a href="#">see page 785</a> )	Function returns the structure object type for an object with the specified number.

	GetUniqueld ( <a href="#">see page 785</a> )	Function returns a unique identifier for an object with the user-specified number.
	IsVolume ( <a href="#">see page 786</a> )	Function returns the True value, if the indicated object is a solid. .
	SetAnalyzeTTMethod ( <a href="#">see page 786</a> )	Function allows for adding/removing objects to/from list of objects taken into consideration in triangular/trapezoidal distribution method.
	SetHost ( <a href="#">see page 786</a> )	Function sets the number of host object for the given object.
	SetHostedObjects ( <a href="#">see page 787</a> )	Function sets a list of objects for which the given object is to be a host.
	SetNameTemplate ( <a href="#">see page 787</a> )	Function sets a name pattern for the specified object.
	SetStructuralType ( <a href="#">see page 787</a> )	Function sets a structure object type for the specified object list.

### II.5.21.3.1 BeginMultiOperation

**C++**

```
HRESULT BeginMultiOperation();
```

**C#**

```
public void BeginMultiOperation();
```

**Visual Basic**

```
Public Sub BeginMultiOperation()
```

**Description**

Function enables speed-up of generation or modification of a whole object group through appropriate optimization of some operations. After completing object generation, EndMultiOperation ([see page 782](#)()) function should be called up to update definitions of objects in a structure. .

**Version**

Available since version 3.5.

### II.5.21.3.2 CalcArea

**C++**

```
HRESULT CalcArea(long _obj_num, long _part_idx = 0, double* ret);
```

**C#**

```
public double CalcArea(long _obj_num, long _part_idx = 0);
```

**Visual Basic**

```
Public Function CalcArea(_obj_num As long, Optional _part_idx As long = 0) As double
```

**Description**

Function calculates area of the indicated object component. If the object component index equals 0, function returns area of the whole object. Attention: Before calling the function, mesh of surface finite elements should be generated. .

**Version**

Available since version 4.5.

### II.5.21.3.3 CalcVol

**C++**

```
HRESULT CalcVol(long _obj_num, long _part_idx = 0, double* ret);
```

**C#**

```
public double CalcVol(long _obj_num, long _part_idx = 0);
```

**Visual Basic**

```
Public Function CalcVol(_obj_num As Long, Optional _part_idx As Long = 0) As Double
```

**Description**

Function calculates volume of the indicated solid or its component. If the object component index equals 0, function returns volume of the whole solid. Attention: Before calling the function, mesh of volumetric finite elements should be generated.

**Version**

Available since version 4.5.

**II.5.21.3.4 Create****C++**

```
HRESULT Create(long _num, IRobotObjObject** ret);
```

**C#**

```
public IRobotObjObject Create(long _num);
```

**Visual Basic**

```
Public Function Create(_num As Long) As IRobotObjObject
```

**Description**

The function creates and returns a new structure object with the user-defined number .

**II.5.21.3.5 CreateArc****C++**

```
HRESULT CreateArc(long _number, IRobotPointsArray* _points, IRobotGeoArcDefinitionMethod
_creation_type = I_GADM_CENTER_BEGIN_END);
```

**C#**

```
public void CreateArc(long _number, IRobotPointsArray _points, IRobotGeoArcDefinitionMethod
_creation_type = I_GADM_CENTER_BEGIN_END);
```

**Visual Basic**

```
Public Sub CreateArc(_number As Long, ByRef _points As IRobotPointsArray, Optional
_creation_type As IRobotGeoArcDefinitionMethod = I_GADM_CENTER_BEGIN_END)
```

**Description**

Function creates an arc based on the indicated points. .

**Version**

Available since version 3.

**II.5.21.3.6 CreateCircle****C++**

```
HRESULT CreateCircle(long _number, IRobotPointsArray* _points);
```

**C#**

```
public void CreateCircle(long _number, IRobotPointsArray _points);
```

**Visual Basic**

```
Public Sub CreateCircle(_number As long, ByRef _points As IRobotPointsArray)
```

**Description**

Function creates a circle of the specified user-defined number based on three points. .

**Version**

Available since version 3.

**II.5.21.3.7 CreateCone****C++**

```
HRESULT CreateCone(long _number, IRobotPointsArray* _points, VARIANT_BOOL _has_base, long
_divisions, long _sides, VARIANT_BOOL _is_volume = true, VARIANT_BOOL _has_top = true,
IRobotObjObject** ret);
```

**C#**

```
public IRobotObjObject CreateCone(long _number, IRobotPointsArray _points, bool _has_base,
long _divisions, long _sides, bool _is_volume = true, bool _has_top = true);
```

**Visual Basic**

```
Public Function CreateCone(_number As long, ByRef _points As IRobotPointsArray, _has_base
As Boolean, _divisions As long, _sides As long, Optional _is_volume As Boolean = true,
Optional _has_top As Boolean = true) As IRobotObjObject
```

**Description**

Function creates a cone based on the indicated four points. First three points define the cone base and the fourth point defines position of the vertex. To create a truncated cone the top radius should be provided as a fifth defining point .

**Version**

Available since version 3.

**II.5.21.3.8 CreateContour****C++**

```
HRESULT CreateContour(long _number, IRobotPointsArray* _points);
```

**C#**

```
public void CreateContour(long _number, IRobotPointsArray _points);
```

**Visual Basic**

```
Public Sub CreateContour(_number As long, ByRef _points As IRobotPointsArray)
```

**Description**

Function creates a contour based on the specified points. .

**Version**

Available since version 3.

**II.5.21.3.9 CreateCube****C++**

```
HRESULT CreateCube(long _number, IRobotPointsArray* _points, VARIANT_BOOL _has_base,
VARIANT_BOOL _has_top, long _divisions, VARIANT_BOOL _is_volume = true, IRobotObjObject** ret);
```

**C#**

```
public IRobotObjObject CreateCube(long _number, IRobotPointsArray _points, bool _has_base,
bool _has_top, long _divisions, bool _is_volume = true);
```

**Visual Basic**

```
Public Function CreateCube(_number As long, ByRef _points As IRobotPointsArray, _has_base
As Boolean, _has_top As Boolean, _divisions As long, Optional _is_volume As Boolean = true)
As IRobotObjObject
```

**Description**

Function creates a cube based on the indicated four points - first three points (P1, P2, P3) define the cube base and the fourth point (PH) defines the height. The cube will be created in such a manner so that the PH point is connected by an edge with the P3 point. .

**Version**

Available since version 3.

**II.5.21.3.10 CreateCylinder****C++**

```
HRESULT CreateCylinder(long _number, IRobotPointsArray* _points, VARIANT_BOOL _has_base,
VARIANT_BOOL _has_top, long _divisions, long _sides, VARIANT_BOOL _is_volume = true,
IRobotObjObject** ret);
```

**C#**

```
public IRobotObjObject CreateCylinder(long _number, IRobotPointsArray _points, bool
_has_base, bool _has_top, long _divisions, long _sides, bool _is_volume = true);
```

**Visual Basic**

```
Public Function CreateCylinder(_number As long, ByRef _points As IRobotPointsArray,
_has_base As Boolean, _has_top As Boolean, _divisions As long, _sides As long, Optional
_is_volume As Boolean = true) As IRobotObjObject
```

**Description**

Function creates a cylinder based on the indicated four points. First three points define the cylinder base and the fourth point (PH) defines its height. The cylinder will be created in such a manner so that the fourth point (PH) is a center of the second base of the cylinder. .

**Version**

Available since version 3.

**II.5.21.3.11 CreateOnFiniteElems****C++**

```
HRESULT CreateOnFiniteElems(BSTR _finite_elems, long _first_number,
IRobotNumbersCollection** ret);
```

**C#**

```
public IRobotNumbersCollection CreateOnFiniteElems(String _finite_elems, long
_first_number);
```

**Visual Basic**

```
Public Function CreateOnFiniteElems(_finite_elems As String, _first_number As long) As
IRobotNumbersCollection
```

**Description**

Function creates one or more objects in such a manner so that they cover the indicated list of finite elements. Function

returns the collection of the created object numbers. .

#### Version

Available since version 3.

### II.5.21.3.12 CreatePolyline

#### C++

```
HRESULT CreatePolyline(long _number, IRobotPointsArray* _points);
```

#### C#

```
public void CreatePolyline(long _number, IRobotPointsArray _points);
```

#### Visual Basic

```
Public Sub CreatePolyline(_number As long, ByRef _points As IRobotPointsArray)
```

#### Description

Function creates a polyline of the specified user-defined number based on the indicated points. .

#### Version

Available since version 3.

### II.5.21.3.13 CreateSolid

#### C++

```
HRESULT CreateSolid(long _number, BSTR _objlist, IRobotObjObject** ret);
```

#### C#

```
public IRobotObjObject CreateSolid(long _number, String _objlist);
```

#### Visual Basic

```
Public Function CreateSolid(_number As long, _objlist As String) As IRobotObjObject
```

#### Description

Function creates a solid based on the specified list of surface objects.

#### Version

Available since version 8.2.

### II.5.21.3.14 EndMultiOperation

#### C++

```
HRESULT EndMultiOperation();
```

#### C#

```
public void EndMultiOperation();
```

#### Visual Basic

```
Public Sub EndMultiOperation()
```

#### Description

Function updates definitions of objects in a structure. It should be called up only if BeginMultiOperation ( see page 778)() function has been called up earlier. .

**Version**

Available since version 3.5.

**II.5.21.3.15 FindWithId****C++**

```
HRESULT FindWithId(long _unique_id, long* ret);
```

**C#**

```
public long FindWithId(long _unique_id);
```

**Visual Basic**

```
Public Function FindWithId(_unique_id As long) As long
```

**Description**

Function returns a number of the object with the specified unique identifier. If the object is not found, then, zero value is returned (0). Available since version 1.7.

**II.5.21.3.16 GetAnalyzeTTMethodEnabled****C++**

```
HRESULT GetAnalyzeTTMethodEnabled(IRobotSelection** ret);
```

**C#**

```
public IRobotSelection GetAnalyzeTTMethodEnabled();
```

**Visual Basic**

```
Public Function GetAnalyzeTTMethodEnabled() As IRobotSelection
```

**Description**

Function returns selection of objects taking into consideration in the triangular / trapezoidal distribution method.

**Version**

Available since version 11.

**II.5.21.3.17 GetFiniteElemsData****C++**

```
HRESULT GetFiniteElemsData(BSTR _obj_sel_txt, IRobotFiniteElementDataSet* _ret_data);
```

**C#**

```
public void GetFiniteElemsData(String _obj_sel_txt, IRobotFiniteElementDataSet _ret_data);
```

**Visual Basic**

```
Public Sub GetFiniteElemsData(_obj_sel_txt As String, ByRef _ret_data As IRobotFiniteElementDataSet)
```

**Description**

Function returns finite element data for the determined object selection.

**Version**

Available since version 9.

### II.5.21.3.18 GetHost

#### C++

```
HRESULT GetHost(long _obj_num, long* ret);
```

#### C#

```
public long GetHost(long _obj_num);
```

#### Visual Basic

```
Public Function GetHost(_obj_num As long) As long
```

#### Description

Function returns the number of host object for the given object.

#### Version

Available since version 11.

### II.5.21.3.19 GetHostedObjects

#### C++

```
HRESULT GetHostedObjects(long _obj_num, IRobotSelection** ret);
```

#### C#

```
public IRobotSelection GetHostedObjects(long _obj_num);
```

#### Visual Basic

```
Public Function GetHostedObjects(_obj_num As long) As IRobotSelection
```

#### Description

Function returns a list of objects for which the given object is a host.

#### Version

Available since version 11.

### II.5.21.3.20 GetName

#### C++

```
HRESULT GetName(long _obj_num, BSTR* ret);
```

#### C#

```
public String GetName(long _obj_num);
```

#### Visual Basic

```
Public Function GetName(_obj_num As long) As String
```

#### Description

Function returns a name of the object of the given number.

#### Version

Available since version 7.5.

### II.5.21.3.21 GetNameTemplate

#### C++

```
HRESULT GetNameTemplate(long _obj_num, BSTR* ret);
```

#### C#

```
public String GetNameTemplate(long _obj_num);
```

#### Visual Basic

```
Public Function GetNameTemplate(_obj_num As long) As String
```

#### Description

Function returns the name pattern of an object of the specified number.

#### Version

Available since version 7.5.

### II.5.21.3.22 GetStructuralType

#### C++

```
HRESULT GetStructuralType(long _obj_num, IRobotObjectStructuralType* ret);
```

#### C#

```
public IRobotObjectStructuralType GetStructuralType(long _obj_num);
```

#### Visual Basic

```
Public Function GetStructuralType(_obj_num As long) As IRobotObjectStructuralType
```

#### Description

Function returns the structure object type for an object with the specified number.

#### Version

Available since version 9.7.

### II.5.21.3.23 GetUniqueId

#### C++

```
HRESULT GetUniqueId(long _obj_num, long* ret);
```

#### C#

```
public long GetUniqueId(long _obj_num);
```

#### Visual Basic

```
Public Function GetUniqueId(_obj_num As long) As long
```

#### Description

Function returns a unique identifier for an object with the user-specified number.

#### Version

Available since version 8.2.

### II.5.21.3.24 IsVolume

#### C++

```
HRESULT IsVolume(long _obj_num, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsVolume(long _obj_num);
```

#### Visual Basic

```
Public Function IsVolume(_obj_num As long) As Boolean
```

#### Description

Function returns the True value, if the indicated object is a solid. .

#### Version

Available since version 4.5.

### II.5.21.3.25 SetAnalyzeTTMethod

#### C++

```
HRESULT SetAnalyzeTTMethod(BSTR _objs, VARIANT_BOOL _analyze);
```

#### C#

```
public void SetAnalyzeTTMethod(String _objs, bool _analyze);
```

#### Visual Basic

```
Public Sub SetAnalyzeTTMethod(_objs As String, _analyze As Boolean)
```

#### Description

Function allows for adding/removing objects to/from list of objects taken into consideration in triangular/trapezoidal distribution method.

#### Version

Available since version 11.

### II.5.21.3.26 SetHost

#### C++

```
HRESULT SetHost(long _obj_num, long _host_num);
```

#### C#

```
public void SetHost(long _obj_num, long _host_num);
```

#### Visual Basic

```
Public Sub SetHost(_obj_num As long, _host_num As long)
```

#### Description

Function sets the number of host object for the given object.

#### Version

Available since version 11.

### II.5.21.3.27 SetHostedObjects

#### C++

```
HRESULT SetHostedObjects(long _obj_num, IRobotSelection* _hosted_objs);
```

#### C#

```
public void SetHostedObjects(long _obj_num, IRobotSelection _hosted_objs);
```

#### Visual Basic

```
Public Sub SetHostedObjects(_obj_num As long, ByRef _hosted_objs As IRobotSelection)
```

#### Description

Function sets a list of objects for which the given object is to be a host.

#### Version

Available since version 11.

### II.5.21.3.28 SetNameTemplate

#### C++

```
HRESULT SetNameTemplate(long _obj_num, BSTR _name_tmpl);
```

#### C#

```
public void SetNameTemplate(long _obj_num, String _name_tmpl);
```

#### Visual Basic

```
Public Sub SetNameTemplate(_obj_num As long, _name_tmpl As String)
```

#### Description

Function sets a name pattern for the specified object.

#### Version

Available since version 7.5.

### II.5.21.3.29 SetStructuralType

#### C++

```
HRESULT SetStructuralType(BSTR _obj_sel, IRobotObjectStructuralType _str_type);
```

#### C#

```
public void SetStructuralType(String _obj_sel, IRobotObjectStructuralType _str_type);
```

#### Visual Basic

```
Public Sub SetStructuralType(_obj_sel As String, _str_type As IRobotObjectStructuralType)
```

#### Description

Function sets a structure object type for the specified object list.

#### Version

Available since version 9.7.

## II.5.22 IRobotObjLocalXDirDefinitionType

### C++

```
enum IRobotObjLocalXDirDefinitionType;
```

### C#

```
public enum IRobotObjLocalXDirDefinitionType;
```

### Visual Basic

```
Public Enum IRobotObjLocalXDirDefinitionType
```

### Members

Members	Description
I_OLXDDT_UNDEFINED = 0	Local X axis direction is not defined (not significant) Available since version 1.7.
I_OLXDDT_CARTESIAN = 1	Definition of local X axis direction in the Cartesian coordinate system Available since version 1.7.
I_OLXDDT_POLAR = 2	Definition of local X axis direction in the polar coordinate system Available since version 1.7.

### Description

Type of direction definition for the local X axis of the plate. .

## II.5.23 IRobotObjPartMain

### Class Hierarchy

### C++

```
interface IRobotObjPartMain : IRobotObjPart;
```

### C#

```
public interface IRobotObjPartMain : IRobotObjPart;
```

### Visual Basic

```
Public Interface IRobotObjPartMain
```

### Description

Main object component. It contains definition of the base geometry and geometric transformations which result in the geometry of the entire object. All object edges are also saved in the main component.

### II.5.23.1 IRobotObjPartMain Members

The following tables list the members exposed by IRobotObjPartMain.

### Public Fields

	Name	Description
❖	Attribs (❑ see page 763)	Attributes of the object component .
❖	CurveDiv (❑ see page 789)	Definition of object discretization Available since version 2.0. .
❖	DefPoints (❑ see page 790)	Collection of points defining the base object geometry Available since version 2.0. .
❖	Edges (❑ see page 790)	Collection of all object edges Available since version 2.0. .
❖	FiniteElems (❑ see page 790)	List of finite elements assigned to the main object component.
❖	Geometry (❑ see page 790)	Definition of the base object geometry Available since version 2.0. .
❖	ModelPoints (❑ see page 791)	Characteristic points used for contour discretization.

◆	Modifications ( <a href="#">see page 791</a> )	Collection of transformations of the base object geometry Available since version 2.0..
◆	Nodes ( <a href="#">see page 791</a> )	
◆	Type ( <a href="#">see page 763</a> )	Object component type Available since version 2.0.

**Public Methods**

	Name	Description
◆	AddModification ( <a href="#">see page 792</a> )	Function adds the indicated transformation to the collection. Available since version 2.0..
◆	CalcArea ( <a href="#">see page 792</a> )	Function calculates area of the object component..
◆	CalcVol ( <a href="#">see page 792</a> )	Function calculates volume of the solid component..
◆	ClearModifications ( <a href="#">see page 793</a> )	Function deletes all transformations from the collection of transformations. Available since version 2.0..
◆	GetGeometry ( <a href="#">see page 763</a> )	Function returns definition of the object component geometry. Available since version 2.0.

**II.5.23.2 IRobotObjPartMain Fields**

The fields of the IRobotObjPartMain class are listed here.

**Public Fields**

	Name	Description
◆	CurveDiv ( <a href="#">see page 789</a> )	Definition of object discretization Available since version 2.0..
◆	DefPoints ( <a href="#">see page 790</a> )	Collection of points defining the base object geometry Available since version 2.0..
◆	Edges ( <a href="#">see page 790</a> )	Collection of all object edges Available since version 2.0..
◆	FiniteElems ( <a href="#">see page 790</a> )	List of finite elements assigned to the main object component.
◆	Geometry ( <a href="#">see page 790</a> )	Definition of the base object geometry Available since version 2.0..
◆	ModelPoints ( <a href="#">see page 791</a> )	Characteristic points used for contour discretization.
◆	Modifications ( <a href="#">see page 791</a> )	Collection of transformations of the base object geometry Available since version 2.0..
◆	Nodes ( <a href="#">see page 791</a> )	

**II.5.23.2.1 CurveDiv****C++**

```
HRESULT get_CurveDiv(IRobotGeoCurveDiv**);
HRESULT put_CurveDiv(IRobotGeoCurveDiv*);
```

**C#**

```
public IRobotGeoCurveDiv CurveDiv { get; set; }
```

**Visual Basic**

```
Public CurveDiv As IRobotGeoCurveDiv
```

**Description**

Definition of object discretization Available since version 2.0..

**II.5.23.2.2 DefPoints****C++**

```
HRESULT get_DefPoints(IRobotGeoPoint3DCollection**);
```

**C#**

```
public IRobotGeoPoint3DCollection DefPoints { get; }
```

**Visual Basic**

```
Public ReadOnly DefPoints As IRobotGeoPoint3DCollection
```

**Description**

Collection of points defining the base object geometry Available since version 2.0. .

**II.5.23.2.3 Edges****C++**

```
HRESULT get_Edges(IRobotObjEdgeCollection**);
```

**C#**

```
public IRobotObjEdgeCollection Edges { get; }
```

**Visual Basic**

```
Public ReadOnly Edges As IRobotObjEdgeCollection
```

**Description**

Collection of all object edges Available since version 2.0. .

**II.5.23.2.4 FiniteElems****C++**

```
HRESULT get_FiniteElems(BSTR*);
```

**C#**

```
public String FiniteElems { get; }
```

**Visual Basic**

```
Public ReadOnly FiniteElems As String
```

**Description**

List of finite elements assigned to the main object component.

**Version**

Available since version 2.5.

**II.5.23.2.5 Geometry****C++**

```
HRESULT get_Geometry(IRobotGeoObject**);
HRESULT put_Geometry(IRobotGeoObject*);
```

**C#**

```
public IRobotGeoObject Geometry { get; set; }
```

**Visual Basic**

```
Public Geometry As IRobotGeoObject
```

**Description**

Definition of the base object geometry Available since version 2.0. .

**II.5.23.2.6 ModelPoints****C++**

```
HRESULT get_ModelPoints(IRobotGeoPoint3DCollection**);
```

**C#**

```
public IRobotGeoPoint3DCollection ModelPoints { get; }
```

**Visual Basic**

```
Public ReadOnly ModelPoints As IRobotGeoPoint3DCollection
```

**Description**

Characteristic points used for contour discretization.

**Version**

Available since version 3.

**II.5.23.2.7 Modifications****C++**

```
HRESULT get_Modifications(IRobotObjModificationCollection**);
```

**C#**

```
public IRobotObjModificationCollection Modifications { get; }
```

**Visual Basic**

```
Public ReadOnly Modifications As IRobotObjModificationCollection
```

**Description**

Collection of transformations of the base object geometry Available since version 2.0. .

**II.5.23.2.8 Nodes****C++**

```
HRESULT get_Nodes(BSTR*);
```

**C#**

```
public String Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As String
```

**Version**

Available since version 4.

**II.5.23.3 IRobotObjPartMain Methods**

The methods of the IRobotObjPartMain class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	AddModification ( [ see page 792 ] )	Function adds the indicated transformation to the collection. Available since version 2.0. .
≡	CalcArea ( [ see page 792 ] )	Function calculates area of the object component. .
≡	CalcVol ( [ see page 792 ] )	Function calculates volume of the solid component..
≡	ClearModifications ( [ see page 793 ] )	Function deletes all transformations from the collection of transformations. Available since version 2.0. .

**II.5.23.3.1 AddModification****C++**

```
HRESULT AddModification(IRobotObjModification* _mod_def);
```

**C#**

```
public void AddModification(IRobotObjModification _mod_def);
```

**Visual Basic**

```
Public Sub AddModification(ByRef _mod_def As IRobotObjModification)
```

**Description**

Function adds the indicated transformation to the collection. Available since version 2.0. .

**II.5.23.3.2 CalcArea****C++**

```
HRESULT CalcArea(double* ret);
```

**C#**

```
public double CalcArea();
```

**Visual Basic**

```
Public Function CalcArea() As double
```

**Description**

Function calculates area of the object component. .

**Version**

Available since version 4.5.

**II.5.23.3.3 CalcVol****C++**

```
HRESULT CalcVol(double* ret);
```

**C#**

```
public double CalcVol();
```

**Visual Basic**

```
Public Function CalcVol() As double
```

**Description**

Function calculates volume of the solid component. .

**Version**

Available since version 4.5.

**II.5.23.3.4 ClearModifications****C++**

```
HRESULT ClearModifications();
```

**C#**

```
public void ClearModifications();
```

**Visual Basic**

```
Public Sub ClearModifications()
```

## Description

Function deletes all transformations from the collection of transformations. Available since version 2.0. .

## II.5.24 IRobotObjPartReference

### Class Hierarchy

#### C++

```
interface IRobotObjPartReference : IRobotObjPart;
```

#### C#

```
public interface IRobotObjPartReference : IRobotObjPart;
```

### Visual Basic

```
Public Interface IRobotObjPartReference
```

### Description

Object component definition which is the reference of the main component generated due to transformations. .

## II.5.24.1 IRobotObjPartReference Members

The following tables list the members exposed by IRobotObjPartReference.

### Public Fields

	Name	Description
❖	Attribs ( <a href="#">see page 763</a> )	Attributes of the object component .
❖	FiniteElems ( <a href="#">see page 794</a> )	List of finite elements assigned to this object component.
❖	Nodes ( <a href="#">see page 794</a> )	
❖	Type ( <a href="#">see page 763</a> )	Object component type Available since version 2.0.

### Public Methods

	Name	Description
❖	CalcArea ( <a href="#">see page 794</a> )	Function calculates area of object component. .
❖	CalcVol ( <a href="#">see page 795</a> )	Function calculates volume of solid component. .
❖	GetGeometry ( <a href="#">see page 763</a> )	Function returns defintion of the object component geometry. Available since version 2.0.

## II.5.24.2 IRobotObjPartReference Fields

The fields of the IRobotObjPartReference class are listed here.

### Public Fields

	Name	Description
❖	FiniteElems ( <a href="#">see page 794</a> )	List of finite elements assigned to this object component.
❖	Nodes ( <a href="#">see page 794</a> )	

## II.5.24.2.1 FiniteElems

#### C++

```
HRESULT get_FiniteElems(BSTR* );
```

#### C#

```
public String FiniteElems { get; }
```

**Visual Basic**

```
Public ReadOnly FiniteElems As String
```

**Description**

List of finite elements assigned to this object component.

**Version**

Available since version 2.5.

**II.5.24.2.2 Nodes****C++**

```
HRESULT get_Nodes(BSTR* );
```

**C#**

```
public String Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As String
```

**Version**

Available since version 4.

**II.5.24.3 IRobotObjPartReference Methods**

The methods of the IRobotObjPartReference class are listed here.

**Public Methods**

	Name	Description
◆	CalcArea (see page 794)	Function calculates area of object component. .
◆	CalcVol (see page 795)	Function calculates volume of solid component. .

**II.5.24.3.1 CalcArea****C++**

```
HRESULT CalcArea(double* ret);
```

**C#**

```
public double CalcArea();
```

**Visual Basic**

```
Public Function CalcArea() As double
```

**Description**

Function calculates area of object component. .

**Version**

Available since version 4.5.

**II.5.24.3.2 CalcVol****C++**

```
HRESULT CalcVol(double* ret);
```

**C#**

```
public double CalcVol();
```

**Visual Basic**

```
Public Function CalcVol() As double
```

**Description**

Function calculates volume of solid component. .

**Version**

Available since version 4.5.

**II.5.25 IRobotObjPart2****Class Hierarchy****C++**

```
interface IRobotObjPart2 : IRobotObjPart;
```

**C#**

```
public interface IRobotObjPart2 : IRobotObjPart;
```

**Visual Basic**

```
Public Interface IRobotObjPart2
```

**Description**

Extended definition of the object component. .

**Version**

Available since version 2.5.

**II.5.25.1 IRobotObjPart2 Members**

The following tables list the members exposed by IRobotObjPart2.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Attribs (↗ see page 763)	Attributes of the object component .
❖	FiniteElems (↗ see page 796)	Text containing the list of finite elements assigned to this object component .
❖	Nodes (↗ see page 796)	
❖	Type (↗ see page 763)	Object component type Available since version 2.0.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	CalcArea (↗ see page 797)	Function calculates area of the object component.
❖	CalcVol (↗ see page 797)	Function calculates volume of the solid component.
❖	GetGeometry (↗ see page 763)	Function returns defintion of the object component geometry. Available since version 2.0.

**II.5.25.2 IRobotObjPart2 Fields**

The fields of the IRobotObjPart2 class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	FiniteElems (↗ see page 796)	Text containing the list of finite elements assigned to this object component .

	Nodes ( <a href="#">see page 796</a> )	
--	----------------------------------------	--

### II.5.25.2.1 FiniteElems

#### C++

```
HRESULT get_FiniteElems(BSTR* );
```

#### C#

```
public String FiniteElems { get; }
```

#### Visual Basic

```
Public ReadOnly FiniteElems As String
```

#### Description

Text containing the list of finite elements assigned to this object component .

#### Version

Available since version 2.5.

### II.5.25.2.2 Nodes

#### C++

```
HRESULT get_Nodes(BSTR* );
```

#### C#

```
public String Nodes { get; }
```

#### Visual Basic

```
Public ReadOnly Nodes As String
```

#### Version

Available since version 4.

### II.5.25.3 IRobotObjPart2 Methods

The methods of the IRobotObjPart2 class are listed here.

#### Public Methods

	Name	Description
	CalcArea ( <a href="#">see page 797</a> )	Function calculates area of the object component.
	CalcVol ( <a href="#">see page 797</a> )	Function calculates volume of the solid component.

### II.5.25.3.1 CalcArea

#### C++

```
HRESULT CalcArea(double* ret);
```

#### C#

```
public double CalcArea();
```

#### Visual Basic

```
Public Function CalcArea() As double
```

#### Description

Function calculates area of the object component.

**Version**

Available since version 4.5.

**II.5.25.3.2 CalcVol****C++**

```
HRESULT CalcVol(double* ret);
```

**C#**

```
public double CalcVol();
```

**Visual Basic**

```
Public Function CalcVol() As double
```

**Description**

Function calculates volume of the solid component.

**Version**

Available since version 4.5.

**II.5.26 IRobotObjMesh****Class Hierarchy****C++**

```
interface IRobotObjMesh : IDispatch;
```

**C#**

```
public interface IRobotObjMesh;
```

**Visual Basic**

```
Public Interface IRobotObjMesh
```

**Description**

Interface allowing management of the finite element mesh.

**Version**

Available since version 3.

**II.5.26.1 IRobotObjMesh Members**

The following tables list the members exposed by IRobotObjMesh.

**Public Fields**

	Name	Description
❖	Freeze (↗ see page 798)	Flag enabling mesh freezing/unfreezing.
❖	IsGenerated (↗ see page 798)	Flag indicating if the finite element mesh has been generated.
❖	Params (↗ see page 799)	Parameters of mesh generation.

**Public Methods**

	Name	Description
❖	Generate (↗ see page 799)	Function generates a finite element mesh.
❖	GetBasePoints (↗ see page 800)	Function returns base points for a finite element mesh.
❖	GetQuality (↗ see page 800)	Function checks a quality of a finite element mesh.

	Remove ( <a href="#">see page 800</a> )	Function removes a finite element mesh.
	SetBasePoints ( <a href="#">see page 801</a> )	Function sets base points for a finite element mesh.

## II.5.26.2 IRobotObjMesh Fields

The fields of the IRobotObjMesh class are listed here.

### Public Fields

	Name	Description
	Freeze ( <a href="#">see page 798</a> )	Flag enabling mesh freezing/unfreezing.
	IsGenerated ( <a href="#">see page 798</a> )	Flag indicating if the finite element mesh has been generated.
	Params ( <a href="#">see page 799</a> )	Parameters of mesh generation.

### II.5.26.2.1 Freeze

#### C++

```
HRESULT get_Freeze(VARIANT_BOOL* );
HRESULT put_Freeze(VARIANT_BOOL);
```

#### C#

```
public bool Freeze { get; set; }
```

#### Visual Basic

```
Public ReadOnly Freeze As Boolean
```

#### Description

Flag enabling mesh freezing/unfreezing.

#### Version

Available since version 3.

### II.5.26.2.2 IsGenerated

#### C++

```
HRESULT get_IsGenerated(VARIANT_BOOL* );
```

#### C#

```
public bool IsGenerated { get; }
```

#### Visual Basic

```
Public ReadOnly IsGenerated As Boolean
```

#### Description

Flag indicating if the finite element mesh has been generated.

#### Version

Available since version 3.

### II.5.26.2.3 Params

#### C++

```
HRESULT get_Params(IRobotMeshParams** );
```

#### C#

```
public IRobotMeshParams Params { get; }
```

**Visual Basic**

```
Public ReadOnly Params As IRobotMeshParams
```

**Description**

Parameters of mesh generation.

**Version**

Available since version 3.

**II.5.26.3 IRobotObjMesh Methods**

The methods of the IRobotObjMesh class are listed here.

**Public Methods**

	Name	Description
💡	Generate ( [ see page 799 )	Function generates a finite element mesh.
💡	GetBasePoints ( [ see page 800 )	Function returns base points for a finite element mesh.
💡	GetQuality ( [ see page 800 )	Function checks a quality of a finite element mesh.
💡	Remove ( [ see page 800 )	Function removes a finite element mesh.
💡	SetBasePoints ( [ see page 801 )	Function sets base points for a finite element mesh.

**II.5.26.3.1 Generate****C++**

```
HRESULT Generate();
```

**C#**

```
public void Generate();
```

**Visual Basic**

```
Public Sub Generate()
```

**Description**

Function generates a finite element mesh.

**Version**

Available since version 3.

**II.5.26.3.2 GetBasePoints****C++**

```
HRESULT GetBasePoints(IRobotNumbersArray** ret);
```

**C#**

```
public IRobotNumbersArray GetBasePoints();
```

**Visual Basic**

```
Public Function GetBasePoints() As IRobotNumbersArray
```

**Description**

Function returns base points for a finite element mesh.

**Version**

Available since version 3.

### II.5.26.3.3 GetQuality

#### C++

```
HRESULT GetQuality(double* _out_q1, double* _out_q2, BSTR _out_fes, double _in_precision = 0.5, BSTR _in_objs = "");
```

#### C#

```
public void GetQuality(double* _out_q1, double* _out_q2, String _out_fes, double _in_precision = 0.5, String _in_objs = "");
```

#### Visual Basic

```
Public Sub GetQuality(ByRef _out_q1 As double*, ByRef _out_q2 As double*, ByRef _out_fes As String, Optional _in_precision As double = 0.5, Optional _in_objs As String = "")
```

#### Description

Function checks a quality of a finite element mesh.

#### Version

Available since version 3.

### II.5.26.3.4 Remove

#### C++

```
HRESULT Remove();
```

#### C#

```
public void Remove();
```

#### Visual Basic

```
Public Sub Remove()
```

#### Description

Function removes a finite element mesh.

#### Version

Available since version 3.

### II.5.26.3.5 SetBasePoints

#### C++

```
HRESULT SetBasePoints(IRobotNumbersArray* _model_points);
```

#### C#

```
public void SetBasePoints(IRobotNumbersArray _model_points);
```

#### Visual Basic

```
Public Sub SetBasePoints(ByRef _model_points As IRobotNumbersArray)
```

#### Description

Function sets base points for a finite element mesh.

#### Version

Available since version 3.

## II.5.27 IRobotObjEdgeSelection

### Class Hierarchy

#### C++

```
interface IRobotObjEdgeSelection : IDispatch;
```

#### C#

```
public interface IRobotObjEdgeSelection;
```

### Visual Basic

```
Public Interface IRobotObjEdgeSelection
```

### Description

An interface which describes a selection of edges of panels and other surface objects.

### Version

Available since version 15.2.

## II.5.27.1 IRobotObjEdgeSelection Members

The following tables list the members exposed by IRobotObjEdgeSelection.

### Public Fields

	Name	Description
◆	Count (see page 802)	A number of edges which are described by this selection.

### Public Methods

	Name	Description
◆	FromText (see page 802)	Initialize this selection of edges from given text representation.
◆	Get (see page 803)	This method returns information about the edge with given index within the selection. It allows iterate through all the edges in the selection when calling this method with and index parameter from1 to Count (see page 802).
◆	ToText (see page 803)	Return a text representation of this selection of edges.

## II.5.27.2 IRobotObjEdgeSelection Fields

The fields of the IRobotObjEdgeSelection class are listed here.

### Public Fields

	Name	Description
◆	Count (see page 802)	A number of edges which are described by this selection.

## II.5.27.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

### Visual Basic

```
Public ReadOnly Count As Long
```

### Description

A number of edges which are described by this selection.

## Version

Available since version 15.2.

### II.5.27.3 IRobotObjEdgeSelection Methods

The methods of the IRobotObjEdgeSelection class are listed here.

#### Public Methods

	Name	Description
💡	FromText (see page 802)	Initialize this selection of edges from given text representation.
💡	Get (see page 803)	This method returns information about the edge with given index within the selection. It allows iterate through all the edges in the selection when calling this method with and index parameter from1 to Count (see page 802).
💡	ToText (see page 803)	Return a text representation of this selection of edges.

#### II.5.27.3.1 FromText

##### C++

```
HRESULT FromText(BSTR _sel_text);
```

##### C#

```
public void FromText(String _sel_text);
```

##### Visual Basic

```
Public Sub FromText(_sel_text As String)
```

#### Description

Initialize this selection of edges from given text representation.

## Version

Available since version 15.2.

#### II.5.27.3.2 Get

##### C++

```
HRESULT Get(long _idx, long* _obj_num, long* _edge_idx);
```

##### C#

```
public void Get(long _idx, long* _obj_num, long* _edge_idx);
```

##### Visual Basic

```
Public Sub Get(_idx As long, ByRef _obj_num As long*, ByRef _edge_idx As long*)
```

#### Description

This method returns information about the edge with given index within the selection. It allows iterate through all the edges in the selection when calling this method with and index parameter from1 to Count (see page 802).

## Version

Available since version 15.2.

#### II.5.27.3.3 ToText

##### C++

```
HRESULT ToText(BSTR* ret);
```

**C#**

```
public String ToText();
```

**Visual Basic**

```
Public Function ToText() As String
```

**Description**

Return a text representation of this selection of edges.

**Version**

Available since version 15.2.

## II.6 Finite elements

Available since version 2.5.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotFiniteElementType ( <a href="#">see page 831</a> )	Available types of finite elements.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotFiniteElement ( <a href="#">see page 828</a> )	Interface describing planar finite element. .
	IRobotFiniteElementNodes ( <a href="#">see page 831</a> )	Collection of nodes defining finite element.
	IRobotFiniteElementServer ( <a href="#">see page 834</a> )	Server of finite elements. .
	IRobotFiniteElementData ( <a href="#">see page 837</a> )	Interface gives access to finite element data.
	IRobotFiniteElementDataSet ( <a href="#">see page 840</a> )	Interface gives access to the finite element set.

### II.6.1 FE mesh generator

Available since version 3.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotMeshType ( <a href="#">see page 805</a> )	Types of generated finite element mesh (surface or volume).
	IRobotMeshForcingRatio ( <a href="#">see page 805</a> )	Forcing ratio of selected meshing parameter.
	IRobotMeshImplementDegree ( <a href="#">see page 806</a> )	Degrees of the selected method application while generating finite element mesh.
	IRobotMeshPanelDivType ( <a href="#">see page 806</a> )	Methods of contour division for the Coons method of finite element mesh generation.
	IRobotMeshSurfaceFEType ( <a href="#">see page 807</a> )	Finite element types applied while generating a surface element mesh. .
	IRobotMeshVolumetricFEType ( <a href="#">see page 807</a> )	Finite element types applied during generation of volumetric element mesh.
	IRobotMeshDelaunayType ( <a href="#">see page 808</a> )	Available variants of Delaunay's method.

	IRobotMeshMethodType ( <a href="#">see page 808</a> )	Acceptable meshing methods.
	IRobotMeshGenerationType ( <a href="#">see page 809</a> )	Methods of finite element mesh generation.
	IRobotMeshAccessType ( <a href="#">see page 827</a> )	Availability of interfaces for mesh generation.
	IRobotMeshRefinementType ( <a href="#">see page 827</a> )	Methods of finite element mesh refinement.

## Interfaces

	Name	Description
	IRobotMeshMethod ( <a href="#">see page 809</a> )	A set of parameters defining a mesh generation method and associated quantities. .
	IRobotMeshGeneration ( <a href="#">see page 811</a> )	Parameters of mesh generation.
	IRobotMeshCoonsParams ( <a href="#">see page 813</a> )	Parameters of Coons method.
	IRobotMeshSurfaceFiniteElems ( <a href="#">see page 815</a> )	Definition of surface finite elements applied during mesh generation. .
	IRobotMeshDelaunayParams ( <a href="#">see page 816</a> )	Parameters of Delaunay's method.
	IRobotMeshSurfaceParams ( <a href="#">see page 820</a> )	Parameters of the mesh generation using planar finite elements.
	IRobotMeshVolumeParams ( <a href="#">see page 823</a> )	Parameters of volumetric element mesh generation. .
	IRobotMeshParams ( <a href="#">see page 825</a> )	Parameters of the finite element mesh generation.

### II.6.1.1 IRobotMeshType

#### C++

```
enum IRobotMeshType;
```

#### C#

```
public enum IRobotMeshType;
```

#### Visual Basic

```
Public Enum IRobotMeshType
```

#### Members

Members	Description
I_MT_FINE = 2	Fine meshing type. Available since version 3.
I_MT_COARSE = 0	Coarse meshing type. Available since version 3.
I_MT_NORMAL = 1	Normal meshing type. Available since version 3.
I_MT_USER = 3	User-defined meshing type. Available since version 3.
I_MT_WALLS_COARSE = 4	Available since version 12.
I_MT_WALLS_NORMAL = 5	Available since version 12.
I_MT_FLOORS_COARSE = 6	Available since version 12.
I_MT_FLOORS_NORMAL = 7	Available since version 12.

**Description**

Types of generated finite element mesh (surface or volume).

**Version**

Available since version 3.

**II.6.1.2 IRobotMeshForcingRatio****C++**

```
enum IRobotMeshForcingRatio;
```

**C#**

```
public enum IRobotMeshForcingRatio;
```

**Visual Basic**

```
Public Enum IRobotMeshForcingRatio
```

**Members**

Members	Description
I_MFR_NONE = 1	Forcing ratio - none. Available since version 3.
I_MFR_ANY = 2	Any forcing ratio. Available since version 3.
I_MFR_PROPOSED = 3	Proposed forcing ratio. Available since version 3.
I_MFR_RECOMMENDED = 4	Recommended forcing ratio. Available since version 3.
I_MFR_FORCED = 5	Forced forcing ratio. Available since version 3.

**Description**

Forcing ratio of selected meshing parameter.

**Version**

Available since version 3.

**II.6.1.3 IRobotMeshImplementDegree****C++**

```
enum IRobotMeshImplementDegree;
```

**C#**

```
public enum IRobotMeshImplementDegree;
```

**Visual Basic**

```
Public Enum IRobotMeshImplementDegree
```

**Members**

Members	Description
I_MIR_OFTEN = 1	Available since version 3.
I_MIR_RARELY = 2	Available since version 3.
I_MIR_NEVER = 3	Available since version 3.

## Description

Degrees of the selected method application while generating finite element mesh.

## Version

Available since version 3.

### II.6.1.4 IRobotMeshPanelDivType

#### C++

```
enum IRobotMeshPanelDivType;
```

#### C#

```
public enum IRobotMeshPanelDivType;
```

## Visual Basic

```
Public Enum IRobotMeshPanelDivType
```

## Members

Members	Description
I_MPDT_TRIANG_IN_TRIANG = 1	Division allowing generation of triangular finite elements in a selected triangular contour. Available since version 3.
I_MPDT_TRIANG_AND_SQUARE_IN_TRIANG = 2	Division allowing generation of triangular and square finite elements in a selected triangular contour . Available since version 3.
I_MPDT_TRIANG_AND_TRAPEZ_IN_TRIANG = 3	Division allowing generation of triangular and rhombic finite elements in a selected triangular contour . Available since version 3.
I_MPDT_SQUARE_IN_RECT = 4	Division allowing generation of square finite elements in a selected rectangular contour . Available since version 3.
I_MPDT_TRIANG_IN_RECT = 5	Division allowing generation of triangular finite elements in a selected rectangular contour . Available since version 3.

## Description

Methods of contour division for the Coons method of finite element mesh generation.

## Version

Available since version 3.

### II.6.1.5 IRobotMeshSurfaceFEType

#### C++

```
enum IRobotMeshSurfaceFEType;
```

#### C#

```
public enum IRobotMeshSurfaceFEType;
```

## Visual Basic

```
Public Enum IRobotMeshSurfaceFEType
```

## Members

Members	Description
I_MSFET_3NODE_TRIANG = 1	3-node triangular surface finite elements . Available since version 3.
I_MSFET_4NODE_QUADRIL = 2	4-node quadrilateral surface finite elements . Available since version 3.
I_MSFET_6NODE_TRIANG = 3	6-node triangular surface finite elements . Available since version 3.
I_MSFET_8NODE_QUADRIL = 4	8-node quadrilateral surface finite elements . Available since version 3.

## Description

Finite element types applied while generating a surface element mesh. .

## Version

Available since version 3.

## II.6.1.6 IRobotMeshVolumetricFEType

### C++

```
enum IRobotMeshVolumetricFEType;
```

### C#

```
public enum IRobotMeshVolumetricFEType;
```

### Visual Basic

```
Public Enum IRobotMeshVolumetricFEType
```

## Members

Members	Description
I_MVFET_4NODE_TETRAHED = 1	4-node tetrahedral volumetric finite elements. Available since version 3.
I_MVFET_8NODE_HEXAHEX = 2	8-node hexahedral volumetric finite elements. Available since version 3.

## Description

Finite element types applied during generation of volumetric element mesh.

## Version

Available since version 3.

## II.6.1.7 IRobotMeshDelaunayType

### C++

```
enum IRobotMeshDelaunayType;
```

### C#

```
public enum IRobotMeshDelaunayType;
```

### Visual Basic

```
Public Enum IRobotMeshDelaunayType
```

## Members

Members	Description
I_MDT_DELAUNAY = 1	Standard Delaunay's method. Available since version 3.
I_MDT_KANG = 2	Kang's method; selection of this variant indicates that the finite element mesh will be generated only on a contour near emitters according to the adopted method parameters (H0, Hmax, and Q). Available since version 3.
I_MDT_DELAUNAY_AND_KANG = 3	Delaunay and Kang method; selection of this variant indicates that near emitters the finite element mesh will be generated by means of Kang's method, whereas on the remaining contour - according to Delaunay's method. Available since version 3.

## Description

Available variants of Delaunay's method.

## Version

Available since version 3.

## II.6.1.8 IRobotMeshMethodType

### C++

```
enum IRobotMeshMethodType;
```

### C#

```
public enum IRobotMeshMethodType;
```

### Visual Basic

```
Public Enum IRobotMeshMethodType
```

## Members

Members	Description
I_MMT_DELAUNAY = 1	Delaunay's method. Available since version 3.
I_MMT_COONS = 2	Coons method . Available since version 3.

## Description

Acceptable meshing methods.

## Version

Available since version 3.

## II.6.1.9 IRobotMeshGenerationType

### C++

```
enum IRobotMeshGenerationType;
```

### C#

```
public enum IRobotMeshGenerationType;
```

## Visual Basic

```
Public Enum IRobotMeshGenerationType
```

### Members

Members	Description
I_MGT_AUTOMATIC = 1	Automatic generation. Available since version 3.
I_MGT_USER = 2	Generation with user-defined parameters applied. Available since version 3.
I_MGT_ELEMENT_SIZE = 3	Available since version 4.5.

### Description

Methods of finite element mesh generation.

### Version

Available since version 3.

## II.6.1.10 IRobotMeshMethod

### Class Hierarchy

#### C++

```
interface IRobotMeshMethod : IDispatch;
```

#### C#

```
public interface IRobotMeshMethod;
```

## Visual Basic

```
Public Interface IRobotMeshMethod
```

### Description

A set of parameters defining a mesh generation method and associated quantities. .

### Version

Available since version 3.

## II.6.1.10.1 IRobotMeshMethod Members

The following tables list the members exposed by IRobotMeshMethod.

### Public Fields

	Name	Description
❖	ForcingRatio (❑ see page 810)	Forcing ratio for the selected method.
❖	ImplementDegree (❑ see page 810)	Degree of application for the selected method during mesh generation.
❖	Method (❑ see page 811)	Meshing method.

## II.6.1.10.2 IRobotMeshMethod Fields

The fields of the IRobotMeshMethod class are listed here.

### Public Fields

	Name	Description
❖	ForcingRatio (❑ see page 810)	Forcing ratio for the selected method.
❖	ImplementDegree (❑ see page 810)	Degree of application for the selected method during mesh generation.
❖	Method (❑ see page 811)	Meshing method.

### II.6.1.10.2.1 ForcingRatio

#### C++

```
HRESULT get_ForceRatio(IRobotMeshForcingRatio*);  
HRESULT put_ForceRatio(IRobotMeshForcingRatio);
```

#### C#

```
public IRobotMeshForcingRatio ForceRatio { get; set; }
```

#### Visual Basic

```
Public ForceRatio As IRobotMeshForcingRatio
```

#### Description

Forcing ratio for the selected method.

#### Version

Available since version 3.

### II.6.1.10.2.2 ImplementDegree

#### C++

```
HRESULT get_ImplementDegree(IRobotMeshImplementDegree*);  
HRESULT put_ImplementDegree(IRobotMeshImplementDegree);
```

#### C#

```
public IRobotMeshImplementDegree ImplementDegree { get; set; }
```

#### Visual Basic

```
Public ImplementDegree As IRobotMeshImplementDegree
```

#### Description

Degree of application for the selected method during mesh generation.

#### Version

Available since version 3.

### II.6.1.10.2.3 Method

#### C++

```
HRESULT get_Method(IRobotMeshMethodType*);  
HRESULT put_Method(IRobotMeshMethodType);
```

#### C#

```
public IRobotMeshMethodType Method { get; set; }
```

#### Visual Basic

```
Public Method As IRobotMeshMethodType
```

#### Description

Meshing method.

#### Version

Available since version 3.

### II.6.1.11 IRobotMeshGeneration

#### Class Hierarchy

**C++**

```
interface IRobotMeshGeneration : IDispatch;
```

**C#**

```
public interface IRobotMeshGeneration;
```

**Visual Basic**

```
Public Interface IRobotMeshGeneration
```

**Description**

Parameters of mesh generation.

**Version**

Available since version 3.

**II.6.1.11.1 IRobotMeshGeneration Members**

The following tables list the members exposed by IRobotMeshGeneration.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Division1 (see page 812)	Division parameter determining the number of elements to be generated on the first contour edge (between first and second contour vertex).
❖	Division2 (see page 812)	Division parameter determining the number of elements to be generated on the second contour edge (between second and third contour vertex) .
❖	ElementSize (see page 813)	Expected length of the finite element edge (in meters) .
❖	Type (see page 813)	Meshing type.

**II.6.1.11.2 IRobotMeshGeneration Fields**

The fields of the IRobotMeshGeneration class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Division1 (see page 812)	Division parameter determining the number of elements to be generated on the first contour edge (between first and second contour vertex).
❖	Division2 (see page 812)	Division parameter determining the number of elements to be generated on the second contour edge (between second and third contour vertex) .
❖	ElementSize (see page 813)	Expected length of the finite element edge (in meters) .
❖	Type (see page 813)	Meshing type.

**II.6.1.11.2.1 Division1****C++**

```
HRESULT get_Division1(double* );
HRESULT put_Division1(double);
```

**C#**

```
public double Division1 { get; set; }
```

**Visual Basic**

```
Public Division1 As Double
```

**Description**

Division parameter determining the number of elements to be generated on the first contour edge (between first and second contour vertex).

**Version**

Available since version 3.

**II.6.1.11.2.2 Division2****C++**

```
HRESULT get_Division2(double*);  
HRESULT put_Division2(double);
```

**C#**

```
public double Division2 { get; set; }
```

**Visual Basic**

```
Public Division2 As Double
```

**Description**

Division parameter determining the number of elements to be generated on the second contour edge (between second and third contour vertex) .

**Version**

Available since version 3.

**II.6.1.11.2.3 ElementSize****C++**

```
HRESULT get_ElementSize(double*);  
HRESULT put_ElementSize(double);
```

**C#**

```
public double ElementSize { get; set; }
```

**Visual Basic**

```
Public ElementSize As Double
```

**Description**

Expected length of the finite element edge (in meters) .

**Version**

Available since version 4.5.

**II.6.1.11.2.4 Type****C++**

```
HRESULT get_Type(IRobotMeshGenerationType*);  
HRESULT put_Type(IRobotMeshGenerationType);
```

**C#**

```
public IRobotMeshGenerationType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotMeshGenerationType
```

**Description**

Meshing type.

**Version**

Available since version 3.

## II.6.1.12 IRobotMeshCoonsParams

### Class Hierarchy

#### C++

```
interface IRobotMeshCoonsParams : IDispatch;
```

#### C#

```
public interface IRobotMeshCoonsParams;
```

### Visual Basic

```
Public Interface IRobotMeshCoonsParams
```

### Description

Parameters of Coons method.

### Version

Available since version 3.

## II.6.1.12.1 IRobotMeshCoonsParams Members

The following tables list the members exposed by IRobotMeshCoonsParams.

### Public Fields

	Name	Description
◆	ForcingRatio ( <a href="#">see page 814</a> )	Forcing ratio for the selected method of contour division.
◆	PanelDivisionType ( <a href="#">see page 814</a> )	Manner of contour division for Coons method.

## II.6.1.12.2 IRobotMeshCoonsParams Fields

The fields of the IRobotMeshCoonsParams class are listed here.

### Public Fields

	Name	Description
◆	ForcingRatio ( <a href="#">see page 814</a> )	Forcing ratio for the selected method of contour division.
◆	PanelDivisionType ( <a href="#">see page 814</a> )	Manner of contour division for Coons method.

## II.6.1.12.2.1 ForcingRatio

#### C++

```
HRESULT get_ForceRatio(IRobotMeshForcingRatio* );
HRESULT put_ForceRatio(IRobotMeshForcingRatio);
```

#### C#

```
public IRobotMeshForcingRatio ForceRatio { get; set; }
```

### Visual Basic

```
Public ForceRatio As IRobotMeshForcingRatio
```

### Description

Forcing ratio for the selected method of contour division.

### Version

Available since version 3.

## II.6.1.12.2.2 PanelDivisionType

### C++

```
HRESULT get_PanelDivisionType(IRobotMeshPanelDivType* );
HRESULT put_PanelDivisionType(IRobotMeshPanelDivType);
```

### C#

```
public IRobotMeshPanelDivType PanelDivisionType { get; set; }
```

### Visual Basic

```
Public PanelDivisionType As IRobotMeshPanelDivType
```

### Description

Manner of contour division for Coons method.

### Version

Available since version 3.

## II.6.1.13 IRobotMeshSurfaceFiniteElems

### Class Hierarchy

### C++

```
interface IRobotMeshSurfaceFiniteElems : IDispatch;
```

### C#

```
public interface IRobotMeshSurfaceFiniteElems;
```

### Visual Basic

```
Public Interface IRobotMeshSurfaceFiniteElems
```

### Description

Definition of surface finite elements applied during mesh generation. .

### Version

Available since version 3.

### II.6.1.13.1 IRobotMeshSurfaceFiniteElems Members

The following tables list the members exposed by IRobotMeshSurfaceFiniteElems.

#### Public Fields

	Name	Description
◆	ConversionCoeff (see page 815)	Triangle-to-quadrilateral conversion coefficient .
◆	ForcingRatio (see page 816)	Forcing ratio for the selected type of surface finite elements.
◆	Type (see page 816)	Finite element types applied during generation of surface element mesh .

### II.6.1.13.2 IRobotMeshSurfaceFiniteElems Fields

The fields of the IRobotMeshSurfaceFiniteElems class are listed here.

#### Public Fields

	Name	Description
◆	ConversionCoeff (see page 815)	Triangle-to-quadrilateral conversion coefficient .
◆	ForcingRatio (see page 816)	Forcing ratio for the selected type of surface finite elements.
◆	Type (see page 816)	Finite element types applied during generation of surface element mesh .

### II.6.1.13.2.1 ConversionCoeff

#### C++

```
HRESULT get_ConversionCoeff(double* );
HRESULT put_ConversionCoeff(double);
```

#### C#

```
public double ConversionCoeff { get; set; }
```

#### Visual Basic

```
Public ConversionCoeff As Double
```

#### Description

Triangle-to-quadrilateral conversion coefficient .

#### Version

Available since version 3.

### II.6.1.13.2.2 ForcingRatio

#### C++

```
HRESULT get_ForceRatio(IRobotMeshForcingRatio* );
HRESULT put_ForceRatio(IRobotMeshForcingRatio);
```

#### C#

```
public IRobotMeshForcingRatio ForceRatio { get; set; }
```

#### Visual Basic

```
Public ForceRatio As IRobotMeshForcingRatio
```

#### Description

Forcing ratio for the selected type of surface finite elements.

#### Version

Available since version 3.

### II.6.1.13.2.3 Type

#### C++

```
HRESULT get_Type(IRobotMeshSurfaceFEType* );
HRESULT put_Type(IRobotMeshSurfaceFEType);
```

#### C#

```
public IRobotMeshSurfaceFEType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotMeshSurfaceFEType
```

#### Description

Finite element types applied during generation of surface element mesh .

#### Version

Available since version 3.

### II.6.1.14 IRobotMeshDelaunayParams

#### Class Hierarchy

**C++**

```
interface IRobotMeshDelaunayParams : IDispatch;
```

**C#**

```
public interface IRobotMeshDelaunayParams;
```

**Visual Basic**

```
Public Interface IRobotMeshDelaunayParams
```

**Description**

Parameters of Delaunay's method.

**Version**

Available since version 3.

**II.6.1.14.1 IRobotMeshDelaunayParams Members**

The following tables list the members exposed by IRobotMeshDelaunayParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	EmittersDefault ( <a href="#">see page 817</a> )	Default emitter.
◆	EmittersSmoothing ( <a href="#">see page 818</a> )	Smoothing of a finite element mesh.
◆	EmittersUser ( <a href="#">see page 818</a> )	User-defined emitter.
◆	H_max ( <a href="#">see page 818</a> )	Length of the penultimate wave before the process of increasing mesh elements is completed .
◆	H0 ( <a href="#">see page 819</a> )	First wave length.
◆	MeshDensity ( <a href="#">see page 819</a> )	Mesh density expressed by a value from the interval from 0 (fine mesh) to 1 (coarse mesh).
◆	NumberOfLevels ( <a href="#">see page 819</a> )	
◆	Q ( <a href="#">see page 820</a> )	Relation of lengths of two successive waves.
◆	RegularMesh ( <a href="#">see page 820</a> )	
◆	Type ( <a href="#">see page 820</a> )	Exact type of mesh generation method according to Delaunay's method.

**II.6.1.14.2 IRobotMeshDelaunayParams Fields**

The fields of the IRobotMeshDelaunayParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	EmittersDefault ( <a href="#">see page 817</a> )	Default emitter.
◆	EmittersSmoothing ( <a href="#">see page 818</a> )	Smoothing of a finite element mesh.
◆	EmittersUser ( <a href="#">see page 818</a> )	User-defined emitter.
◆	H_max ( <a href="#">see page 818</a> )	Length of the penultimate wave before the process of increasing mesh elements is completed .
◆	H0 ( <a href="#">see page 819</a> )	First wave length.
◆	MeshDensity ( <a href="#">see page 819</a> )	Mesh density expressed by a value from the interval from 0 (fine mesh) to 1 (coarse mesh).
◆	NumberOfLevels ( <a href="#">see page 819</a> )	
◆	Q ( <a href="#">see page 820</a> )	Relation of lengths of two successive waves.
◆	RegularMesh ( <a href="#">see page 820</a> )	
◆	Type ( <a href="#">see page 820</a> )	Exact type of mesh generation method according to Delaunay's method.

### II.6.1.14.2.1 EmittersDefault

#### C++

```
HRESULT get_EmittersDefault(VARIANT_BOOL* );
HRESULT put_EmittersDefault(VARIANT_BOOL);
```

#### C#

```
public bool EmittersDefault { get; set; }
```

#### Visual Basic

```
Public EmittersDefault As Boolean
```

#### Description

Default emitter.

#### Version

Available since version 3.

### II.6.1.14.2.2 EmittersSmoothing

#### C++

```
HRESULT get_EmittersSmoothing(VARIANT_BOOL* );
HRESULT put_EmittersSmoothing(VARIANT_BOOL);
```

#### C#

```
public bool EmittersSmoothing { get; set; }
```

#### Visual Basic

```
Public EmittersSmoothing As Boolean
```

#### Description

Smoothing of a finite element mesh.

#### Version

Available since version 3.

### II.6.1.14.2.3 EmittersUser

#### C++

```
HRESULT get_EmittersUser(VARIANT_BOOL* );
HRESULT put_EmittersUser(VARIANT_BOOL);
```

#### C#

```
public bool EmittersUser { get; set; }
```

#### Visual Basic

```
Public EmittersUser As Boolean
```

#### Description

User-defined emitter.

#### Version

Available since version 3.

#### II.6.1.14.2.4 H\_max

##### C++

```
HRESULT get_H_max(double*);  
HRESULT put_H_max(double);
```

##### C#

```
public double H_max { get; set; }
```

##### Visual Basic

```
Public H_max As double
```

##### Description

Length of the penultimate wave before the process of increasing mesh elements is completed .

##### Version

Available since version 3.

#### II.6.1.14.2.5 H0

##### C++

```
HRESULT get_H0(double*);  
HRESULT put_H0(double);
```

##### C#

```
public double H0 { get; set; }
```

##### Visual Basic

```
Public H0 As double
```

##### Description

First wave length.

##### Version

Available since version 3.

#### II.6.1.14.2.6 MeshDensity

##### C++

```
HRESULT get_MeshDensity(double*);  
HRESULT put_MeshDensity(double);
```

##### C#

```
public double MeshDensity { get; set; }
```

##### Visual Basic

```
Public MeshDensity As double
```

##### Description

Mesh density expressed by a value from the interval from 0 (fine mesh) to 1 (coarse mesh).

##### Version

Available since version 13.

### II.6.1.14.2.7 NumberOfLevels

#### C++

```
HRESULT get_NumberOfLevels(long* );
HRESULT put_NumberOfLevels(long);
```

#### C#

```
public long NumberOfLevels { get; set; }
```

#### Visual Basic

```
Public NumberOfLevels As long
```

#### Version

Available since version 13.

### II.6.1.14.2.8 Q

#### C++

```
HRESULT get_Q(double* );
HRESULT put_Q(double);
```

#### C#

```
public double Q { get; set; }
```

#### Visual Basic

```
Public Q As double
```

#### Description

Relation of lengths of two successive waves.

#### Version

Available since version 3.

### II.6.1.14.2.9 RegularMesh

#### C++

```
HRESULT get-RegularMesh(VARIANT_BOOL* );
HRESULT put-RegularMesh(VARIANT_BOOL);
```

#### C#

```
public bool RegularMesh { get; set; }
```

#### Visual Basic

```
Public RegularMesh As Boolean
```

#### Version

Available since version 9.5.

### II.6.1.14.2.10 Type

#### C++

```
HRESULT get_Type(IRobotMeshDelaunayType* );
HRESULT put_Type(IRobotMeshDelaunayType);
```

#### C#

```
public IRobotMeshDelaunayType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotMeshDelaunayType
```

**Description**

Exact type of mesh generation method according to Delaunay's method.

**Version**

Available since version 3.

**II.6.1.15 IRobotMeshSurfaceParams****Class Hierarchy****C++**

```
interface IRobotMeshSurfaceParams : IDispatch;
```

**C#**

```
public interface IRobotMeshSurfaceParams;
```

**Visual Basic**

```
Public Interface IRobotMeshSurfaceParams
```

**Description**

Parameters of the mesh generation using planar finite elements.

**Version**

Available since version 3.

**II.6.1.15.1 IRobotMeshSurfaceParams Members**

The following tables list the members exposed by IRobotMeshSurfaceParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Coons ( <a href="#">see page 821</a> )	Parameters of Coons method.
◆	Delaunay ( <a href="#">see page 822</a> )	Parameters of Delaunay's method.
◆	FiniteElems ( <a href="#">see page 822</a> )	Definition of surface finite elements used during mesh generation.
◆	Generation ( <a href="#">see page 822</a> )	Parameters of mesh generation.
◆	Method ( <a href="#">see page 822</a> )	Definition of meshing method.

**II.6.1.15.2 IRobotMeshSurfaceParams Fields**

The fields of the IRobotMeshSurfaceParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Coons ( <a href="#">see page 821</a> )	Parameters of Coons method.
◆	Delaunay ( <a href="#">see page 822</a> )	Parameters of Delaunay's method.
◆	FiniteElems ( <a href="#">see page 822</a> )	Definition of surface finite elements used during mesh generation.
◆	Generation ( <a href="#">see page 822</a> )	Parameters of mesh generation.
◆	Method ( <a href="#">see page 822</a> )	Definition of meshing method.

**II.6.1.15.2.1 Coons****C++**

```
HRESULT get_Coons( IRobotMeshCoonsParams** );
```

**C#**

```
public IRobotMeshCoonsParams Coons { get; }
```

**Visual Basic**

```
Public ReadOnly Coons As IRobotMeshCoonsParams
```

**Description**

Parameters of Coons method.

**Version**

Available since version 3.

## II.6.1.15.2.2 Delaunay

**C++**

```
HRESULT get_Delaunay(IRobotMeshDelaunayParams**);
```

**C#**

```
public IRobotMeshDelaunayParams Delaunay { get; }
```

**Visual Basic**

```
Public ReadOnly Delaunay As IRobotMeshDelaunayParams
```

**Description**

Parameters of Delaunay's method.

**Version**

Available since version 3.

## II.6.1.15.2.3 FiniteElems

**C++**

```
HRESULT get_FiniteElems(IRobotMeshSurfaceFiniteElems**);
```

**C#**

```
public IRobotMeshSurfaceFiniteElems FiniteElems { get; }
```

**Visual Basic**

```
Public ReadOnly FiniteElems As IRobotMeshSurfaceFiniteElems
```

**Description**

Definition of surface finite elements used during mesh generation.

**Version**

Available since version 3.

## II.6.1.15.2.4 Generation

**C++**

```
HRESULT get_Generation(IRobotMeshGeneration**);
```

**C#**

```
public IRobotMeshGeneration Generation { get; }
```

**Visual Basic**

```
Public ReadOnly Generation As IRobotMeshGeneration
```

**Description**

Parameters of mesh generation.

**Version**

Available since version 3.

**II.6.1.15.2.5 Method****C++**

```
HRESULT get_Method( IRobotMeshMethod** );
```

**C#**

```
public IRobotMeshMethod Method { get; }
```

**Visual Basic**

```
Public ReadOnly Method As IRobotMeshMethod
```

**Description**

Definition of meshing method.

**Version**

Available since version 3.

**II.6.1.16 IRobotMeshVolumeParams****Class Hierarchy****C++**

```
interface IRobotMeshVolumeParams : IDispatch;
```

**C#**

```
public interface IRobotMeshVolumeParams;
```

**Visual Basic**

```
Public Interface IRobotMeshVolumeParams
```

**Description**

Parameters of volumetric element mesh generation. .

**Version**

Available since version 3.

**II.6.1.16.1 IRobotMeshVolumeParams Members**

The following tables list the members exposed by IRobotMeshVolumeParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AdditionalSurfaceMeshing ( <a href="#">see page 824</a> )	Flag switching on additional mesh generation on a solid surface while generating a mesh of volumetric finite elements; if this option is switched on, it enables additional mesh generation on a solid (surface) contour which will affect the size of mesh elements within a solid .
◆	FiniteElemsType ( <a href="#">see page 824</a> )	Finite element type applied during generation of a volumetric element mesh.
◆	MeshDensity ( <a href="#">see page 824</a> )	Refinement degree for volumetric finite element mesh .

## II.6.1.16.2 IRobotMeshVolumeParams Fields

The fields of the IRobotMeshVolumeParams class are listed here.

### Public Fields

	Name	Description
◆	AdditionalSurfaceMeshing (see page 824)	Flag switching on additional mesh generation on a solid surface while generating a mesh of volumetric finite elements; if this option is switched on, it enables additional mesh generation on a solid (surface) contour which will affect the size of mesh elements within a solid .
◆	FiniteElemsType (see page 824)	Finite element type applied during generation of a volumetric element mesh.
◆	MeshDensity (see page 824)	Refinement degree for volumetric finite element mesh .

### II.6.1.16.2.1 AdditionalSurfaceMeshing

#### C++

```
HRESULT get_AdditionalSurfaceMeshing(VARIANT_BOOL* );
HRESULT put_AdditionalSurfaceMeshing(VARIANT_BOOL);
```

#### C#

```
public bool AdditionalSurfaceMeshing { get; set; }
```

#### Visual Basic

```
Public AdditionalSurfaceMeshing As Boolean
```

#### Description

Flag switching on additional mesh generation on a solid surface while generating a mesh of volumetric finite elements; if this option is switched on, it enables additional mesh generation on a solid (surface) contour which will affect the size of mesh elements within a solid .

#### Version

Available since version 3.

### II.6.1.16.2.2 FiniteElemsType

#### C++

```
HRESULT get_FiniteElemsType(IRobotMeshSurfaceFEType* );
HRESULT put_FiniteElemsType(IRobotMeshSurfaceFEType);
```

#### C#

```
public IRobotMeshSurfaceFEType FiniteElemsType { get; set; }
```

#### Visual Basic

```
Public FiniteElemsType As IRobotMeshSurfaceFEType
```

#### Description

Finite element type applied during generation of a volumetric element mesh.

#### Version

Available since version 3.

### II.6.1.16.2.3 MeshDensity

#### C++

```
HRESULT get_MeshDensity(double* );
HRESULT put_MeshDensity(double);
```

**C#**

```
public double MeshDensity { get; set; }
```

**Visual Basic**

```
Public MeshDensity As Double
```

**Description**

Refinement degree for volumetric finite element mesh .

**Version**

Available since version 3.

**II.6.1.17 IRobotMeshParams****Class Hierarchy****C++**

```
interface IRobotMeshParams : IDispatch;
```

**C#**

```
public interface IRobotMeshParams;
```

**Visual Basic**

```
Public Interface IRobotMeshParams
```

**Description**

Parameters of the finite element mesh generation.

**Version**

Available since version 3.

**II.6.1.17.1 IRobotMeshParams Members**

The following tables list the members exposed by IRobotMeshParams.

**Public Fields**

	Name	Description
◆	Flag ( [ see page 825) )	Flag of access to parameters of surface and volumetric mesh model.
◆	MeshType ( [ see page 826) )	Types of generated mesh for surface or volumetric finite elements.
◆	SurfaceParams ( [ see page 826) )	Set of attributes for generation of a surface element mesh.
◆	VolumeParams ( [ see page 826) )	Set of attributes for generation of a volumetric element mesh.

**II.6.1.17.2 IRobotMeshParams Fields**

The fields of the IRobotMeshParams class are listed here.

**Public Fields**

	Name	Description
◆	Flag ( [ see page 825) )	Flag of access to parameters of surface and volumetric mesh model.
◆	MeshType ( [ see page 826) )	Types of generated mesh for surface or volumetric finite elements.
◆	SurfaceParams ( [ see page 826) )	Set of attributes for generation of a surface element mesh.
◆	VolumeParams ( [ see page 826) )	Set of attributes for generation of a volumetric element mesh.

### II.6.1.17.2.1 Flag

#### C++

```
HRESULT get_Flag(IRobotMeshAccessType*);
```

#### C#

```
public IRobotMeshAccessType Flag { get; }
```

#### Visual Basic

```
Public ReadOnly Flag As IRobotMeshAccessType
```

#### Description

Flag of access to parameters of surface and volumetric mesh model.

#### Version

Available since version 3.

### II.6.1.17.2.2 MeshType

#### C++

```
HRESULT get_MeshType(IRobotMeshType*);  
HRESULT put_MeshType(IRobotMeshType);
```

#### C#

```
public IRobotMeshType MeshType { get; set; }
```

#### Visual Basic

```
Public MeshType As IRobotMeshType
```

#### Description

Types of generated mesh for surface or volumetric finite elements.

#### Version

Available since version 3.

### II.6.1.17.2.3 SurfaceParams

#### C++

```
HRESULT get_SurfaceParams(IRobotMeshSurfaceParams**);
```

#### C#

```
public IRobotMeshSurfaceParams SurfaceParams { get; }
```

#### Visual Basic

```
Public ReadOnly SurfaceParams As IRobotMeshSurfaceParams
```

#### Description

Set of attributes for generation of a surface element mesh.

#### Version

Available since version 3.

### II.6.1.17.2.4 VolumeParams

#### C++

```
HRESULT get_VolumeParams(IRobotMeshVolumeParams**);
```

**C#**

```
public IRobotMeshVolumeParams VolumeParams { get; }
```

**Visual Basic**

```
Public ReadOnly VolumeParams As IRobotMeshVolumeParams
```

**Description**

Set of attributes for generation of a volumetric element mesh.

**Version**

Available since version 3.

**II.6.1.18 IRobotMeshAccessType****C++**

```
enum IRobotMeshAccessType;
```

**C#**

```
public enum IRobotMeshAccessType;
```

**Visual Basic**

```
Public Enum IRobotMeshAccessType
```

**Members**

Members	Description
I_MAT_SURFACE_AND_VOLUMETRIC = 1	Flag indicating availability of the interface of surface mesh parameters and the interface of volumetric mesh parameters. Available since version 3.
I_MAT_SURFACE_ONLY = 2	Flag indicating availability only of the interface of surface mesh parameters. Available since version 3.
I_MAT_VOLUMETRIC_ONLY = 3	Flag indicating availability only of the interface of volumetric mesh parameters. Available since version 3.

**Description**

Availability of interfaces for mesh generation.

**Version**

Available since version 3.

**II.6.1.19 IRobotMeshRefinementType****C++**

```
enum IRobotMeshRefinementType;
```

**C#**

```
public enum IRobotMeshRefinementType;
```

**Visual Basic**

```
Public Enum IRobotMeshRefinementType
```

## Members

Members	Description
I_MRT_DOUBLE = 1	Each finite element edge is divided into two parts. Available since version 3.
I_MRT_TRIPLE = 2	Each finite element edge is divided into three parts. Available since version 3.
I_MRT_SIMPLE = 3	Finite element edges will not be divided. Available since version 3.

## Description

Methods of finite element mesh refinement.

## Version

Available since version 3.

## II.6.2 IRobotFiniteElement

### Class Hierarchy

#### C++

```
interface IRobotFiniteElement : IRobotDataObject;
```

#### C#

```
public interface IRobotFiniteElement : IRobotDataObject;
```

### Visual Basic

```
Public Interface IRobotFiniteElement
```

## Description

Interface describing planar finite element. .

## Version

Available since version 2.5.

## II.6.2.1 IRobotFiniteElement Members

The following tables list the members exposed by IRobotFiniteElement.

### Public Fields

	Name	Description
◆	FeType ( <a href="#">see page 829</a> )	Finite element type.
◆	Nodes ( <a href="#">see page 829</a> )	Nodes defining a finite element.
◆	Number ( <a href="#">see page 30</a> )	Object number is assigned by the user; it is unique among all the components of the same type.. .
◆	ObjectNumber ( <a href="#">see page 829</a> )	Number ( <a href="#">see page 30</a> ) of the object including the finite element.
◆	ObjectPartIdx ( <a href="#">see page 830</a> )	Index of the object component including the element.
◆	Uniqueld ( <a href="#">see page 830</a> )	Unique element identifier .

### Public Methods

	Name	Description
◆	CalcArea ( <a href="#">see page 830</a> )	Function calculates a finite element area.
◆	GetLabel ( <a href="#">see page 31</a> )	The function returns a label of the indicated type applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .

	GetLabelName (see page 31)	The function returns the name of a label of the indicated type, applied to the object. If the object does not have any label of the type, running the function leads to a critical error. .
	GetLabels (see page 31)	The function returns a collection containing all labels (of different types) defined for a given object. .
	HasLabel (see page 32)	The function returns True (non-zero value) if a label of the type has already been defined for the object. .
	RemoveLabel (see page 32)	The function deletes a label of the indicated type. .
	SetLabel (see page 32)	The function applies the defined label to an object. At a given moment, an object may have only one label of a type applied (e.g. there may not be two supports defined for one node). If an object has a formerly applied label of the same type, it is replaced by the new one. .

## II.6.2.2 IRobotFiniteElement Fields

The fields of the IRobotFiniteElement class are listed here.

### Public Fields

	Name	Description
	FeType (see page 829)	Finite element type.
	Nodes (see page 829)	Nodes defining a finite element.
	ObjectNumber (see page 829)	Number (see page 30) of the object including the finite element.
	ObjectPartIdx (see page 830)	Index of the object component including the element.
	Uniqueld (see page 830)	Unique element identifier .

### II.6.2.2.1 FeType

#### C++

```
HRESULT get_FeType( IRobotFiniteElementType* );
HRESULT put_FeType( IRobotFiniteElementType );
```

#### C#

```
public IRobotFiniteElementType FeType { get; set; }
```

#### Visual Basic

```
Public FeType As IRobotFiniteElementType
```

#### Description

Finite element type.

#### Version

Available since version 2.5.

### II.6.2.2.2 Nodes

#### C++

```
HRESULT get_Nodes( IRobotFiniteElementNodes** );
```

#### C#

```
public IRobotFiniteElementNodes Nodes { get; }
```

#### Visual Basic

```
Public ReadOnly Nodes As IRobotFiniteElementNodes
```

#### Description

Nodes defining a finite element.

**Version**

Available since version 2.5.

**II.6.2.2.3 ObjectNumber****C++**

```
HRESULT get_ObjectNumber(long*);
```

**C#**

```
public long ObjectNumber { get; }
```

**Visual Basic**

```
Public ReadOnly ObjectNumber As long
```

**Description**

Number (see page 30) of the object including the finite element.

**Version**

Available since version 2.5.

**II.6.2.2.4 ObjectPartIdx****C++**

```
HRESULT get_ObjectPartIdx(long*);
```

**C#**

```
public long ObjectPartIdx { get; }
```

**Visual Basic**

```
Public ReadOnly ObjectPartIdx As long
```

**Description**

Index of the object component including the element.

**Version**

Available since version 2.5.

**II.6.2.2.5 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique element identifier .

**Version**

Available since version 2.5.

**II.6.2.3 IRobotFiniteElement Methods**

The methods of the IRobotFiniteElement class are listed here.

## Public Methods

	Name	Description
💡	CalcArea (🔗 see page 830)	Function calculates a finite element area.

### II.6.2.3.1 CalcArea

#### C++

```
HRESULT CalcArea(double* ret);
```

#### C#

```
public double CalcArea();
```

#### Visual Basic

```
Public Function CalcArea() As double
```

#### Description

Function calculates a finite element area.

#### Version

Available since version 7.5.

## II.6.3 IRobotFiniteElementType

#### C++

```
enum IRobotFiniteElementType;
```

#### C#

```
public enum IRobotFiniteElementType;
```

#### Visual Basic

```
Public Enum IRobotFiniteElementType
```

#### Members

Members	Description
I_FET_T3 = 3	Three-node element. Available since version 2.5.
I_FET_Q4 = 4	Four-node element. Available since version 2.5.
I_FET_T6 = 1	Six-node element. Available since version 2.5.
I_FET_Q8 = 2	Eight-node element. Available since version 2.5.
I_FET_VOL_B8 = 5	Available since version 7.5.
I_FET_VOL_T4 = 6	Available since version 7.5.
I_FET_VOL_W6 = 7	Available since version 7.5.
I_FET_VOL_B20 = 8	Available since version 7.5.
I_FET_VOL_T10 = 9	Available since version 7.5.
I_FET_VOL_W15 = 10	Available since version 7.5.

#### Description

Available types of finite elements.

**Version**

Available since version 2.5.

**II.6.4 IRobotFiniteElementNodes****Class Hierarchy****C++**

```
interface IRobotFiniteElementNodes : IDispatch;
```

**C#**

```
public interface IRobotFiniteElementNodes;
```

**Visual Basic**

```
Public Interface IRobotFiniteElementNodes
```

**Description**

Collection of nodes defining finite element.

**Version**

Available since version 2.5.

**II.6.4.1 IRobotFiniteElementNodes Members**

The following tables list the members exposed by IRobotFiniteElementNodes.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 832</a> )	Number of nodes in the collection (simultaneously, it is the index of the last node).

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Get ( <a href="#">see page 833</a> )	Function returns number of the node with the specified index in the collection.
◆	GetAll ( <a href="#">see page 833</a> )	Function returns the table with numbers of nodes defining the finite element.
◆	Set ( <a href="#">see page 833</a> )	Function inserts the specified node onto the indicated index in the collection of nodes defining finite element. .
◆	SetAll ( <a href="#">see page 834</a> )	Function defines all nodes of the finite element.

**II.6.4.2 IRobotFiniteElementNodes Fields**

The fields of the IRobotFiniteElementNodes class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 832</a> )	Number of nodes in the collection (simultaneously, it is the index of the last node).

**II.6.4.2.1 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of nodes in the collection (simultaneously, it is the index of the last node).

**Version**

Available since version 2.5.

**II.6.4.3 IRobotFiniteElementNodes Methods**

The methods of the IRobotFiniteElementNodes class are listed here.

**Public Methods**

	Name	Description
≡	Get ( <a href="#">see page 833</a> )	Function returns number of the node with the specified index in the collection.
≡	GetAll ( <a href="#">see page 833</a> )	Function returns the table with numbers of nodes defining the finite element. .
≡	Set ( <a href="#">see page 833</a> )	Function inserts the specified node onto the indicated index in the collection of nodes defining finite element. .
≡	SetAll ( <a href="#">see page 834</a> )	Function defines all nodes of the finite element.

**II.6.4.3.1 Get****C++**

```
HRESULT Get(long _node_idx, long* ret);
```

**C#**

```
public long Get(long _node_idx);
```

**Visual Basic**

```
Public Function Get(_node_idx As Long) As Long
```

**Description**

Function returns number of the node with the specified index in the collection.

**Version**

Available since version 2.5.

**II.6.4.3.2 GetAll****C++**

```
HRESULT GetAll(IRobotNumbersArray** ret);
```

**C#**

```
public IRobotNumbersArray GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotNumbersArray
```

**Description**

Function returns the table with numbers of nodes defining the finite element. .

**Version**

Available since version 2.5.

**II.6.4.3.3 Set****C++**

```
HRESULT Set(long _node_idx, long _node_num, VARIANT_BOOL* ret);
```

**C#**

```
public bool Set(long _node_idx, long _node_num);
```

**Visual Basic**

```
Public Function Set(_node_idx As long, _node_num As long) As Boolean
```

**Description**

Function inserts the specified node onto the indicated index in the collection of nodes defining finite element. .

**Version**

Available since version 2.5.

**II.6.4.3.4 SetAll****C++**

```
HRESULT SetAll(IRobotNumbersArray* _nodes_array);
```

**C#**

```
public void SetAll(IRobotNumbersArray _nodes_array);
```

**Visual Basic**

```
Public Sub SetAll(ByRef _nodes_array As IRobotNumbersArray)
```

**Description**

Function defines all nodes of the finite element.

**Version**

Available since version 2.5.

**II.6.5 IRobotFiniteElementServer****Class Hierarchy****C++**

```
interface IRobotFiniteElementServer : IRobotDataObjectServer;
```

**C#**

```
public interface IRobotFiniteElementServer : IRobotDataObjectServer;
```

**Visual Basic**

```
Public Interface IRobotFiniteElementServer
```

**Description**

Server of finite elements. .

**Version**

Available since version 2.5.

## II.6.5.1 IRobotFiniteElementServer Members

The following tables list the members exposed by IRobotFiniteElementServer.

### Public Fields

	Name	Description
◆	FreeNumber ( <a href="#">see page 835</a> )	First available user-defined number that can be assigned to a finite element.

### Public Methods

	Name	Description
◆	CalcArea ( <a href="#">see page 836</a> )	Function calculates an area of the finite element of the given number.
◆	Create ( <a href="#">see page 836</a> )	Function creates a new finite element defined on the indicated nodes. . .
◆	Delete ( <a href="#">see page 34</a> )	The function deletes the object of the indicated number. . .
◆	DeleteMany ( <a href="#">see page 34</a> )	The function deletes all objects that meet the criteria of the indicated selection. . .
◆	Exist ( <a href="#">see page 34</a> )	The function returns True (non-zero value) if the object of the indicated name already exists. . .
◆	Get ( <a href="#">see page 34</a> )	The function returns an object with the indicated user-defined number. The type of the returned object agrees with the type of objects managed by the server. If the object of the indicated number does not exist, running the function leads to critical error. . .
◆	GetAll ( <a href="#">see page 35</a> )	The function returns a collection containing all objects managed by the server. . .
◆	GetMany ( <a href="#">see page 35</a> )	The function returns a collection of objects that meet the criteria of the indicated selection. . .
◆	MeshConcentrate ( <a href="#">see page 836</a> )	Function carries out mesh refinement for the indicated selection of finite elements. . .
◆	MeshConsolidate ( <a href="#">see page 837</a> )	Function performs mesh consolidation for the indicated finite element list.
◆	RemoveLabel ( <a href="#">see page 35</a> )	The function removes the labels of the indicated type from all objects that meet the criteria of the indicated selection. . .
◆	SetLabel ( <a href="#">see page 36</a> )	The function applies a label (identified by the type and name) to all objects that meet the criteria of the indicated selection. . .
◆	Update ( <a href="#">see page 837</a> )	Function reconstructs relations between finite elements and structure nodes. It should be activated, if the Create ( <a href="#">see page 836</a> ) function with delayed model update, i.e., with parameter_update_model set as False, has been used earlier. . .

## II.6.5.2 IRobotFiniteElementServer Fields

The fields of the IRobotFiniteElementServer class are listed here.

### Public Fields

	Name	Description
◆	FreeNumber ( <a href="#">see page 835</a> )	First available user-defined number that can be assigned to a finite element.

## II.6.5.2.1 FreeNumber

### C++

```
HRESULT get_FreeNumber( long* );
```

### C#

```
public long FreeNumber { get; }
```

### Visual Basic

```
Public ReadOnly FreeNumber As long
```

## Description

First available user-defined number that can be assigned to a finite element.

## Version

Available since version 2.5.

### II.6.5.3 IRobotFiniteElementServer Methods

The methods of the IRobotFiniteElementServer class are listed here.

#### Public Methods

	Name	Description
💡	CalcArea (see page 836)	Function calculates an area of the finite element of the given number.
💡	Create (see page 836)	Function creates a new finite element defined on the indicated nodes. .
💡	MeshConcentrate (see page 836)	Function carries out mesh refinement for the indicated selection of finite elements. .
💡	MeshConsolidate (see page 837)	Function performs mesh consolidation for the indicated finite element list.
💡	Update (see page 837)	Function reconstructs relations between finite elements and structure nodes. It should be activated, if the Create (see page 836) function with delayed model update, i.e., with parameter_update_model set as False, has been used earlier. .

#### II.6.5.3.1 CalcArea

##### C++

```
HRESULT CalcArea(long _ele_num, double* ret);
```

##### C#

```
public double CalcArea(long _ele_num);
```

##### Visual Basic

```
Public Function CalcArea(_ele_num As long) As double
```

## Description

Function calculates an area of the finite element of the given number.

## Version

Available since version 7.5.

#### II.6.5.3.2 Create

##### C++

```
HRESULT Create(long _number, IRobotNumbersArray* _nodes, VARIANT_BOOL _update_model = True);
```

##### C#

```
public void Create(long _number, IRobotNumbersArray _nodes, bool _update_model = True);
```

##### Visual Basic

```
Public Sub Create(_number As long, ByRef _nodes As IRobotNumbersArray, Optional
_update_model As Boolean = True)
```

## Description

Function creates a new finite element defined on the indicated nodes. .

**Version**

Available since version 3.

**II.6.5.3.3 MeshConcentrate****C++**

```
HRESULT MeshConcentrate(IRobotMeshRefinementType _type, IRobotSelection* _sel, VARIANT_BOOL _auto_freeze = true);
```

**C#**

```
public void MeshConcentrate(IRobotMeshRefinementType _type, IRobotSelection _sel, bool _auto_freeze = true);
```

**Visual Basic**

```
Public Sub MeshConcentrate(_type As IRobotMeshRefinementType, ByRef _sel As IRobotSelection, Optional _auto_freeze As Boolean = true)
```

**Description**

Function carries out mesh refinement for the indicated selection of finite elements. .

**Version**

Available since version 3.

**II.6.5.3.4 MeshConsolidate****C++**

```
HRESULT MeshConsolidate(double _coeff, IRobotSelection* _sel, VARIANT_BOOL _auto_freeze = true);
```

**C#**

```
public void MeshConsolidate(double _coeff, IRobotSelection _sel, bool _auto_freeze = true);
```

**Visual Basic**

```
Public Sub MeshConsolidate(_coeff As double, ByRef _sel As IRobotSelection, Optional _auto_freeze As Boolean = true)
```

**Description**

Function performs mesh consolidation for the indicated finite element list.

**Version**

Available since version 3.

**II.6.5.3.5 Update****C++**

```
HRESULT Update();
```

**C#**

```
public void Update();
```

**Visual Basic**

```
Public Sub Update()
```

**Description**

Function reconstructs relations between finite elements and structure nodes. It should be activated, if the Create ( see

page 836) function with delayed model update, i.e., with parameter\_update\_model set as False, has been used earlier. .

## Version

Available since version 3.

## II.6.6 IRobotFiniteElementData

### Class Hierarchy

#### C++

```
interface IRobotFiniteElementData : IDispatch;
```

#### C#

```
public interface IRobotFiniteElementData;
```

### Visual Basic

```
Public Interface IRobotFiniteElementData
```

### Description

Interface gives access to finite element data.

## Version

Available since version 9.

## II.6.6.1 IRobotFiniteElementData Members

The following tables list the members exposed by IRobotFiniteElementData.

### Public Fields

	Name	Description
❖	NodeCount (see page 838)	Number (see page 839) of nodes defining the element.
❖	Number (see page 839)	Finite element number assigned by the user.
❖	Object (see page 839)	Number (see page 839) of an object that the finite element belongs to.

### Public Methods

	Name	Description
❖	GetNode (see page 839)	Function returns a number of the node with the specified index and the node coordinates.

## II.6.6.2 IRobotFiniteElementData Fields

The fields of the IRobotFiniteElementData class are listed here.

### Public Fields

	Name	Description
❖	NodeCount (see page 838)	Number (see page 839) of nodes defining the element.
❖	Number (see page 839)	Finite element number assigned by the user.
❖	Object (see page 839)	Number (see page 839) of an object that the finite element belongs to.

## II.6.6.2.1 NodeCount

#### C++

```
HRESULT get_NodeCount(long*);
```

#### C#

```
public long NodeCount { get; }
```

**Visual Basic**

```
Public ReadOnly NodeCount As long
```

**Description**

Number (see page 839) of nodes defining the element.

**Version**

Available since version 9.

**II.6.6.2.2 Number****C++**

```
HRESULT get_Number(long*);
```

**C#**

```
public long Number { get; }
```

**Visual Basic**

```
Public ReadOnly Number As long
```

**Description**

Finite element number assigned by the user.

**Version**

Available since version 9.

**II.6.6.2.3 Object****C++**

```
HRESULT get_Object(long*);
```

**C#**

```
public long Object { get; }
```

**Visual Basic**

```
Public ReadOnly Object As long
```

**Description**

Number (see page 839) of an object that the finite element belongs to.

**Version**

Available since version 9.

**II.6.6.3 IRobotFiniteElementData Methods**

The methods of the IRobotFiniteElementData class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetNode (see page 839)	Function returns a number of the node with the specified index and the node coordinates.

**II.6.6.3.1 GetNode****C++**

```
HRESULT GetNode(long _node_idx, double* _ret_x, double* _ret_y, double* _ret_z, long* ret);
```

**C#**

```
public long GetNode(long _node_idx, double* _ret_x, double* _ret_y, double* _ret_z);
```

**Visual Basic**

```
Public Function GetNode(_node_idx As long, ByRef _ret_x As double*, ByRef _ret_y As
double*, ByRef _ret_z As double*) As long
```

**Description**

Function returns a number of the node with the specified index and the node coordinates.

**Version**

Available since version 9.

**II.6.7 IRobotFiniteElementDataSet****Class Hierarchy****C++**

```
interface IRobotFiniteElementDataSet : IDispatch;
```

**C#**

```
public interface IRobotFiniteElementDataSet;
```

**Visual Basic**

```
Public Interface IRobotFiniteElementDataSet
```

**Description**

Interface gives access to the finite element set.

**Version**

Available since version 9.

**II.6.7.1 IRobotFiniteElementDataSet Members**

The following tables list the members exposed by IRobotFiniteElementDataSet.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	ElemCount ( <a href="#">see page 841</a> )	Number of elements belonging to the set.
◆	ObjectCount ( <a href="#">see page 841</a> )	Number of objects whose elements are included in the set.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	GetElem ( <a href="#">see page 842</a> )	Function returns a number of the element with the specified index and gives access to its data.
◆	GetElemCountForObject ( <a href="#">see page 842</a> )	Function returns a number of elements for the object with the specified index.
◆	GetElemForObject ( <a href="#">see page 842</a> )	Function returns data for the determined element of a given object.
◆	GetObjectNumber ( <a href="#">see page 843</a> )	Function returns a number of the object with the specified index.

**II.6.7.2 IRobotFiniteElementDataSet Fields**

The fields of the IRobotFiniteElementDataSet class are listed here.

## Public Fields

	Name	Description
◆	ElemCount ( <a href="#">see page 841</a> )	Number of elements belonging to the set.
◆	ObjectCount ( <a href="#">see page 841</a> )	Number of objects whose elements are included in the set.

### II.6.7.2.1 ElemCount

#### C++

```
HRESULT get_ElemCount(long*);
```

#### C#

```
public long ElemCount { get; }
```

#### Visual Basic

```
Public ReadOnly ElemCount As long
```

#### Description

Number of elements belonging to the set.

#### Version

Available since version 9.

### II.6.7.2.2 ObjectCount

#### C++

```
HRESULT get_ObjectCount(long*);
```

#### C#

```
public long ObjectCount { get; }
```

#### Visual Basic

```
Public ReadOnly ObjectCount As long
```

#### Description

Number of objects whose elements are included in the set.

#### Version

Available since version 9.

### II.6.7.3 IRobotFiniteElementDataSet Methods

The methods of the IRobotFiniteElementDataSet class are listed here.

#### Public Methods

	Name	Description
◆	GetElem ( <a href="#">see page 842</a> )	Function returns a number of the element with the specified index and gives access to its data.
◆	GetElemCountForObject ( <a href="#">see page 842</a> )	Function returns a number of elements for the object with the specified index.
◆	GetElemForObject ( <a href="#">see page 842</a> )	Function returns data for the determined element of a given object.
◆	GetObjectNumber ( <a href="#">see page 843</a> )	Function returns a number of the object with the specified index.

### II.6.7.3.1 GetElem

#### C++

```
HRESULT GetElem(long _elem_idx, IRobotFiniteElementData* _elem_data, long* ret);
```

#### C#

```
public long GetElem(long _elem_idx, IRobotFiniteElementData _elem_data);
```

#### Visual Basic

```
Public Function GetElem(_elem_idx As long, ByRef _elem_data As IRobotFiniteElementData) As long
```

#### Description

Function returns a number of the element with the specified index and gives access to its data.

#### Version

Available since version 9.

### II.6.7.3.2 GetElemCountForObject

#### C++

```
HRESULT GetElemCountForObject(long _obj_idx, long* ret);
```

#### C#

```
public long GetElemCountForObject(long _obj_idx);
```

#### Visual Basic

```
Public Function GetElemCountForObject(_obj_idx As long) As long
```

#### Description

Function returns a number of elements for the object with the specified index.

#### Version

Available since version 9.

### II.6.7.3.3 GetElemForObject

#### C++

```
HRESULT GetElemForObject(long _obj_idx, long _elem_idx, IRobotFiniteElementData* _ret_data, long* ret);
```

#### C#

```
public long GetElemForObject(long _obj_idx, long _elem_idx, IRobotFiniteElementData _ret_data);
```

#### Visual Basic

```
Public Function GetElemForObject(_obj_idx As long, _elem_idx As long, ByRef _ret_data As IRobotFiniteElementData) As long
```

#### Description

Function returns data for the determined element of a given object.

#### Version

Available since version 9.

### II.6.7.3.4 GetObjectNumber

**C++**

```
HRESULT GetObjectNumber(long _obj_idx, long* ret);
```

**C#**

```
public long GetObjectNumber(long _obj_idx);
```

**Visual Basic**

```
Public Function GetObjectNumber(_obj_idx As long) As long
```

**Description**

Function returns a number of the object with the specified index.

**Version**

Available since version 9.

## II.7 Grouping of objects

Available since version 7.5.

**Interfaces**

	Name	Description
	IRobotGroupObjectServer ( <a href="#">see page 843</a> )	Servers allows operations of grouping / ungrouping objects.

### II.7.1 IRobotGroupObjectServer

**Class Hierarchy**

**C++**

```
interface IRobotGroupObjectServer : IDispatch;
```

**C#**

```
public interface IRobotGroupObjectServer;
```

**Visual Basic**

```
Public Interface IRobotGroupObjectServer
```

**Description**

Servers allows operations of grouping / ungrouping objects.

**Version**

Available since version 7.5.

#### II.7.1.1 IRobotGroupObjectServer Members

The following tables list the members exposed by IRobotGroupObjectServer.

**Public Methods**

	Name	Description
	Explode ( <a href="#">see page 844</a> )	Function ungroups a group of structure elements of the specified number.
	FindFirst ( <a href="#">see page 844</a> )	Function returns the number of the first object grouping structure elements. If no objects of this type have been defined, function returns zero value.

	FindNext ( <a href="#">see page 845</a> )	Function returns the number of the next object grouping structure elements. If no more grouping objects have been defined, function returns zero value.
	GetContents ( <a href="#">see page 845</a> )	Function returns selection lists of structure elements included in the specified "grouped" object.
	GroupGiven ( <a href="#">see page 845</a> )	Function groups (combines into one object) structure elements described by selection lists. Function returns the number of a created object.
	GroupSelected ( <a href="#">see page 846</a> )	Function groups (combines into one object) structure elements belonging to the current selection. Function returns the number of a created object.

## II.7.1.2 IRobotGroupObjectServer Methods

The methods of the IRobotGroupObjectServer class are listed here.

### Public Methods

	Name	Description
	Explode ( <a href="#">see page 844</a> )	Function ungroups a group of structure elements of the specified number.
	FindFirst ( <a href="#">see page 844</a> )	Function returns the number of the first object grouping structure elements. If no objects of this type have been defined, function returns zero value.
	FindNext ( <a href="#">see page 845</a> )	Function returns the number of the next object grouping structure elements. If no more grouping objects have been defined, function returns zero value.
	GetContents ( <a href="#">see page 845</a> )	Function returns selection lists of structure elements included in the specified "grouped" object.
	GroupGiven ( <a href="#">see page 845</a> )	Function groups (combines into one object) structure elements described by selection lists. Function returns the number of a created object.
	GroupSelected ( <a href="#">see page 846</a> )	Function groups (combines into one object) structure elements belonging to the current selection. Function returns the number of a created object.

### II.7.1.2.1 Explode

#### C++

```
HRESULT Explode(long _group_obj);
```

#### C#

```
public void Explode(long _group_obj);
```

#### Visual Basic

```
Public Sub Explode(_group_obj As long)
```

#### Description

Function ungroups a group of structure elements of the specified number.

#### Version

Available since version 7.5.

### II.7.1.2.2 FindFirst

#### C++

```
HRESULT FindFirst(long* ret);
```

#### C#

```
public long FindFirst();
```

**Visual Basic**

```
Public Function FindFirst() As long
```

**Description**

Function returns the number of the first object grouping structure elements. If no objects of this type have been defined, function returns zero value.

**Version**

Available since version 7.5.

**II.7.1.2.3 FindNext****C++**

```
HRESULT FindNext(long _prev_group_obj, long* ret);
```

**C#**

```
public long FindNext(long _prev_group_obj);
```

**Visual Basic**

```
Public Function FindNext(_prev_group_obj As long) As long
```

**Description**

Function returns the number of the next object grouping structure elements. If no more grouping objects have been defined, function returns zero value.

**Version**

Available since version 7.5.

**II.7.1.2.4 GetContents****C++**

```
HRESULT GetContents(long _group_obj, BSTR _object_sel, BSTR _fe_sel, BSTR _node_sel);
```

**C#**

```
public void GetContents(long _group_obj, String _object_sel, String _fe_sel, String _node_sel);
```

**Visual Basic**

```
Public Sub GetContents(_group_obj As long, ByRef _object_sel As String, ByRef _fe_sel As String, ByRef _node_sel As String)
```

**Description**

Function returns selection lists of structure elements included in the specified "grouped" object.

**Version**

Available since version 7.5.

**II.7.1.2.5 GroupGiven****C++**

```
HRESULT GroupGiven(BSTR _object_sel, BSTR _fe_sel, BSTR _node_sel, VARIANT_BOOL _auto_include = True, long* ret);
```

**C#**

```
public long GroupGiven(String _object_sel, String _fe_sel, String _node_sel, bool
```

```
_auto_include = True);
```

#### Visual Basic

```
Public Function GroupGiven(_object_sel As String, _fe_sel As String, _node_sel As String,  
Optional _auto_include As Boolean = True) As long
```

#### Description

Function groups (combines into one object) structure elements described by selection lists. Function returns the number of a created object.

#### Version

Available since version 7.5.

### II.7.1.2.6 GroupSelected

#### C++

```
HRESULT GroupSelected(long* ret);
```

#### C#

```
public long GroupSelected();
```

#### Visual Basic

```
Public Function GroupSelected() As long
```

#### Description

Function groups (combines into one object) structure elements belonging to the current selection. Function returns the number of a created object.

#### Version

Available since version 7.5.

## II.8 IRobotObjectStructuralType

#### C++

```
enum IRobotObjectStructuralType;
```

#### C#

```
public enum IRobotObjectStructuralType;
```

#### Visual Basic

```
Public Enum IRobotObjectStructuralType
```

#### Members

Members	Description
I_OST_UNDEFINED = 0	Available since version 9.7.
I_OST_BEAM = 1	Available since version 9.7.
I_OST_COLUMN = 2	Available since version 9.7.
I_OST_SLAB = 4	Available since version 9.7.
I_OST_WALL = 5	Available since version 9.7.

#### Description

Type of object as a structure component.

## Version

Available since version 9.7.

# III Results

## Enumerations

	Name	Description
	IRobotUniversalResultType (see page 1013)	Types of results made available by the RobotUniversalResultAccess object.
	IRobotResultParamType (see page 1019)	Set of parameters describing calculation results.
	IRobotResultQueryReturnType (see page 1027)	
	IRobotResultStatusType (see page 1027)	Type of data describing status of results.

## Interfaces

	Name	Description
	IRobotResultServer (see page 997)	The server makes available all types of results obtained by performing structure calculations. .
	IRobotUniversalResultAccess (see page 1002)	Universal access interface for calculation results. In order to obtain the relevant result the object should be first appropriately parametrized by determining what result type, for which load case and for which structure element is to be obtained, and then ResultValue (see page 1009) should be taken. .
	IRobotExtremeResultServer (see page 1012)	Server of extreme results.
	IRobotStructureValues (see page 1013)	Physical properties for the structure model.
	IRobotResultRow (see page 1017)	Row of results calculated for determined values of a parameter set.
	IRobotResultRowSet (see page 1021)	Set of rows with calculation results.
	IRobotResultQueryParams (see page 1024)	Definition of a query to the result server.
	IRobotCalculationResume (see page 1028)	.

## III.1 Calculation results for nodes

### Interfaces

	Name	Description
	IRobtnodeResultServer (see page 848)	All types or results for nodes are made accessible by the appropriate server of the RobotNodeResultServer type. .
	IRobotReactionServer (see page 849)	The server providing access to values of reactions for nodes. .
	IRobtnodeDisplacementServer (see page 859)	The server provides access to all values of nodal displacements. .
	IRobtnodeBucklingServer (see page 861)	The server provides results of buckling analysis for nodes.
	IRobotPseudostaticForceServer (see page 863)	Server providing access to pseudostatic forces.

### III.1.1 IRobotNodeResultServer

#### Class Hierarchy

##### C++

```
interface IRobotNodeResultServer : IDispatch;
```

##### C#

```
public interface IRobotNodeResultServer;
```

#### Visual Basic

```
Public Interface IRobotNodeResultServer
```

#### Description

All types or results for nodes are made accessible by the appropriate server of the RobotNodeResultServer type. .

### III.1.1.1 IRobotNodeResultServer Members

The following tables list the members exposed by IRobotNodeResultServer.

#### Public Fields

	Name	Description
◆	Buckling ( [ see page 848 )	The server of results of buckling analysis for nodes.
◆	Displacements ( [ see page 849 )	Server provides access to nodal displacements .
◆	PseudostaticForces ( [ see page 849 )	Server providing access to values of pseudostatic forces Available since version 1.7.
◆	Reactions ( [ see page 849 )	Server provides access to reaction values in nodes..

### III.1.1.2 IRobotNodeResultServer Fields

The fields of the IRobotNodeResultServer class are listed here.

#### Public Fields

	Name	Description
◆	Buckling ( [ see page 848 )	The server of results of buckling analysis for nodes.
◆	Displacements ( [ see page 849 )	Server provides access to nodal displacements .
◆	PseudostaticForces ( [ see page 849 )	Server providing access to values of pseudostatic forces Available since version 1.7.
◆	Reactions ( [ see page 849 )	Server provides access to reaction values in nodes..

### III.1.1.2.1 Buckling

#### C++

```
HRESULT get_Buckling(IRobotNodeBucklingServer**);
```

#### C#

```
public IRobotNodeBucklingServer Buckling { get; }
```

#### Visual Basic

```
Public ReadOnly Buckling As IRobotNodeBucklingServer
```

#### Description

The server of results of buckling analysis for nodes.

### III.1.1.2.2 Displacements

**C++**

```
HRESULT get_Displacements(IRobotNodeDisplacementServer**);
```

**C#**

```
public IRobotNodeDisplacementServer Displacements { get; }
```

**Visual Basic**

```
Public ReadOnly Displacements As IRobotNodeDisplacementServer
```

**Description**

Server provides access to nodal displacements .

### III.1.1.2.3 PseudostaticForces

**C++**

```
HRESULT get_PseudostaticForces(IRobotPseudostaticForceServer**);
```

**C#**

```
public IRobotPseudostaticForceServer PseudostaticForces { get; }
```

**Visual Basic**

```
Public ReadOnly PseudostaticForces As IRobotPseudostaticForceServer
```

**Description**

Server providing access to values of pseudostatic forces Available since version 1.7.

### III.1.1.2.4 Reactions

**C++**

```
HRESULT get_Reactions(IRobotReactionServer**);
```

**C#**

```
public IRobotReactionServer Reactions { get; }
```

**Visual Basic**

```
Public ReadOnly Reactions As IRobotReactionServer
```

**Description**

Server provides access to reaction values in nodes. .

## III.1.2 IRobotReactionServer

**Class Hierarchy**

**C++**

```
interface IRobotReactionServer : IDispatch;
```

**C#**

```
public interface IRobotReactionServer;
```

**Visual Basic**

```
Public Interface IRobotReactionServer
```

## Description

The server providing access to values of reactions for nodes. .

### III.1.2.1 IRobotReactionServer Members

The following tables list the members exposed by IRobotReactionServer.

#### Public Methods

	Name	Description
≡	DDC ( <a href="#">see page 852</a> )	The function returns a structure containing Load Report reaction values for the indicated node and load case .
≡	DDCEx ( <a href="#">see page 852</a> )	Available since version 1.7.
≡	DDCLocal ( <a href="#">see page 852</a> )	The function returns a structure containing Load Report reaction values for the indicated node and load case .
≡	DDCLocalEx ( <a href="#">see page 852</a> )	Available since version 1.7.
≡	DDCSum ( <a href="#">see page 853</a> )	The function returns a structure containing values of reaction sums of the Load Report type for the indicated load case. .
≡	DDCSumEx ( <a href="#">see page 853</a> )	Available since version 1.7.
≡	DynCombDDC ( <a href="#">see page 853</a> )	Available since version 1.7.
≡	DynCombDDCLocal ( <a href="#">see page 854</a> )	Available since version 1.7.
≡	DynCombDDCSum ( <a href="#">see page 854</a> )	Available since version 1.7.
≡	DynCombLocal ( <a href="#">see page 854</a> )	Available since version 1.7.
≡	DynCombSum ( <a href="#">see page 854</a> )	Available since version 1.7.
≡	DynCombSumForce ( <a href="#">see page 855</a> )	Available since version 1.7.
≡	DynCombValue ( <a href="#">see page 855</a> )	Available since version 1.7.
≡	DynDDC ( <a href="#">see page 855</a> )	Available since version 1.7.
≡	DynDDCLocal ( <a href="#">see page 856</a> )	Available since version 1.7.
≡	DynDDCSum ( <a href="#">see page 856</a> )	Available since version 1.7.
≡	DynLocal ( <a href="#">see page 856</a> )	Available since version 1.7.
≡	DynSum ( <a href="#">see page 856</a> )	Available since version 1.7.
≡	DynSumForce ( <a href="#">see page 857</a> )	Available since version 1.7.
≡	DynValue ( <a href="#">see page 857</a> )	Available since version 1.7.
≡	Local ( <a href="#">see page 857</a> )	The function returns a structure containing reaction values in the local coordinate system for the indicated node and load case. .
≡	LocalEx ( <a href="#">see page 857</a> )	Available since version 1.7.
≡	Sum ( <a href="#">see page 858</a> )	The function returns a structure containing values of reaction sums for the indicated load case .
≡	SumEx ( <a href="#">see page 858</a> )	Available since version 1.7.
≡	SumForce ( <a href="#">see page 858</a> )	The function returns a structure containing the value of sums of forces for the indicated load case. .
≡	SumForceEx ( <a href="#">see page 859</a> )	Available since version 1.7.
≡	Value ( <a href="#">see page 859</a> )	The function returns a structure containing the values of reactions for a given node and load case..
≡	ValueEx ( <a href="#">see page 859</a> )	Function returns the structure containing reaction values for the specified node and load case. Available since version 1.7.

### III.1.2.2 IRobotReactionServer Methods

The methods of the IRobotReactionServer class are listed here.

## Public Methods

	Name	Description
»	DDC ( <a href="#">see page 852</a> )	The function returns a structure containing Load Report reaction values for the indicated node and load case .
»	DDCEx ( <a href="#">see page 852</a> )	Available since version 1.7.
»	DDCLocal ( <a href="#">see page 852</a> )	The function returns a structure containing Load Report reaction values for the indicated node and load case .
»	DDCLocalEx ( <a href="#">see page 852</a> )	Available since version 1.7.
»	DDCSum ( <a href="#">see page 853</a> )	The function returns a structure containing values of reaction sums of the Load Report type for the indicated load case. .
»	DDCSumEx ( <a href="#">see page 853</a> )	Available since version 1.7.
»	DynCombDDC ( <a href="#">see page 853</a> )	Available since version 1.7.
»	DynCombDDCLocal ( <a href="#">see page 854</a> )	Available since version 1.7.
»	DynCombDDCSum ( <a href="#">see page 854</a> )	Available since version 1.7.
»	DynCombLocal ( <a href="#">see page 854</a> )	Available since version 1.7.
»	DynCombSum ( <a href="#">see page 854</a> )	Available since version 1.7.
»	DynCombSumForce ( <a href="#">see page 855</a> )	Available since version 1.7.
»	DynCombValue ( <a href="#">see page 855</a> )	Available since version 1.7.
»	DynDDC ( <a href="#">see page 855</a> )	Available since version 1.7.
»	DynDDCLocal ( <a href="#">see page 856</a> )	Available since version 1.7.
»	DynDDCSum ( <a href="#">see page 856</a> )	Available since version 1.7.
»	DynLocal ( <a href="#">see page 856</a> )	Available since version 1.7.
»	DynSum ( <a href="#">see page 856</a> )	Available since version 1.7.
»	DynSumForce ( <a href="#">see page 857</a> )	Available since version 1.7.
»	DynValue ( <a href="#">see page 857</a> )	Available since version 1.7.
»	Local ( <a href="#">see page 857</a> )	The function returns a structure containing reaction values in the local coordinate system for the indicated node and load case. .
»	LocalEx ( <a href="#">see page 857</a> )	Available since version 1.7.
»	Sum ( <a href="#">see page 858</a> )	The function returns a structure containing values of reaction sums for the indicated load case .
»	SumEx ( <a href="#">see page 858</a> )	Available since version 1.7.
»	SumForce ( <a href="#">see page 858</a> )	The function returns a structure containing the value of sums of forces for the indicated load case. .
»	SumForceEx ( <a href="#">see page 859</a> )	Available since version 1.7.
»	Value ( <a href="#">see page 859</a> )	The function returns a structure containing the values of reactions for a given node and load case. .
»	ValueEx ( <a href="#">see page 859</a> )	Function returns the structure containing reaction values for the specified node and load case. Available since version 1.7.

### III.1.2.2.1 DDC

#### C++

```
HRESULT DDC(long _node, long _case_num, IRobotReactionData** ret);
```

#### C#

```
public IRobotReactionData DDC(long _node, long _case_num);
```

#### Visual Basic

```
Public Function DDC(_node As long, _case_num As long) As IRobotReactionData
```

## Description

The function returns a structure containing Load Report reaction values for the indicated node and load case .

### III.1.2.2.2 DDCEx

#### C++

```
HRESULT DDCEx(long _node, long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

#### C#

```
public IRobotReactionData DDCEx(long _node, long _case, long _case_cmpnt = 1);
```

#### Visual Basic

```
Public Function DDCEx(_node As long, _case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

## Description

Available since version 1.7.

### III.1.2.2.3 DDCLocal

#### C++

```
HRESULT DDCLocal(long _node, long _case_num, IRobotReactionData** ret);
```

#### C#

```
public IRobotReactionData DDCLocal(long _node, long _case_num);
```

#### Visual Basic

```
Public Function DDCLocal(_node As long, _case_num As long) As IRobotReactionData
```

## Description

The function returns a structure containing Load Report reaction values for the indicated node and load case .

### III.1.2.2.4 DDCLocalEx

#### C++

```
HRESULT DDCLocalEx(long _node, long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

#### C#

```
public IRobotReactionData DDCLocalEx(long _node, long _case, long _case_cmpnt = 1);
```

#### Visual Basic

```
Public Function DDCLocalEx(_node As long, _case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

## Description

Available since version 1.7.

### III.1.2.2.5 DDCSum

#### C++

```
HRESULT DDCSum(long _case_num, IRobotReactionData** ret);
```

#### C#

```
public IRobotReactionData DDCSum(long _case_num);
```

**Visual Basic**

```
Public Function DDCSum(_case_num As long) As IRobotReactionData
```

**Description**

The function returns a structure containing values of reaction sums of the Load Report type for the indicated load case. .

**III.1.2.2.6 DDCSumEx****C++**

```
HRESULT DDCSumEx(long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DDCSumEx(long _case, long _case_cmpnt = 1);
```

**Visual Basic**

```
Public Function DDCSumEx(_case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.7 DynCombDDC****C++**

```
HRESULT DynCombDDC(long _node, long _case, IRobotModeCombinationType _mode_cmb,
IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombDDC(long _node, long _case, IRobotModeCombinationType
_mode_cmb);
```

**Visual Basic**

```
Public Function DynCombDDC(_node As long, _case As long, _mode_cmb As
IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.8 DynCombDDCLocal****C++**

```
HRESULT DynCombDDCLocal(long _node, long _case, IRobotModeCombinationType _mode_cmb,
IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombDDCLocal(long _node, long _case, IRobotModeCombinationType
_mode_cmb);
```

**Visual Basic**

```
Public Function DynCombDDCLocal(_node As long, _case As long, _mode_cmb As
IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.9 DynCombDDCSum

**C++**

```
HRESULT DynCombDDCSum(long _case, IRobotModeCombinationType _mode_cmb, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombDDCSum(long _case, IRobotModeCombinationType _mode_cmb);
```

**Visual Basic**

```
Public Function DynCombDDCSum(_case As long, _mode_cmb As IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.10 DynCombLocal

**C++**

```
HRESULT DynCombLocal(long _node, long _case, IRobotModeCombinationType _mode_cmb, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombLocal(long _node, long _case, IRobotModeCombinationType _mode_cmb);
```

**Visual Basic**

```
Public Function DynCombLocal(_node As long, _case As long, _mode_cmb As IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.11 DynCombSum

**C++**

```
HRESULT DynCombSum(long _case, IRobotModeCombinationType _mode_cmb, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombSum(long _case, IRobotModeCombinationType _mode_cmb);
```

**Visual Basic**

```
Public Function DynCombSum(_case As long, _mode_cmb As IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.12 DynCombSumForce

**C++**

```
HRESULT DynCombSumForce(long _case, IRobotModeCombinationType _mode_cmb, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombSumForce(long _case, IRobotModeCombinationType _mode_cmb);
```

**Visual Basic**

```
Public Function DynCombSumForce(_case As long, _mode_cmb As IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.13 DynCombValue****C++**

```
HRESULT DynCombValue(long _node, long _case, IRobotModeCombinationType _mode_cmb,  
IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynCombValue(long _node, long _case, IRobotModeCombinationType  
_mode_cmb);
```

**Visual Basic**

```
Public Function DynCombValue(_node As long, _case As long, _mode_cmb As  
IRobotModeCombinationType) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.14 DynDDC****C++**

```
HRESULT DynDDC(long _node, long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynDDC(long _node, long _case, long _mode);
```

**Visual Basic**

```
Public Function DynDDC(_node As long, _case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.15 DynDDCLocal****C++**

```
HRESULT DynDDCLocal(long _node, long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynDDCLocal(long _node, long _case, long _mode);
```

**Visual Basic**

```
Public Function DynDDCLocal(_node As long, _case As long, _mode As long) As  
IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.16 DynDDCSum

**C++**

```
HRESULT DynDDCSum(long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynDDCSum(long _case, long _mode);
```

**Visual Basic**

```
Public Function DynDDCSum(_case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.17 DynLocal

**C++**

```
HRESULT DynLocal(long _node, long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynLocal(long _node, long _case, long _mode);
```

**Visual Basic**

```
Public Function DynLocal(_node As long, _case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.18 DynSum

**C++**

```
HRESULT DynSum(long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynSum(long _case, long _mode);
```

**Visual Basic**

```
Public Function DynSum(_case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

### III.1.2.2.19 DynSumForce

**C++**

```
HRESULT DynSumForce(long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynSumForce(long _case, long _mode);
```

**Visual Basic**

```
Public Function DynSumForce(_case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.20 DynValue****C++**

```
HRESULT DynValue(long _node, long _case, long _mode, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData DynValue(long _node, long _case, long _mode);
```

**Visual Basic**

```
Public Function DynValue(_node As long, _case As long, _mode As long) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.21 Local****C++**

```
HRESULT Local(long _node, long _case_num, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData Local(long _node, long _case_num);
```

**Visual Basic**

```
Public Function Local(_node As long, _case_num As long) As IRobotReactionData
```

**Description**

The function returns a structure containing reaction values in the local coordinate system for the indicated node and load case..

**III.1.2.2.22 LocalEx****C++**

```
HRESULT LocalEx(long _node, long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData LocalEx(long _node, long _case, long _case_cmpnt = 1);
```

**Visual Basic**

```
Public Function LocalEx(_node As long, _case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.23 Sum****C++**

```
HRESULT Sum(long _case_num, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData Sum(long _case_num);
```

**Visual Basic**

```
Public Function Sum(_case_num As long) As IRobotReactionData
```

**Description**

The function returns a structure containing values of reaction sums for the indicated load case .

**III.1.2.2.24 SumEx****C++**

```
HRESULT SumEx(long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData SumEx(long _case, long _case_cmpnt = 1);
```

**Visual Basic**

```
Public Function SumEx(_case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.25 SumForce****C++**

```
HRESULT SumForce(long _case_num, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData SumForce(long _case_num);
```

**Visual Basic**

```
Public Function SumForce(_case_num As long) As IRobotReactionData
```

**Description**

The function returns a structure containing the value of sums of forces for the indicated load case..

**III.1.2.2.26 SumForceEx****C++**

```
HRESULT SumForceEx(long _case, long _case_cmpnt = 1, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData SumForceEx(long _case, long _case_cmpnt = 1);
```

**Visual Basic**

```
Public Function SumForceEx(_case As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

**Description**

Available since version 1.7.

**III.1.2.2.27 Value****C++**

```
HRESULT Value(long _node, long _case_num, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData Value(long _node, long _case_num);
```

**Visual Basic**

```
Public Function Value(_node As long, _case_num As long) As IRobotReactionData
```

**Description**

The function returns a structure containing the values of reactions for a given node and load case. .

**III.1.2.2.28 ValueEx****C++**

```
HRESULT ValueEx(long _node_num, long _case_num, long _case_cmpnt = 1, IRobotReactionData** ret);
```

**C#**

```
public IRobotReactionData ValueEx(long _node_num, long _case_num, long _case_cmpnt = 1);
```

**Visual Basic**

```
Public Function ValueEx(_node_num As long, _case_num As long, Optional _case_cmpnt As long = 1) As IRobotReactionData
```

**Description**

Function returns the structure containing reaction values for the specified node and load case. Available since version 1.7.

**III.1.3 IRobotNodeDisplacementServer****Class Hierarchy****C++**

```
interface IRobotNodeDisplacementServer : IDispatch;
```

**C#**

```
public interface IRobotNodeDisplacementServer;
```

**Visual Basic**

```
Public Interface IRobotNodeDisplacementServer
```

**Description**

The server provides access to all values of nodal displacements. .

**III.1.3.1 IRobotNodeDisplacementServer Members**

The following tables list the members exposed by IRobotNodeDisplacementServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	DynCombValue (see page 860)	Function returns the data structure containing displacements for the specified node and defined mode combination of a selected load case. Available since version 1.7.
✳	DynValue (see page 861)	Function returns the data structure describing displacements in the defined node for a specified load case and specified mode. Available since version 1.7.
✳	Value (see page 861)	The function returns an object (set of data) describing displacements in the indicated node for the indicated load case. .

	ValueEx (see page 861)	Available since version 1.7.
--	------------------------	------------------------------

### III.1.3.2 IRobotNodeDisplacementServer Methods

The methods of the IRobotNodeDisplacementServer class are listed here.

#### Public Methods

	Name	Description
	DynCombValue (see page 860)	Function returns the data structure containing displacements for the specified node and defined mode combination of a selected load case. Available since version 1.7.
	DynValue (see page 861)	Function returns the data structure describing displacements in the defined node for a specified load case and specified mode. Available since version 1.7.
	Value (see page 861)	The function returns an object (set of data) describing displacements in the indicated node for the indicated load case. .
	ValueEx (see page 861)	Available since version 1.7.

#### III.1.3.2.1 DynCombValue

##### C++

```
HRESULT DynCombValue(long _node_num, long _case_num, IRobotModeCombinationType _mode_cmb,
IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData DynCombValue(long _node_num, long _case_num,
IRobotModeCombinationType _mode_cmb);
```

##### Visual Basic

```
Public Function DynCombValue(_node_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType) As IRobotDisplacementData
```

##### Description

Function returns the data structure containing displacements for the specified node and defined mode combination of a selected load case. Available since version 1.7.

#### III.1.3.2.2 DynValue

##### C++

```
HRESULT DynValue(long _node_num, long _case_num, long _mode, IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData DynValue(long _node_num, long _case_num, long _mode);
```

##### Visual Basic

```
Public Function DynValue(_node_num As long, _case_num As long, _mode As long) As
IRobotDisplacementData
```

##### Description

Function returns the data structure describing displacements in the defined node for a specified load case and specified mode. Available since version 1.7.

#### III.1.3.2.3 Value

##### C++

```
HRESULT Value(long _node_num, long _case_num, IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData Value(long _node_num, long _case_num);
```

**Visual Basic**

```
Public Function Value(_node_num As long, _case_num As long) As IRobotDisplacementData
```

**Description**

The function returns an object (set of data) describing displacements in the indicated node for the indicated load case. .

**III.1.3.2.4 ValueEx****C++**

```
HRESULT ValueEx(long _node, long _case, long _case_cmpnt, IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData ValueEx(long _node, long _case, long _case_cmpnt);
```

**Visual Basic**

```
Public Function ValueEx(_node As long, _case As long, _case_cmpnt As long) As IRobotDisplacementData
```

**Description**

Available since version 1.7.

**III.1.4 IRobotNodeBucklingServer****Class Hierarchy****C++**

```
interface IRobotNodeBucklingServer : IDispatch;
```

**C#**

```
public interface IRobotNodeBucklingServer;
```

**Visual Basic**

```
Public Interface IRobotNodeBucklingServer
```

**Description**

The server provides results of buckling analysis for nodes.

**III.1.4.1 IRobotNodeBucklingServer Members**

The following tables list the members exposed by IRobotNodeBucklingServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	EigenVector (🔗 see page 862)	The function returns the displacement vector for the indicated node, load case and mode.
💡	EigenVectorCmb (🔗 see page 862)	Available since version 1.7.

**III.1.4.2 IRobotNodeBucklingServer Methods**

The methods of the IRobotNodeBucklingServer class are listed here.

## Public Methods

	Name	Description
💡	EigenVector (see page 862)	The function returns the displacement vector for the indicated node, load case and mode.
💡	EigenVectorCmb (see page 862)	Available since version 1.7.

### III.1.4.2.1 EigenVector

#### C++

```
HRESULT EigenVector(long _node, long _case, long _mode, IRobotDisplacementData** ret);
```

#### C#

```
public IRobotDisplacementData EigenVector(long _node, long _case, long _mode);
```

#### Visual Basic

```
Public Function EigenVector(_node As long, _case As long, _mode As long) As IRobotDisplacementData
```

#### Description

The function returns the displacement vector for the indicated node, load case and mode.

### III.1.4.2.2 EigenVectorCmb

#### C++

```
HRESULT EigenVectorCmb(long _node, long _case, IRobotModeCombinationType _mode_cmb,
IRobotDisplacementData** ret);
```

#### C#

```
public IRobotDisplacementData EigenVectorCmb(long _node, long _case,
IRobotModeCombinationType _mode_cmb);
```

#### Visual Basic

```
Public Function EigenVectorCmb(_node As long, _case As long, _mode_cmb As IRobotModeCombinationType) As IRobotDisplacementData
```

#### Description

Available since version 1.7.

## III.1.5 IRobotPseudostaticForceServer

#### Class Hierarchy

#### C++

```
interface IRobotPseudostaticForceServer : IDispatch;
```

#### C#

```
public interface IRobotPseudostaticForceServer;
```

#### Visual Basic

```
Public Interface IRobotPseudostaticForceServer
```

#### Description

Server providing access to pseudostatic forces.

### III.1.5.1 IRobotPseudostaticForceServer Members

The following tables list the members exposed by IRobotPseudostaticForceServer.

#### Public Methods

	Name	Description
💡	CombValue (see page 863)	Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.
💡	Value (see page 864)	Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.

### III.1.5.2 IRobotPseudostaticForceServer Methods

The methods of the IRobotPseudostaticForceServer class are listed here.

#### Public Methods

	Name	Description
💡	CombValue (see page 863)	Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.
💡	Value (see page 864)	Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.

#### III.1.5.2.1 CombValue

##### C++

```
HRESULT CombValue(long _node_num, long _case_num, IRobotModeCombinationType _mode_cmb,
IRobotForcesData** ret);
```

##### C#

```
public IRobotForcesData CombValue(long _node_num, long _case_num, IRobotModeCombinationType
_mode_cmb);
```

##### Visual Basic

```
Public Function CombValue(_node_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType) As IRobotForcesData
```

##### Description

Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.

#### III.1.5.2.2 Value

##### C++

```
HRESULT Value(long _node_num, long _case_num, long _mode_num, IRobotForcesData** ret);
```

##### C#

```
public IRobotForcesData Value(long _node_num, long _case_num, long _mode_num);
```

##### Visual Basic

```
Public Function Value(_node_num As long, _case_num As long, _mode_num As long) As
IRobotForcesData
```

##### Description

Function returns pseudostatic force values in a specified node for a defined load case and mode. Available since version 1.7.

## III.2 Calculation results for bars

### Interfaces

	Name	Description
↪	IRobotBarResultServer (see page 864)	Server makes accessible all types of calculation results for bars. .
↪	IRobotBarForceServer (see page 867)	Server provides access to values of forces and moments for bars. .
↪	IRobotBarDeflectionServer (see page 870)	The server makes deflection values for bars available.
↪	IRobotBarStressServer (see page 874)	Server allows to get stress values for bars.
↪	IRobotBarBucklingServer (see page 876)	The server provides results of buckling analysis for bars.
↪	IRobotBarDisplacementServer (see page 877)	Server providing access to displacement results for bars. .

### III.2.1 IRobotBarResultServer

#### Class Hierarchy

#### C++

```
interface IRobotBarResultServer : IDispatch;
```

#### C#

```
public interface IRobotBarResultServer;
```

#### Visual Basic

```
Public Interface IRobotBarResultServer
```

#### Description

Server makes accessible all types of calculation results for bars. .

#### III.2.1.1 IRobotBarResultServer Members

The following tables list the members exposed by IRobotBarResultServer.

#### Public Fields

	Name	Description
❖	Buckling (see page 865)	Server of results of buckling analysis for bars.
❖	Deflections (see page 865)	Deflections.
❖	Displacements (see page 866)	Displacement server.
❖	Forces (see page 866)	Internal forces.
❖	Stresses (see page 866)	Stresses.

#### Public Methods

	Name	Description
❖	Geolimperfections (see page 867)	Function returns calculated deflection values in individual directions for the indicated bar.

#### III.2.1.2 IRobotBarResultServer Fields

The fields of the IRobotBarResultServer class are listed here.

## Public Fields

	Name	Description
◆	Buckling (see page 865)	Server of results of buckling analysis for bars.
◆	Deflections (see page 865)	Deflections.
◆	Displacements (see page 866)	Displacement server.
◆	Forces (see page 866)	Internal forces.
◆	Stresses (see page 866)	Stresses.

### III.2.1.2.1 Buckling

#### C++

```
HRESULT get_Buckling(IRobotBarBucklingServer**);
```

#### C#

```
public IRobotBarBucklingServer Buckling { get; }
```

#### Visual Basic

```
Public ReadOnly Buckling As IRobotBarBucklingServer
```

#### Description

Server of results of buckling analysis for bars.

### III.2.1.2.2 Deflections

#### C++

```
HRESULT get_Deflections(IRobotBarDeflectionServer**);
```

#### C#

```
public IRobotBarDeflectionServer Deflections { get; }
```

#### Visual Basic

```
Public ReadOnly Deflections As IRobotBarDeflectionServer
```

#### Description

Deflections.

### III.2.1.2.3 Displacements

#### C++

```
HRESULT get_Displacements(IRobotBarDisplacementServer**);
```

#### C#

```
public IRobotBarDisplacementServer Displacements { get; }
```

#### Visual Basic

```
Public ReadOnly Displacements As IRobotBarDisplacementServer
```

#### Description

Displacement server.

#### Version

Available since version 2.5.

### III.2.1.2.4 Forces

#### C++

```
HRESULT get_Forces(IRobotBarForceServer*);
```

#### C#

```
public IRobotBarForceServer Forces { get; }
```

#### Visual Basic

```
Public ReadOnly Forces As IRobotBarForceServer
```

#### Description

Internal forces.

### III.2.1.2.5 Stresses

#### C++

```
HRESULT get_Stresses(IRobotBarStressServer**);
```

#### C#

```
public IRobotBarStressServer Stresses { get; }
```

#### Visual Basic

```
Public ReadOnly Stresses As IRobotBarStressServer
```

#### Description

Stresses.

### III.2.1.3 IRobotBarResultServer Methods

The methods of the IRobotBarResultServer class are listed here.

#### Public Methods

	Name	Description
✳	GeoImperfections (see page 867)	Function returns calculated deflection values in individual directions for the indicated bar.

### III.2.1.3.1 GeoImperfections

#### C++

```
HRESULT GeoImperfections(long _bar, IRobotBarDeflectionData** ret);
```

#### C#

```
public IRobotBarDeflectionData GeoImperfections(long _bar);
```

#### Visual Basic

```
Public Function GeoImperfections(_bar As long) As IRobotBarDeflectionData
```

#### Description

Function returns calculated deflection values in individual directions for the indicated bar.

#### Version

Available since version 3.

## III.2.2 IRobotBarForceServer

### Class Hierarchy

#### C++

```
interface IRobotBarForceServer : IDispatch;
```

#### C#

```
public interface IRobotBarForceServer;
```

#### Visual Basic

```
Public Interface IRobotBarForceServer
```

### Description

Server provides access to values of forces and moments for bars. .

## III.2.2.1 IRobotBarForceServer Members

The following tables list the members exposed by IRobotBarForceServer.

### Public Methods

	Name	Description
➊	DynCombValue (see page 868)	Available since version 1.7.
➋	DynCombValueByNPoints (see page 868)	Available since version 1.7.
➌	DynValue (see page 869)	Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and point on the bar. Available since version 1.7.
➍	DynValueByNPoints (see page 869)	Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and point on the bar. Available since version 1.7.
➎	Value (see page 869)	The function returns a structure containing the values of all forces and moments for the indicated bar, load case and point along a bar. .
➏	ValueByNPoints (see page 870)	The function returns a structure containing the values of all forces and moments for the indicated bar, load case at the indicated division point. .
➐	ValueByNPointsEx (see page 870)	Available since version 1.7.
➑	ValueEx (see page 870)	Available since version 1.7.

## III.2.2.2 IRobotBarForceServer Methods

The methods of the IRobotBarForceServer class are listed here.

### Public Methods

	Name	Description
➊	DynCombValue (see page 868)	Available since version 1.7.
➋	DynCombValueByNPoints (see page 868)	Available since version 1.7.
➌	DynValue (see page 869)	Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and point on the bar. Available since version 1.7.
➍	DynValueByNPoints (see page 869)	Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and point on the bar. Available since version 1.7.
➎	Value (see page 869)	The function returns a structure containing the values of all forces and moments for the indicated bar, load case and point along a bar. .

	ValueByNPoints (see page 870)	The function returns a structure containing the values of all forces and moments for the indicated bar, load case at the indicated division point. .
	ValueByNPointsEx (see page 870)	Available since version 1.7.
	ValueEx (see page 870)	Available since version 1.7.

### III.2.2.2.1 DynCombValue

C++

```
HRESULT DynCombValue(long _bar_num, long _case_num, IRobotModeCombinationType _mode_cmb,
double _point, IRobotBarForceData** ret);
```

C#

```
public IRobotBarForceData DynCombValue(long _bar_num, long _case_num,
IRobotModeCombinationType _mode_cmb, double _point);
```

Visual Basic

```
Public Function DynCombValue(_bar_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType, _point As double) As IRobotBarForceData
```

Description

Available since version 1.7.

### III.2.2.2.2 DynCombValueByNPoints

C++

```
HRESULT DynCombValueByNPoints(long _bar_num, long _case_num, IRobotModeCombinationType
_mode_cmb, int _no_points, int _point, IRobotBarForceData** ret);
```

C#

```
public IRobotBarForceData DynCombValueByNPoints(long _bar_num, long _case_num,
IRobotModeCombinationType _mode_cmb, int _no_points, int _point);
```

Visual Basic

```
Public Function DynCombValueByNPoints(_bar_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType, _no_points As int, _point As int) As IRobotBarForceData
```

Description

Available since version 1.7.

### III.2.2.2.3 DynValue

C++

```
HRESULT DynValue(long _bar_num, long _case_num, long _mode, double _point,
IRobotBarForceData** ret);
```

C#

```
public IRobotBarForceData DynValue(long _bar_num, long _case_num, long _mode, double
_point);
```

Visual Basic

```
Public Function DynValue(_bar_num As long, _case_num As long, _mode As long, _point As
double) As IRobotBarForceData
```

Description

Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and

point on the bar. Available since version 1.7.

### III.2.2.2.4 DynValueByNPoints

#### C++

```
HRESULT DynValueByNPoints(long _bar_num, long _case_num, long _mode, int _no_points, int
_point, IRobotBarForceData** ret);
```

#### C#

```
public IRobotBarForceData DynValueByNPoints(long _bar_num, long _case_num, long _mode, int
_no_points, int _point);
```

#### Visual Basic

```
Public Function DynValueByNPoints(_bar_num As long, _case_num As long, _mode As long,
_no_points As int, _point As int) As IRobotBarForceData
```

#### Description

Function returns the structure containing values of all forces and moments for a specified bar, mode of the load case and point on the bar. Available since version 1.7.

### III.2.2.2.5 Value

#### C++

```
HRESULT Value(long _bar_num, long _case_num, double _point, IRobotBarForceData** ret);
```

#### C#

```
public IRobotBarForceData Value(long _bar_num, long _case_num, double _point);
```

#### Visual Basic

```
Public Function Value(_bar_num As long, _case_num As long, _point As double) As
IRobotBarForceData
```

#### Description

The function returns a structure containing the values of all forces and moments for the indicated bar, load case and point along a bar. .

### III.2.2.2.6 ValueByNPoints

#### C++

```
HRESULT ValueByNPoints(long _bar_num, long _case_num, int _no_points, int _point,
IRobotBarForceData* ret);
```

#### C#

```
public IRobotBarForceData ValueByNPoints(long _bar_num, long _case_num, int _no_points, int
_point);
```

#### Visual Basic

```
Public Function ValueByNPoints(_bar_num As long, _case_num As long, _no_points As int,
_point As int) As IRobotBarForceData
```

#### Description

The function returns a structure containing the values of all forces and moments for the indicated bar, load case at the indicated division point. .

### III.2.2.2.7 ValueByNPointsEx

**C++**

```
HRESULT ValueByNPointsEx(long _bar_num, long _case_num, long _case_cmpnt, int _no_points,
int _point, IRobotBarForceData** ret);
```

**C#**

```
public IRobotBarForceData ValueByNPointsEx(long _bar_num, long _case_num, long _case_cmpnt,
int _no_points, int _point);
```

**Visual Basic**

```
Public Function ValueByNPointsEx(_bar_num As long, _case_num As long, _case_cmpnt As long,
_no_points As int, _point As int) As IRobotBarForceData
```

**Description**

Available since version 1.7.

### III.2.2.8 ValueEx

**C++**

```
HRESULT ValueEx(long _bar_num, long _case_num, long _case_cmpnt, double _point,
IRobotBarForceData** ret);
```

**C#**

```
public IRobotBarForceData ValueEx(long _bar_num, long _case_num, long _case_cmpnt, double
_point);
```

**Visual Basic**

```
Public Function ValueEx(_bar_num As long, _case_num As long, _case_cmpnt As long, _point As
double) As IRobotBarForceData
```

**Description**

Available since version 1.7.

## III.2.3 IRobotBarDeflectionServer

**Class Hierarchy**

**C++**

```
interface IRobotBarDeflectionServer : IDispatch;
```

**C#**

```
public interface IRobotBarDeflectionServer;
```

**Visual Basic**

```
Public Interface IRobotBarDeflectionServer
```

**Description**

The server makes deflection values for bars available.

### III.2.3.1 IRobotBarDeflectionServer Members

The following tables list the members exposed by IRobotBarDeflectionServer.

## Public Methods

	Name	Description
ESH	DynCombMaxValue (see page 872)	The function makes available the maximum deflections for the square combination and appropriate positions on a bar.
ESH	DynCombValue (see page 872)	The function makes available the deflection values for an indicated combination.
ESH	DynMaxValue (see page 872)	The function makes maximum deflection values and appropriate positions on a bar in dynamics available.
ESH	DynValue (see page 872)	The function gives back deflection values in dynamics.
ESH	MaxValue (see page 873)	The function gives back maximum deflection values for a given bar, load case and appropriate positions on a bar.
ESH	MaxValueEx (see page 873)	Available since version 1.7.
ESH	Value (see page 873)	The function gives back a data stucture containing deflection values for a given bar, load case and a point on a bar. .
ESH	ValueEx (see page 874)	Available since version 1.7.

### III.2.3.2 IRobotBarDeflectionServer Methods

The methods of the IRobotBarDeflectionServer class are listed here.

## Public Methods

	Name	Description
ESH	DynCombMaxValue (see page 872)	The function makes available the maximum deflections for the square combination and appropriate positions on a bar.
ESH	DynCombValue (see page 872)	The function makes available the deflection values for an indicated combination.
ESH	DynMaxValue (see page 872)	The function makes maximum deflection values and appropriate positions on a bar in dynamics available.
ESH	DynValue (see page 872)	The function gives back deflection values in dynamics.
ESH	MaxValue (see page 873)	The function gives back maximum deflection values for a given bar, load case and appropriate positions on a bar.
ESH	MaxValueEx (see page 873)	Available since version 1.7.
ESH	Value (see page 873)	The function gives back a data stucture containing deflection values for a given bar, load case and a point on a bar. .
ESH	ValueEx (see page 874)	Available since version 1.7.

#### III.2.3.2.1 DynCombMaxValue

##### C++

```
HRESULT DynCombMaxValue(long _bar_num, long _case_num, IRobotModeCombinationType _cmb,
IRobotBarDeflectionData** ret);
```

##### C#

```
public IRobotBarDeflectionData DynCombMaxValue(long _bar_num, long _case_num,
IRobotModeCombinationType _cmb);
```

##### Visual Basic

```
Public Function DynCombMaxValue(_bar_num As long, _case_num As long, _cmb As
IRobotModeCombinationType) As IRobotBarDeflectionData
```

##### Description

The function makes available the maximum deflections for the square combination and appropriate positions on a bar.

### III.2.3.2.2 DynCombValue

**C++**

```
HRESULT DynCombValue(long _bar_num, long _case_num, double _pos, IRobotModeCombinationType
_cmb, IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData DynCombValue(long _bar_num, long _case_num, double _pos,
IRobotModeCombinationType _cmb);
```

**Visual Basic**

```
Public Function DynCombValue(_bar_num As long, _case_num As long, _pos As double, _cmb As
IRobotModeCombinationType) As IRobotBarDeflectionData
```

**Description**

The function makes available the deflection values for an indicated combination.

### III.2.3.2.3 Dyn.MaxValue

**C++**

```
HRESULT Dyn.MaxValue(long _bar_num, long _case_num, long _mode_num,
IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData Dyn.MaxValue(long _bar_num, long _case_num, long _mode_num);
```

**Visual Basic**

```
Public Function Dyn.MaxValue(_bar_num As long, _case_num As long, _mode_num As long) As
IRobotBarDeflectionData
```

**Description**

The function makes maximum deflection values and appropriate positions on a bar in dynamics available.

### III.2.3.2.4 DynValue

**C++**

```
HRESULT DynValue(long _bar_num, long _case_num, long _mode_num, double _pos,
IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData DynValue(long _bar_num, long _case_num, long _mode_num,
double _pos);
```

**Visual Basic**

```
Public Function DynValue(_bar_num As long, _case_num As long, _mode_num As long, _pos As
double) As IRobotBarDeflectionData
```

**Description**

The function gives back deflection values in dynamics.

### III.2.3.2.5 MaxValue

**C++**

```
HRESULT MaxValue(long _bar_num, long _case_num, IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData MaxValue(long _bar_num, long _case_num);
```

**Visual Basic**

```
Public Function MaxValue(_bar_num As long, _case_num As long) As IRobotBarDeflectionData
```

**Description**

The function gives back maximum deflection values for a given bar, load case and appropriate positions on a bar.

**III.2.3.2.6 MaxValueEx****C++**

```
HRESULT MaxValueEx(long _bar_num, long _case_num, long _case_cmpnt,  
IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData MaxValueEx(long _bar_num, long _case_num, long _case_cmpnt);
```

**Visual Basic**

```
Public Function MaxValueEx(_bar_num As long, _case_num As long, _case_cmpnt As long) As  
IRobotBarDeflectionData
```

**Description**

Available since version 1.7.

**III.2.3.2.7 Value****C++**

```
HRESULT Value(long _bar_num, long _case_num, double _pos, IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData Value(long _bar_num, long _case_num, double _pos);
```

**Visual Basic**

```
Public Function Value(_bar_num As long, _case_num As long, _pos As double) As  
IRobotBarDeflectionData
```

**Description**

The function gives back a data stucture containing deflection values for a given bar, load case and a point on a bar. .

**III.2.3.2.8 ValueEx****C++**

```
HRESULT ValueEx(long _bar_num, long _case_num, long _case_cmpnt, double _pos,  
IRobotBarDeflectionData** ret);
```

**C#**

```
public IRobotBarDeflectionData ValueEx(long _bar_num, long _case_num, long _case_cmpnt,  
double _pos);
```

**Visual Basic**

```
Public Function ValueEx(_bar_num As long, _case_num As long, _case_cmpnt As long, _pos As  
double) As IRobotBarDeflectionData
```

## Description

Available since version 1.7.

### III.2.4 IRobotBarStressServer

#### Class Hierarchy

##### C++

```
interface IRobotBarStressServer : IDispatch;
```

##### C#

```
public interface IRobotBarStressServer;
```

#### Visual Basic

```
Public Interface IRobotBarStressServer
```

#### Description

Server allows to get stress values for bars.

### III.2.4.1 IRobotBarStressServer Members

The following tables list the members exposed by IRobotBarStressServer.

#### Public Methods

	Name	Description
➊	DynCombValue (see page 875)	Function returns stress values for a selected load case and specified mode combination, in the selected point on the bar. Available since version 1.7.
➋	DynValue (see page 875)	Function returns stress values for a selected load case and mode, in the selected point on the bar. Available since version 1.7.
➌	Value (see page 875)	The function makes the stress values for a given bar, load case and relative position on a bar available.
➍	ValueEx (see page 876)	Available since version 1.7.

### III.2.4.2 IRobotBarStressServer Methods

The methods of the IRobotBarStressServer class are listed here.

#### Public Methods

	Name	Description
➊	DynCombValue (see page 875)	Function returns stress values for a selected load case and specified mode combination, in the selected point on the bar. Available since version 1.7.
➋	DynValue (see page 875)	Function returns stress values for a selected load case and mode, in the selected point on the bar. Available since version 1.7.
➌	Value (see page 875)	The function makes the stress values for a given bar, load case and relative position on a bar available.
➍	ValueEx (see page 876)	Available since version 1.7.

### III.2.4.2.1 DynCombValue

##### C++

```
HRESULT DynCombValue(long _bar_num, long _case_num, IRobotModeCombinationType _mode_cmb,
double _pos, IRobotBarStressData** ret);
```

**C#**

```
public IRobotBarStressData DynCombValue(long _bar_num, long _case_num,
IRobotModeCombinationType _mode_cmb, double _pos);
```

**Visual Basic**

```
Public Function DynCombValue(_bar_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType, _pos As double) As IRobotBarStressData
```

**Description**

Function returns stress values for a selected load case and specified mode combination, in the selected point on the bar.  
Available since version 1.7.

**III.2.4.2.2 DynValue****C++**

```
HRESULT DynValue(long _bar_num, long _case_num, long _mode_num, double _pos,
IRobotBarStressData** ret);
```

**C#**

```
public IRobotBarStressData DynValue(long _bar_num, long _case_num, long _mode_num, double
_pos);
```

**Visual Basic**

```
Public Function DynValue(_bar_num As long, _case_num As long, _mode_num As long, _pos As
double) As IRobotBarStressData
```

**Description**

Function returns stress values for a selected load case and mode, in the selected point on the bar. Available since version 1.7.

**III.2.4.2.3 Value****C++**

```
HRESULT Value(long _bar_num, long _case_num, double _pos, IRobotBarStressData** ret);
```

**C#**

```
public IRobotBarStressData Value(long _bar_num, long _case_num, double _pos);
```

**Visual Basic**

```
Public Function Value(_bar_num As long, _case_num As long, _pos As double) As
IRobotBarStressData
```

**Description**

The function makes the stress values for a given bar, load case and relative position on a bar available.

**III.2.4.2.4 ValueEx****C++**

```
HRESULT ValueEx(long _bar_num, long _case_num, long _case_cmpnt, double _pos,
IRobotBarStressData** ret);
```

**C#**

```
public IRobotBarStressData ValueEx(long _bar_num, long _case_num, long _case_cmpnt, double
_pos);
```

## Visual Basic

```
Public Function ValueEx(_bar_num As long, _case_num As long, _case_cmpnt As long, _pos As double) As IRobotBarStressData
```

## Description

Available since version 1.7.

## III.2.5 IRobotBarBucklingServer

### Class Hierarchy

#### C++

```
interface IRobotBarBucklingServer : IDispatch;
```

#### C#

```
public interface IRobotBarBucklingServer;
```

## Visual Basic

```
Public Interface IRobotBarBucklingServer
```

## Description

The server provides results of buckling analysis for bars.

## III.2.5.1 IRobotBarBucklingServer Members

The following tables list the members exposed by IRobotBarBucklingServer.

### Public Methods

	Name	Description
💡	CriticalCoef (see page 877)	The function returns the value of the critical coefficient for the indicated load case and mode.
💡	EigenValue (see page 877)	The function returns eigenvalues for the indicated bar, load case and mode.

## III.2.5.2 IRobotBarBucklingServer Methods

The methods of the IRobotBarBucklingServer class are listed here.

### Public Methods

	Name	Description
💡	CriticalCoef (see page 877)	The function returns the value of the critical coefficient for the indicated load case and mode.
💡	EigenValue (see page 877)	The function returns eigenvalues for the indicated bar, load case and mode.

## III.2.5.2.1 CriticalCoef

#### C++

```
HRESULT CriticalCoef(long _case_num, long _mode_num, double* ret);
```

#### C#

```
public double CriticalCoef(long _case_num, long _mode_num);
```

## Visual Basic

```
Public Function CriticalCoef(_case_num As long, _mode_num As long) As double
```

## Description

The function returns the value of the critical coefficient for the indicated load case and mode.

### III.2.5.2.2 EigenValue

#### C++

```
HRESULT EigenValue(long _bar, long _case, long _mode, IRobotBarBucklingData** ret);
```

#### C#

```
public IRobotBarBucklingData EigenValue(long _bar, long _case, long _mode);
```

#### Visual Basic

```
Public Function EigenValue(_bar As long, _case As long, _mode As long) As IRobotBarBucklingData
```

## Description

The function returns eigenvalues for the indicated bar, load case and mode.

### III.2.6 IRobotBarDisplacementServer

#### Class Hierarchy

#### C++

```
interface IRobotBarDisplacementServer : IDispatch;
```

#### C#

```
public interface IRobotBarDisplacementServer;
```

#### Visual Basic

```
Public Interface IRobotBarDisplacementServer
```

## Description

Server providing access to displacement results for bars. .

## Version

Available since version 2.5.

### III.2.6.1 IRobotBarDisplacementServer Members

The following tables list the members exposed by IRobotBarDisplacementServer.

#### Public Methods

	Name	Description
➊	DynCombValue (¶ see page 878)	Function returns the structure containing displacement values for the indicated bar, load case and mode combinations.
➋	DynValue (¶ see page 878)	Function returns the structure containing displacement values for the indicated bar, load case and mode.
➌	Value (¶ see page 879)	Function returns the structure containing displacement values for the indicated bar and load case. .

### III.2.6.2 IRobotBarDisplacementServer Methods

The methods of the IRobotBarDisplacementServer class are listed here.

## Public Methods

	Name	Description
ESH	DynCombValue (see page 878)	Function returns the structure containing displacement values for the indicated bar, load case and mode combinations.
ESH	DynValue (see page 878)	Function returns the structure containing displacement values for the indicated bar, load case and mode.
ESH	Value (see page 879)	Function returns the structure containing displacement values for the indicated bar and load case. .

### III.2.6.2.1 DynCombValue

#### C++

```
HRESULT DynCombValue(long _bar_num, double _pos, long _case_num, IRobotModeCombinationType
_mode_cmb, IRobotDisplacementData** ret);
```

#### C#

```
public IRobotDisplacementData DynCombValue(long _bar_num, double _pos, long _case_num,
IRobotModeCombinationType _mode_cmb);
```

#### Visual Basic

```
Public Function DynCombValue(_bar_num As long, _pos As double, _case_num As long, _mode_cmb
As IRobotModeCombinationType) As IRobotDisplacementData
```

#### Description

Function returns the structure containing displacement values for the indicated bar, load case and mode combinations.

#### Version

Available since version 2.5.

### III.2.6.2.2 DynValue

#### C++

```
HRESULT DynValue(long _bar_num, double _pos, long _case_num, long _mode,
IRobotDisplacementData** ret);
```

#### C#

```
public IRobotDisplacementData DynValue(long _bar_num, double _pos, long _case_num, long
_mode);
```

#### Visual Basic

```
Public Function DynValue(_bar_num As long, _pos As double, _case_num As long, _mode As
long) As IRobotDisplacementData
```

#### Description

Function returns the structure containing displacement values for the indicated bar, load case and mode.

#### Version

Available since version 2.5.

### III.2.6.2.3 Value

#### C++

```
HRESULT Value(long _bar_num, double _pos, long _case_num, long _case_cmpnt = 1,
IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData Value(long _bar_num, double _pos, long _case_num, long
_case_cmpnt = 1);
```

**Visual Basic**

```
Public Function Value(_bar_num As long, _pos As double, _case_num As long, Optional
_case_cmpnt As long = 1) As IRobotDisplacementData
```

**Description**

Function returns the structure containing displacement values for the indicated bar and load case. .

**Version**

Available since version 2.5.

## III.3 Data structures containing complex calculation results

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotExtremeValueType ( <a href="#">see page 910</a> )	Set of results for which the extreme values are made accessible. .

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotReactionData ( <a href="#">see page 880</a> )	The structure describing the value of reactions. .
	IRobotDisplacementData ( <a href="#">see page 882</a> )	Structure describing the value of displacements. .
	IRobotBarForceData ( <a href="#">see page 884</a> )	Structure describing values of forces, moments and elastic ground reaction for a bar. .
	IRobotBarStressData ( <a href="#">see page 887</a> )	A structure includes an information about stresses in a selected bar point for a selected load case.
	IRobotBarDeflectionData ( <a href="#">see page 890</a> )	A type of data describing deflections for the selected bar and case in a given bar point.
	IRobotBarBucklingData ( <a href="#">see page 892</a> )	The structure describing the eigenvalues for a single member, load case and mode. .
	IRobotEigenvalues ( <a href="#">see page 894</a> )	Data structure containing eigenvalues. .
	IRobotForcesData ( <a href="#">see page 897</a> )	Data structure containing force and moment values.
	IRobotTimeHistoryResults ( <a href="#">see page 900</a> )	Results for time history analysis.
	IRobotExtremeValue ( <a href="#">see page 905</a> )	Strucuture providing access to information about the extreme value. .
	IRobotExtremeParams ( <a href="#">see page 908</a> )	Strucuture containing parameters for which the extreme value is searched. .
	IRobotFRFResults ( <a href="#">see page 913</a> )	Set of results for FRF analysis.
	IRobotFootfallResults ( <a href="#">see page 915</a> )	Set of results for Footfall analysis.

### III.3.1 IRobotReactionData

**Class Hierarchy****C++**

```
interface IRobotReactionData : IDispatch;
```

**C#**

```
public interface IRobotReactionData;
```

**Visual Basic**

```
Public Interface IRobotReactionData
```

**Description**

The structure describing the value of reactions. .

**III.3.1.1 IRobotReactionData Members**

The following tables list the members exposed by IRobotReactionData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	FX ( <a href="#">see page 881</a> )	
◆	FY ( <a href="#">see page 881</a> )	
◆	FZ ( <a href="#">see page 881</a> )	
◆	MX ( <a href="#">see page 881</a> )	
◆	MY ( <a href="#">see page 881</a> )	
◆	MZ ( <a href="#">see page 881</a> )	

**III.3.1.2 IRobotReactionData Fields**

The fields of the IRobotReactionData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	FX ( <a href="#">see page 881</a> )	
◆	FY ( <a href="#">see page 881</a> )	
◆	FZ ( <a href="#">see page 881</a> )	
◆	MX ( <a href="#">see page 881</a> )	
◆	MY ( <a href="#">see page 881</a> )	
◆	MZ ( <a href="#">see page 881</a> )	

**III.3.1.2.1 FX****C++**

```
HRESULT get_FX(double*);
```

**C#**

```
public double FX { get; }
```

**Visual Basic**

```
Public ReadOnly FX As Double
```

**III.3.1.2.2 FY****C++**

```
HRESULT get_FY(double*);
```

**C#**

```
public double FY { get; }
```

**Visual Basic**

```
Public ReadOnly FY As double
```

**III.3.1.2.3 FZ****C++**

```
HRESULT get_FZ(double*);
```

**C#**

```
public double FZ { get; }
```

**Visual Basic**

```
Public ReadOnly FZ As double
```

**III.3.1.2.4 MX****C++**

```
HRESULT get_MX(double*);
```

**C#**

```
public double MX { get; }
```

**Visual Basic**

```
Public ReadOnly MX As double
```

**III.3.1.2.5 MY****C++**

```
HRESULT get_MY(double*);
```

**C#**

```
public double MY { get; }
```

**Visual Basic**

```
Public ReadOnly MY As double
```

**III.3.1.2.6 MZ****C++**

```
HRESULT get_MZ(double*);
```

**C#**

```
public double MZ { get; }
```

**Visual Basic**

```
Public ReadOnly MZ As double
```

**III.3.2 IRobotDisplacementData****Class Hierarchy****C++**

```
interface IRobotDisplacementData : IDispatch;
```

**C#**

```
public interface IRobotDisplacementData;
```

**Visual Basic**

```
Public Interface IRobotDisplacementData
```

**Description**

Structure describing the value of displacements. .

**III.3.2.1 IRobotDisplacementData Members**

The following tables list the members exposed by IRobotDisplacementData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	RX ( [ see page 882)	
◆	RY ( [ see page 883)	
◆	RZ ( [ see page 883)	
◆	UX ( [ see page 883)	
◆	UY ( [ see page 883)	
◆	UZ ( [ see page 883)	

**III.3.2.2 IRobotDisplacementData Fields**

The fields of the IRobotDisplacementData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	RX ( [ see page 882)	
◆	RY ( [ see page 883)	
◆	RZ ( [ see page 883)	
◆	UX ( [ see page 883)	
◆	UY ( [ see page 883)	
◆	UZ ( [ see page 883)	

**III.3.2.2.1 RX****C++**

```
HRESULT get_RX(double*);
```

**C#**

```
public double RX { get; }
```

**Visual Basic**

```
Public ReadOnly RX As double
```

**III.3.2.2.2 RY****C++**

```
HRESULT get_RY(double*);
```

**C#**

```
public double RY { get; }
```

**Visual Basic**

```
Public ReadOnly RY As double
```

### III.3.2.2.3 RZ

**C++**

```
HRESULT get_RZ(double*);
```

**C#**

```
public double RZ { get; }
```

**Visual Basic**

```
Public ReadOnly RZ As double
```

### III.3.2.2.4 UX

**C++**

```
HRESULT get_UX(double*);
```

**C#**

```
public double UX { get; }
```

**Visual Basic**

```
Public ReadOnly UX As double
```

### III.3.2.2.5 UY

**C++**

```
HRESULT get_UY(double*);
```

**C#**

```
public double UY { get; }
```

**Visual Basic**

```
Public ReadOnly UY As double
```

### III.3.2.2.6 UZ

**C++**

```
HRESULT get_UZ(double*);
```

**C#**

```
public double UZ { get; }
```

**Visual Basic**

```
Public ReadOnly UZ As double
```

## III.3.3 IRobotBarForceData

**Class Hierarchy**

**C++**

```
interface IRobotBarForceData : IDispatch;
```

**C#**

```
public interface IRobotBarForceData;
```

**Visual Basic**

```
Public Interface IRobotBarForceData
```

## Description

Structure describing values of forces, moments and elastic ground reaction for a bar. .

### III.3.3.1 IRobotBarForceData Members

The following tables list the members exposed by IRobotBarForceData.

#### Public Fields

	Name	Description
◆	FX ( <a href="#">see page 885</a> )	
◆	FY ( <a href="#">see page 885</a> )	
◆	FZ ( <a href="#">see page 885</a> )	
◆	KY ( <a href="#">see page 885</a> )	
◆	KYAvailable ( <a href="#">see page 885</a> )	Flag indicating availability of the value of elastic ground reaction KY ( <a href="#">see page 885</a> ).
◆	KZ ( <a href="#">see page 886</a> )	
◆	KZAvailable ( <a href="#">see page 886</a> )	Flag indicating availability of the value of elastic ground reaction KZ ( <a href="#">see page 886</a> ).
◆	MX ( <a href="#">see page 886</a> )	
◆	MY ( <a href="#">see page 886</a> )	
◆	MZ ( <a href="#">see page 887</a> )	

### III.3.3.2 IRobotBarForceData Fields

The fields of the IRobotBarForceData class are listed here.

#### Public Fields

	Name	Description
◆	FX ( <a href="#">see page 885</a> )	
◆	FY ( <a href="#">see page 885</a> )	
◆	FZ ( <a href="#">see page 885</a> )	
◆	KY ( <a href="#">see page 885</a> )	
◆	KYAvailable ( <a href="#">see page 885</a> )	Flag indicating availability of the value of elastic ground reaction KY ( <a href="#">see page 885</a> ).
◆	KZ ( <a href="#">see page 886</a> )	
◆	KZAvailable ( <a href="#">see page 886</a> )	Flag indicating availability of the value of elastic ground reaction KZ ( <a href="#">see page 886</a> ).
◆	MX ( <a href="#">see page 886</a> )	
◆	MY ( <a href="#">see page 886</a> )	
◆	MZ ( <a href="#">see page 887</a> )	

### III.3.3.2.1 FX

#### C++

```
HRESULT get_FX(double*);
```

#### C#

```
public double FX { get; }
```

#### Visual Basic

```
Public ReadOnly FX As double
```

### III.3.3.2.2 FY

#### C++

```
HRESULT get_FY(double*);
```

#### C#

```
public double FY { get; }
```

#### Visual Basic

```
Public ReadOnly FY As double
```

### III.3.3.2.3 FZ

#### C++

```
HRESULT get_FZ(double*);
```

#### C#

```
public double FZ { get; }
```

#### Visual Basic

```
Public ReadOnly FZ As double
```

### III.3.3.2.4 KY

#### C++

```
HRESULT get_KY(double*);
```

#### C#

```
public double KY { get; }
```

#### Visual Basic

```
Public ReadOnly KY As double
```

#### Version

Available since version 3.5.

### III.3.3.2.5 KYAvailable

#### C++

```
HRESULT get_KYAvailable(VARIANT_BOOL*);
```

#### C#

```
public bool KYAvailable { get; }
```

#### Visual Basic

```
Public ReadOnly KYAvailable As Boolean
```

#### Description

Flag indicating availability of the value of elastic ground reaction KY (see page 885).

#### Version

Available since version 3.5.

### III.3.3.2.6 KZ

#### C++

```
HRESULT get_KZ(double*);
```

**C#**

```
public double KZ { get; }
```

**Visual Basic**

```
Public ReadOnly KZ As Double
```

**Version**

Available since version 3.5.

### III.3.3.2.7 KZAvailable

**C++**

```
HRESULT get_KZAvailable(VARIANT_BOOL* );
```

**C#**

```
public bool KZAvailable { get; }
```

**Visual Basic**

```
Public ReadOnly KZAvailable As Boolean
```

**Description**

Flag indicating availability of the value of elastic ground reaction KZ (see page 886).

**Version**

Available since version 3.5.

### III.3.3.2.8 MX

**C++**

```
HRESULT get_MX(double* );
```

**C#**

```
public double MX { get; }
```

**Visual Basic**

```
Public ReadOnly MX As Double
```

### III.3.3.2.9 MY

**C++**

```
HRESULT get_MY(double* );
```

**C#**

```
public double MY { get; }
```

**Visual Basic**

```
Public ReadOnly MY As Double
```

### III.3.3.2.10 MZ

**C++**

```
HRESULT get_MZ(double* );
```

**C#**

```
public double MZ { get; }
```

**Visual Basic**

```
Public ReadOnly MZ As Double
```

### III.3.4 IRobotBarStressData

#### Class Hierarchy

#### C++

```
interface IRobotBarStressData : IDispatch;
```

#### C#

```
public interface IRobotBarStressData;
```

#### Visual Basic

```
Public Interface IRobotBarStressData
```

#### Description

A structure includes an information about stresses in a selected bar point for a selected load case.

#### III.3.4.1 IRobotBarStressData Members

The following tables list the members exposed by IRobotBarStressData.

#### Public Fields

	Name	Description
◆	FXSX (↗ see page 888)	
◆	ShearY (↗ see page 888)	
◆	ShearZ (↗ see page 888)	
◆	Smax (↗ see page 888)	
◆	SmaxMY (↗ see page 889)	
◆	SmaxMZ (↗ see page 889)	
◆	Smin (↗ see page 889)	
◆	SminMY (↗ see page 889)	
◆	SminMZ (↗ see page 889)	
◆	Torsion (↗ see page 889)	

#### III.3.4.2 IRobotBarStressData Fields

The fields of the IRobotBarStressData class are listed here.

#### Public Fields

	Name	Description
◆	FXSX (↗ see page 888)	
◆	ShearY (↗ see page 888)	
◆	ShearZ (↗ see page 888)	
◆	Smax (↗ see page 888)	
◆	SmaxMY (↗ see page 889)	
◆	SmaxMZ (↗ see page 889)	
◆	Smin (↗ see page 889)	
◆	SminMY (↗ see page 889)	
◆	SminMZ (↗ see page 889)	
◆	Torsion (↗ see page 889)	

#### III.3.4.2.1 FXSX

#### C++

```
HRESULT get_FXSX(double*);
```

**C#**

```
public double FXSX { get; }
```

**Visual Basic**

```
Public ReadOnly FXSX As Double
```

**III.3.4.2.2 ShearY****C++**

```
HRESULT get_ShearY(double*);
```

**C#**

```
public double Sheary { get; }
```

**Visual Basic**

```
Public ReadOnly Sheary As Double
```

**III.3.4.2.3 ShearZ****C++**

```
HRESULT get_ShearZ(double*);
```

**C#**

```
public double ShearZ { get; }
```

**Visual Basic**

```
Public ReadOnly ShearZ As Double
```

**III.3.4.2.4 Smax****C++**

```
HRESULT get_Smax(double*);
```

**C#**

```
public double Smax { get; }
```

**Visual Basic**

```
Public ReadOnly Smax As Double
```

**III.3.4.2.5 SmaxMY****C++**

```
HRESULT get_SmaxMY(double*);
```

**C#**

```
public double SmaxMY { get; }
```

**Visual Basic**

```
Public ReadOnly SmaxMY As Double
```

**III.3.4.2.6 SmaxMZ****C++**

```
HRESULT get_SmaxMZ(double*);
```

**C#**

```
public double SmaxMZ { get; }
```

**Visual Basic**

```
Public ReadOnly SmaxMZ As double
```

**III.3.4.2.7 Smin****C++**

```
HRESULT get_Smin(double*);
```

**C#**

```
public double Smin { get; }
```

**Visual Basic**

```
Public ReadOnly Smin As double
```

**III.3.4.2.8 SminMY****C++**

```
HRESULT get_SminMY(double*);
```

**C#**

```
public double SminMY { get; }
```

**Visual Basic**

```
Public ReadOnly SminMY As double
```

**III.3.4.2.9 SminMZ****C++**

```
HRESULT get_SminMZ(double*);
```

**C#**

```
public double SminMZ { get; }
```

**Visual Basic**

```
Public ReadOnly SminMZ As double
```

**III.3.4.2.10 Torsion****C++**

```
HRESULT get_Torsion(double*);
```

**C#**

```
public double Torsion { get; }
```

**Visual Basic**

```
Public ReadOnly Torsion As double
```

**III.3.5 IRobotBarDeflectionData****Class Hierarchy****C++**

```
interface IRobotBarDeflectionData : IDispatch;
```

**C#**

```
public interface IRobotBarDeflectionData;
```

## Visual Basic

```
Public Interface IRobotBarDeflectionData
```

### Description

A type of data describing deflections for the selected bar and case in a given bar point.

#### III.3.5.1 IRobotBarDeflectionData Members

The following tables list the members exposed by IRobotBarDeflectionData.

### Public Fields

	Name	Description
◆	PosUX ( [ see page 891 )	A relative point position on a bar for which UX ( [ see page 891 ) value was read .
◆	PosUY ( [ see page 891 )	A relative point position for which UY ( [ see page 891 ) value was read .
◆	PosUZ ( [ see page 891 )	A relative point position for which UZ ( [ see page 892 ) value was read .
◆	UX ( [ see page 891 )	
◆	UY ( [ see page 891 )	
◆	UZ ( [ see page 892 )	

#### III.3.5.2 IRobotBarDeflectionData Fields

The fields of the IRobotBarDeflectionData class are listed here.

### Public Fields

	Name	Description
◆	PosUX ( [ see page 891 )	A relative point position on a bar for which UX ( [ see page 891 ) value was read .
◆	PosUY ( [ see page 891 )	A relative point position for which UY ( [ see page 891 ) value was read .
◆	PosUZ ( [ see page 891 )	A relative point position for which UZ ( [ see page 892 ) value was read .
◆	UX ( [ see page 891 )	
◆	UY ( [ see page 891 )	
◆	UZ ( [ see page 892 )	

#### III.3.5.2.1 PosUX

### C++

```
HRESULT get_PosUX(double* );
```

### C#

```
public double PosUX { get; }
```

## Visual Basic

```
Public ReadOnly PosUX As Double
```

### Description

A relative point position on a bar for which UX ( [ see page 891 ) value was read .

#### III.3.5.2.2 PosUY

### C++

```
HRESULT get_PosUY(double* );
```

### C#

```
public double PosUY { get; }
```

**Visual Basic**

```
Public ReadOnly PosUY As double
```

**Description**

A relative point position for which UY (see page 891) value was read.

**III.3.5.2.3 PosUZ****C++**

```
HRESULT get_PosUZ(double*);
```

**C#**

```
public double PosUZ { get; }
```

**Visual Basic**

```
Public ReadOnly PosUZ As double
```

**Description**

A relative point position for which UZ (see page 892) value was read.

**III.3.5.2.4 UX****C++**

```
HRESULT get_UX(double*);
```

**C#**

```
public double UX { get; }
```

**Visual Basic**

```
Public ReadOnly UX As double
```

**III.3.5.2.5 UY****C++**

```
HRESULT get_UY(double*);
```

**C#**

```
public double UY { get; }
```

**Visual Basic**

```
Public ReadOnly UY As double
```

**III.3.5.2.6 UZ****C++**

```
HRESULT get_UZ(double*);
```

**C#**

```
public double UZ { get; }
```

**Visual Basic**

```
Public ReadOnly UZ As double
```

**III.3.6 IRobotBarBucklingData****Class Hierarchy**

**C++**

```
interface IRobotBarBucklingData : IDispatch;
```

**C#**

```
public interface IRobotBarBucklingData;
```

**Visual Basic**

```
Public Interface IRobotBarBucklingData
```

**Description**

The structure describing the eigenvalues for a single member, load case and mode. .

**III.3.6.1 IRobotBarBucklingData Members**

The following tables list the members exposed by IRobotBarBucklingData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BuckLengthY (see page 893)	Buckling length Y.
❖	BuckLengthZ (see page 893)	Buckling length Z.
❖	CriticalCoef (see page 893)	Critical coefficient .
❖	CriticalForce (see page 894)	Critical force.
❖	SlendY (see page 894)	Slenderness Y.
❖	SlendZ (see page 894)	Slenderness Z.

**III.3.6.2 IRobotBarBucklingData Fields**

The fields of the IRobotBarBucklingData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BuckLengthY (see page 893)	Buckling length Y.
❖	BuckLengthZ (see page 893)	Buckling length Z.
❖	CriticalCoef (see page 893)	Critical coefficient .
❖	CriticalForce (see page 894)	Critical force.
❖	SlendY (see page 894)	Slenderness Y.
❖	SlendZ (see page 894)	Slenderness Z.

**III.3.6.2.1 BuckLengthY****C++**

```
HRESULT get_BuckLengthY(double* );
```

**C#**

```
public double BuckLengthY { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthY As Double
```

**Description**

Buckling length Y.

### III.3.6.2.2 BuckLengthZ

**C++**

```
HRESULT get_BuckLengthZ(double*);
```

**C#**

```
public double BuckLengthZ { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthZ As double
```

**Description**

Buckling length Z.

### III.3.6.2.3 CriticalCoef

**C++**

```
HRESULT get_CriticalCoef(double*);
```

**C#**

```
public double CriticalCoef { get; }
```

**Visual Basic**

```
Public ReadOnly CriticalCoef As double
```

**Description**

Critical coefficient .

### III.3.6.2.4 CriticalForce

**C++**

```
HRESULT get_CriticalForce(double*);
```

**C#**

```
public double CriticalForce { get; }
```

**Visual Basic**

```
Public ReadOnly CriticalForce As double
```

**Description**

Critical force.

### III.3.6.2.5 SlendY

**C++**

```
HRESULT get_SlendY(double*);
```

**C#**

```
public double SlendY { get; }
```

**Visual Basic**

```
Public ReadOnly SlendY As double
```

**Description**

Slenderness Y.

### III.3.6.2.6 SlendZ

#### C++

```
HRESULT get_SlendZ(double*);
```

#### C#

```
public double SlendZ { get; }
```

#### Visual Basic

```
Public ReadOnly SlendZ As Double
```

#### Description

Slenderness Z.

## III.3.7 IRobotEigenvalues

#### Class Hierarchy

#### C++

```
interface IRobotEigenvalues : IDispatch;
```

#### C#

```
public interface IRobotEigenvalues;
```

#### Visual Basic

```
Public Interface IRobotEigenvalues
```

#### Description

Data structure containing eigenvalues..

### III.3.7.1 IRobotEigenvalues Members

The following tables list the members exposed by IRobotEigenvalues.

#### Public Fields

	Name	Description
❖	AvPartCoeff ( <a href="#">see page 895</a> )	Available since version 1.7.
❖	Damping ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	EigenValue ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	Energy ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	Frequence ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	Period ( <a href="#">see page 897</a> )	Available since version 1.7.
❖	Precision ( <a href="#">see page 897</a> )	Available since version 1.7.
❖	Pulsation ( <a href="#">see page 897</a> )	Available since version 1.7.

### III.3.7.2 IRobotEigenvalues Fields

The fields of the IRobotEigenvalues class are listed here.

#### Public Fields

	Name	Description
❖	AvPartCoeff ( <a href="#">see page 895</a> )	Available since version 1.7.
❖	Damping ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	EigenValue ( <a href="#">see page 896</a> )	Available since version 1.7.
❖	Energy ( <a href="#">see page 896</a> )	Available since version 1.7.

◆	Frequency (see page 896)	Available since version 1.7.
◆	Period (see page 897)	Available since version 1.7.
◆	Precision (see page 897)	Available since version 1.7.
◆	Pulsation (see page 897)	Available since version 1.7.

### III.3.7.2.1 AvPartCoeff

**C++**

```
HRESULT get_AvPartCoeff(double*);
```

**C#**

```
public double AvPartCoeff { get; }
```

**Visual Basic**

```
Public ReadOnly AvPartCoeff As double
```

**Description**

Available since version 1.7.

### III.3.7.2.2 Damping

**C++**

```
HRESULT get_Damping(double*);
```

**C#**

```
public double Damping { get; }
```

**Visual Basic**

```
Public ReadOnly Damping As double
```

**Description**

Available since version 1.7.

### III.3.7.2.3 EigenValue

**C++**

```
HRESULT get_EigenValue(double*);
```

**C#**

```
public double EigenValue { get; }
```

**Visual Basic**

```
Public ReadOnly EigenValue As double
```

**Description**

Available since version 1.7.

### III.3.7.2.4 Energy

**C++**

```
HRESULT get_Energy(double*);
```

**C#**

```
public double Energy { get; }
```

**Visual Basic**

```
Public ReadOnly Energy As double
```

**Description**

Available since version 1.7.

**III.3.7.2.5 Frequence****C++**

```
HRESULT get_Frequence(double*);
```

**C#**

```
public double Frequence { get; }
```

**Visual Basic**

```
Public ReadOnly Frequence As double
```

**Description**

Available since version 1.7.

**III.3.7.2.6 Period****C++**

```
HRESULT get_Period(double*);
```

**C#**

```
public double Period { get; }
```

**Visual Basic**

```
Public ReadOnly Period As double
```

**Description**

Available since version 1.7.

**III.3.7.2.7 Precision****C++**

```
HRESULT get_Precision(double*);
```

**C#**

```
public double Precision { get; }
```

**Visual Basic**

```
Public ReadOnly Precision As double
```

**Description**

Available since version 1.7.

**III.3.7.2.8 Pulsation****C++**

```
HRESULT get_Pulsation(double*);
```

**C#**

```
public double Pulsation { get; }
```

**Visual Basic**

```
Public ReadOnly Pulsation As double
```

**Description**

Available since version 1.7.

**III.3.8 IRobotForcesData****Class Hierarchy****C++**

```
interface IRobotForcesData : IDispatch;
```

**C#**

```
public interface IRobotForcesData;
```

**Visual Basic**

```
Public Interface IRobotForcesData
```

**Description**

Data structure containing force and moment values.

**III.3.8.1 IRobotForcesData Members**

The following tables list the members exposed by IRobotForcesData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	FX ( [ see page 898 )	Available since version 1.7.
◆	FY ( [ see page 898 )	Available since version 1.7.
◆	FZ ( [ see page 899 )	Available since version 1.7.
◆	MX ( [ see page 899 )	Available since version 1.7.
◆	MY ( [ see page 899 )	Available since version 1.7.
◆	MZ ( [ see page 899 )	Available since version 1.7.

**III.3.8.2 IRobotForcesData Fields**

The fields of the IRobotForcesData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	FX ( [ see page 898 )	Available since version 1.7.
◆	FY ( [ see page 898 )	Available since version 1.7.
◆	FZ ( [ see page 899 )	Available since version 1.7.
◆	MX ( [ see page 899 )	Available since version 1.7.
◆	MY ( [ see page 899 )	Available since version 1.7.
◆	MZ ( [ see page 899 )	Available since version 1.7.

**III.3.8.2.1 FX****C++**

```
HRESULT get_FX(double* );
```

**C#**

```
public double FX { get; }
```

**Visual Basic**

```
Public ReadOnly FX As Double
```

**Description**

Available since version 1.7.

### III.3.8.2.2 FY

**C++**

```
HRESULT get_FY(double*);
```

**C#**

```
public double FY { get; }
```

**Visual Basic**

```
Public ReadOnly FY As Double
```

**Description**

Available since version 1.7.

### III.3.8.2.3 FZ

**C++**

```
HRESULT get_FZ(double*);
```

**C#**

```
public double FZ { get; }
```

**Visual Basic**

```
Public ReadOnly FZ As Double
```

**Description**

Available since version 1.7.

### III.3.8.2.4 MX

**C++**

```
HRESULT get_MX(double*);
```

**C#**

```
public double MX { get; }
```

**Visual Basic**

```
Public ReadOnly MX As Double
```

**Description**

Available since version 1.7.

### III.3.8.2.5 MY

**C++**

```
HRESULT get_MY(double*);
```

**C#**

```
public double MY { get; }
```

**Visual Basic**

```
Public ReadOnly MY As Double
```

**Description**

Available since version 1.7.

**III.3.8.2.6 MZ****C++**

```
HRESULT get_MZ(double*);
```

**C#**

```
public double MZ { get; }
```

**Visual Basic**

```
Public ReadOnly MZ As Double
```

**Description**

Available since version 1.7.

**III.3.9 IRobotTimeHistoryResults****Class Hierarchy****C++**

```
interface IRobotTimeHistoryResults : IDispatch;
```

**C#**

```
public interface IRobotTimeHistoryResults;
```

**Visual Basic**

```
Public Interface IRobotTimeHistoryResults
```

**Description**

Results for time history analysis.

**Version**

Available since version 3.

**III.3.9.1 IRobotTimeHistoryResults Members**

The following tables list the members exposed by IRobotTimeHistoryResults.

**Public Fields**

	Name	Description
◆	ARX (↗ see page 901)	Value of acceleration component ARX.
◆	ARY (↗ see page 901)	Value of acceleration component ARY.
◆	ARZ (↗ see page 901)	Value of acceleration component ARZ.
◆	AX (↗ see page 902)	Value of acceleration component AX.
◆	AY (↗ see page 902)	Value of acceleration component AY.
◆	AZ (↗ see page 902)	Value of acceleration component AZ.
◆	Time (↗ see page 903)	Time corresponding to the specified step.

◆	VRX (see page 903)	Value of velocity component VRX.
◆	VRY (see page 903)	Value of velocity component VRY.
◆	VRZ (see page 903)	Value of velocity component VRZ.
◆	VX (see page 904)	Value of velocity component VX.
◆	VY (see page 904)	Value of velocity component VY.
◆	VZ (see page 904)	Value of velocity component VZ.

### III.3.9.2 IRobotTimeHistoryResults Fields

The fields of the IRobotTimeHistoryResults class are listed here.

#### Public Fields

	Name	Description
◆	ARX (see page 901)	Value of acceleration component ARX.
◆	ARY (see page 901)	Value of acceleration component ARY.
◆	ARZ (see page 901)	Value of acceleration component ARZ.
◆	AX (see page 902)	Value of acceleration component AX.
◆	AY (see page 902)	Value of acceleration component AY.
◆	AZ (see page 902)	Value of acceleration component AZ.
◆	Time (see page 903)	Time corresponding to the specified step.
◆	VRX (see page 903)	Value of velocity component VRX.
◆	VRY (see page 903)	Value of velocity component VRY.
◆	VRZ (see page 903)	Value of velocity component VRZ.
◆	VX (see page 904)	Value of velocity component VX.
◆	VY (see page 904)	Value of velocity component VY.
◆	VZ (see page 904)	Value of velocity component VZ.

#### III.3.9.2.1 ARX

##### C++

```
HRESULT get_ARX( double* );
```

##### C#

```
public double ARX { get; }
```

##### Visual Basic

```
Public ReadOnly ARX As Double
```

##### Description

Value of acceleration component ARX.

##### Version

Available since version 3.

#### III.3.9.2.2 ARY

##### C++

```
HRESULT get_ARY( double* );
```

##### C#

```
public double ARY { get; }
```

##### Visual Basic

```
Public ReadOnly ARY As Double
```

**Description**

Value of acceleration component ARY.

**Version**

Available since version 3.

**III.3.9.2.3 ARZ****C++**

```
HRESULT get_ARZ( double* );
```

**C#**

```
public double ARZ { get; }
```

**Visual Basic**

```
Public ReadOnly ARZ As double
```

**Description**

Value of acceleration component ARZ.

**Version**

Available since version 3.

**III.3.9.2.4 AX****C++**

```
HRESULT get_AX( double* );
```

**C#**

```
public double AX { get; }
```

**Visual Basic**

```
Public ReadOnly AX As double
```

**Description**

Value of acceleration component AX.

**Version**

Available since version 3.

**III.3.9.2.5 AY****C++**

```
HRESULT get_AY( double* );
```

**C#**

```
public double AY { get; }
```

**Visual Basic**

```
Public ReadOnly AY As double
```

**Description**

Value of acceleration component AY.

**Version**

Available since version 3.

### III.3.9.2.6 AZ

**C++**

```
HRESULT get_AZ( double* );
```

**C#**

```
public double AZ { get; }
```

**Visual Basic**

```
Public ReadOnly AZ As Double
```

**Description**

Value of acceleration component AZ.

**Version**

Available since version 3.

### III.3.9.2.7 Time

**C++**

```
HRESULT get_Time( double* );
```

**C#**

```
public double Time { get; }
```

**Visual Basic**

```
Public ReadOnly Time As Double
```

**Description**

Time corresponding to the specified step.

**Version**

Available since version 3.

### III.3.9.2.8 VRX

**C++**

```
HRESULT get_VRX( double* );
```

**C#**

```
public double VRX { get; }
```

**Visual Basic**

```
Public ReadOnly VRX As Double
```

**Description**

Value of velocity component VRX.

**Version**

Available since version 3.

### III.3.9.2.9 VRY

**C++**

```
HRESULT get_VRY( double* );
```

**C#**

```
public double VRY { get; }
```

**Visual Basic**

```
Public ReadOnly VRY As Double
```

**Description**

Value of velocity component VRY.

**Version**

Available since version 3.

**III.3.9.2.10 VRZ****C++**

```
HRESULT get_VRZ( Double* );
```

**C#**

```
public double VRZ { get; }
```

**Visual Basic**

```
Public ReadOnly VRZ As Double
```

**Description**

Value of velocity component VRZ.

**Version**

Available since version 3.

**III.3.9.2.11 VX****C++**

```
HRESULT get_VX( Double* );
```

**C#**

```
public double VX { get; }
```

**Visual Basic**

```
Public ReadOnly VX As Double
```

**Description**

Value of velocity component VX.

**Version**

Available since version 3.

**III.3.9.2.12 VY****C++**

```
HRESULT get_VY( Double* );
```

**C#**

```
public double VY { get; }
```

**Visual Basic**

```
Public ReadOnly VY As Double
```

**Description**

Value of velocity component VY.

**Version**

Available since version 3.

**III.3.9.2.13 VZ****C++**

```
HRESULT get_VZ( double* );
```

**C#**

```
public double VZ { get; }
```

**Visual Basic**

```
Public ReadOnly VZ As Double
```

**Description**

Value of velocity component VZ.

**Version**

Available since version 3.

**III.3.10 IRobotExtremeValue****Class Hierarchy****C++**

```
interface IRobotExtremeValue : IDispatch;
```

**C#**

```
public interface IRobotExtremeValue;
```

**Visual Basic**

```
Public Interface IRobotExtremeValue
```

**Description**

Structure providing access to information about the extreme value. .

**Version**

Available since version 3.5.

**III.3.10.1 IRobotExtremeValue Members**

The following tables list the members exposed by IRobotExtremeValue.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Bar (see page 906)	Number of the bar for which the extreme value has been obtained .
◆	Case (see page 906)	Number of the load case for which the specified extreme value has been obtained .

◆	CaseCmpnt (see page 906)	Additional number indicating a load case component <ul style="list-style-type: none"> <li>• for a code combination, it is a number of its component</li> <li>• for modal analysis, it is a mode number</li> <li>• for results of time history analysis, it is a step number.</li> </ul>
◆	IsAvailable (see page 907)	Flag indicating if the extreme value has been taken correctly from the result server .
◆	ModeCmb (see page 907)	Designation of the mode combination for which the specified extreme value has been obtained (significant only when CaseCmpnt (see page 906) component equals zero) .
◆	Node (see page 907)	Number of the node in which the extreme value has been obtained .
◆	Position (see page 908)	Relative position on a bar in which the extreme value has been obtained .
◆	Value (see page 908)	Value.

### III.3.10.2 IRobotExtremeValue Fields

The fields of the IRobotExtremeValue class are listed here.

#### Public Fields

	Name	Description
◆	Bar (see page 906)	Number of the bar for which the extreme value has been obtained .
◆	Case (see page 906)	Number of the load case for which the specified extreme value has been obtained .
◆	CaseCmpnt (see page 906)	Additional number indicating a load case component <ul style="list-style-type: none"> <li>• for a code combination, it is a number of its component</li> <li>• for modal analysis, it is a mode number</li> <li>• for results of time history analysis, it is a step number.</li> </ul>
◆	IsAvailable (see page 907)	Flag indicating if the extreme value has been taken correctly from the result server .
◆	ModeCmb (see page 907)	Designation of the mode combination for which the specified extreme value has been obtained (significant only when CaseCmpnt (see page 906) component equals zero) .
◆	Node (see page 907)	Number of the node in which the extreme value has been obtained .
◆	Position (see page 908)	Relative position on a bar in which the extreme value has been obtained .
◆	Value (see page 908)	Value.

### III.3.10.2.1 Bar

#### C++

```
HRESULT get_Bar(long*);
```

#### C#

```
public long Bar { get; }
```

#### Visual Basic

```
Public ReadOnly Bar As long
```

#### Description

Number of the bar for which the extreme value has been obtained .

#### Version

Available since version 3.5.

### III.3.10.2.2 Case

#### C++

```
HRESULT get_Case(long*);
```

#### C#

```
public long Case { get; }
```

#### Visual Basic

```
Public ReadOnly Case As long
```

#### Description

Number of the load case for which the specified extreme value has been obtained .

#### Version

Available since version 3.5.

### III.3.10.2.3 CaseCmpnt

#### C++

```
HRESULT get_CaseCmpnt(long*);
```

#### C#

```
public long CaseCmpnt { get; }
```

#### Visual Basic

```
Public ReadOnly CaseCmpnt As long
```

#### Description

Additional number indicating a load case component

- for a code combination, it is a number of its component
- for modal analysis, it is a mode number
- for results of time history analysis, it is a step number.

#### Version

Available since version 3.5.

### III.3.10.2.4 IsAvailable

#### C++

```
HRESULT get_IsAvailable(VARIANT_BOOL*);
```

#### C#

```
public bool IsAvailable { get; }
```

#### Visual Basic

```
Public ReadOnly IsAvailable As Boolean
```

#### Description

Flag indicating if the extreme value has been taken correctly from the result server .

#### Version

Available since version 3.5.

### III.3.10.2.5 ModeCmb

#### C++

```
HRESULT get_ModeCmb( IRobotModeCombinationType* );
```

#### C#

```
public IRobotModeCombinationType ModeCmb { get; }
```

#### Visual Basic

```
Public ReadOnly ModeCmb As IRobotModeCombinationType
```

#### Description

Designation of the mode combination for which the specified extreme value has been obtained (significant only when CaseCmpnt (see page 906) component equals zero).

#### Version

Available since version 3.5.

### III.3.10.2.6 Node

#### C++

```
HRESULT get_Node( long* );
```

#### C#

```
public long Node { get; }
```

#### Visual Basic

```
Public ReadOnly Node As long
```

#### Description

Number of the node in which the extreme value has been obtained.

#### Version

Available since version 3.5.

### III.3.10.2.7 Position

#### C++

```
HRESULT get_Position( double* );
```

#### C#

```
public double Position { get; }
```

#### Visual Basic

```
Public ReadOnly Position As double
```

#### Description

Relative position on a bar in which the extreme value has been obtained.

#### Version

Available since version 3.5.

### III.3.10.2.8 Value

#### C++

```
HRESULT get_Value( double* );
```

**C#**

```
public double Value { get; }
```

**Visual Basic**

```
Public ReadOnly Value As Double
```

**Description**

Value.

**Version**

Available since version 3.5.

### III.3.11 IRobotExtremeParams

**Class Hierarchy****C++**

```
interface IRobotExtremeParams : IDispatch;
```

**C#**

```
public interface IRobotExtremeParams;
```

**Visual Basic**

```
Public Interface IRobotExtremeParams
```

**Description**

Structure containing parameters for which the extreme value is searched. .

**Version**

Available since version 3.5.

#### III.3.11.1 IRobotExtremeParams Members

The following tables list the members exposed by IRobotExtremeParams.

**Public Fields**

	Name	Description
◆	BarDivision (see page 909)	Number of bar divisions - significant only for bar results.
◆	Selection (see page 909)	Selection of structure components (nodes, bars, load cases) that should be considered while searching the extreme value.
◆	ValueType (see page 910)	Type of the result for which the extreme value is to be taken .

#### III.3.11.2 IRobotExtremeParams Fields

The fields of the IRobotExtremeParams class are listed here.

**Public Fields**

	Name	Description
◆	BarDivision (see page 909)	Number of bar divisions - significant only for bar results.
◆	Selection (see page 909)	Selection of structure components (nodes, bars, load cases) that should be considered while searching the extreme value.
◆	ValueType (see page 910)	Type of the result for which the extreme value is to be taken .

##### III.3.11.2.1 BarDivision

**C++**

```
HRESULT get_BarDivision(long*);
```

```
HRESULT put_BarDivision(long);

C#
public long BarDivision { get; set; }
```

**Visual Basic**

```
Public BarDivision As long
```

**Description**

Number of bar divisions - significant only for bar results.

**Version**

Available since version 3.5.

**III.3.11.2.2 Selection****C++**

```
HRESULT get_Selection(IRobotMultiSelection**);
```

**C#**

```
public IRobotMultiSelection Selection { get; }
```

**Visual Basic**

```
Public ReadOnly Selection As IRobotMultiSelection
```

**Description**

Selection of structure components (nodes, bars, load cases) that should be considered while searching the extreme value.

**Version**

Available since version 3.5.

**III.3.11.2.3 ValueType****C++**

```
HRESULT get_ValueType(IRobotExtremeValueType* );
HRESULT put_ValueType(IRobotExtremeValueType);
```

**C#**

```
public IRobotExtremeValueType ValueType { get; set; }
```

**Visual Basic**

```
Public ValueType As IRobotExtremeValueType
```

**Description**

Type of the result for which the extreme value is to be taken .

**Version**

Available since version 3.5.

**III.3.12 IRobotExtremeValueType****C++**

```
enum IRobotExtremeValueType;
```

**C#**

```
public enum IRobotExtremeValueType;
```

## Visual Basic

```
Public Enum IRobotExtremeValueType
```

### Members

Members	Description
I_EVTREACTION_FX = 109	Available since version 3.5.
I_EVTREACTION_FY = 110	Available since version 3.5.
I_EVTREACTION_FZ = 111	Available since version 3.5.
I_EVTREACTION_MX = 112	Available since version 3.5.
I_EVTREACTION_MY = 113	Available since version 3.5.
I_EVTREACTION_MZ = 114	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_UX = 141	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_UY = 142	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_UZ = 143	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_RX = 144	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_RY = 145	Available since version 3.5.
I_EVTDISPLACEMENT_NODE_RZ = 146	Available since version 3.5.
I_EVTDEFLECTION_UX = 153	Available since version 3.5.
I_EVTDEFLECTION_UY = 154	Available since version 3.5.
I_EVTDEFLECTION_UZ = 155	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_UX = 147	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_UY = 148	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_UZ = 149	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_RX = 150	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_RY = 151	Available since version 3.5.
I_EVTDISPLACEMENT_BAR_RZ = 152	Available since version 3.5.
I_EVTFORCE_BAR_FX = 165	Available since version 3.5.
I_EVTFORCE_BAR_FY = 166	Available since version 3.5.
I_EVTFORCE_BAR_FZ = 167	Available since version 3.5.
I_EVTFORCE_BAR_MX = 168	Available since version 3.5.
I_EVTFORCE_BAR_MY = 169	Available since version 3.5.
I_EVTFORCE_BAR_MZ = 170	Available since version 3.5.
I_EVTSTRESS_BAR_SMAX = 171	Available since version 3.5.
I_EVTSTRESS_BAR_SMIN = 172	Available since version 3.5.
I_EVTSTRESS_BAR_SMAX_MY = 174	Available since version 3.5.
I_EVTSTRESS_BAR_SMAX_MZ = 175	Available since version 3.5.
I_EVTSTRESS_BAR_SMIN_MY = 176	Available since version 3.5.
I_EVTSTRESS_BAR_SMIN_MZ = 177	Available since version 3.5.
I_EVTSTRESS_BAR_FX_SX = 173	Available since version 3.5.
I_EVTSTRESS_BAR_TY = 179	Available since version 3.5.
I_EVTSTRESS_BAR_TZ = 180	Available since version 3.5.
I_EVTSTRESS_BAR_T = 182	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_FX = 251	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_FY = 252	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_FZ = 253	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_MX = 254	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_MY = 255	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_MZ = 256	Available since version 3.5.
I_EVTPSEUDOSTATIC_FORCE_GX = 1328	Available since version 3.5.

I_EVT_PSEUDOSTATIC_FORCE_GY = 1329	Available since version 3.5.
I_EVT_PSEUDOSTATIC_FORCE_GZ = 1330	Available since version 3.5.
I_EVT_PSEUDOSTATIC_FORCE_TX = 1331	Available since version 3.5.
I_EVT_PSEUDOSTATIC_FORCE_TY = 1332	Available since version 3.5.
I_EVT_PSEUDOSTATIC_FORCE_TZ = 1333	Available since version 3.5.
I_EVT_TIME_VX = 257	Available since version 3.5.
I_EVT_TIME_VY = 258	Available since version 3.5.
I_EVT_TIME_VZ = 259	Available since version 3.5.
I_EVT_TIME_VRX = 260	Available since version 3.5.
I_EVT_TIME_VRY = 261	Available since version 3.5.
I_EVT_TIME_VRZ = 262	Available since version 3.5.
I_EVT_TIME_AX = 263	Available since version 3.5.
I_EVT_TIME_AY = 264	Available since version 3.5.
I_EVT_TIME_AZ = 265	Available since version 3.5.
I_EVT_TIME_ARX = 266	Available since version 3.5.
I_EVT_TIME_ARY = 267	Available since version 3.5.
I_EVT_TIME_ARZ = 268	Available since version 3.5.
I_EVT_FRF_VX = 1783	VX velocity for FRF analysis. Available since version 9.7.
I_EVT_FRF_VY = 1784	VY velocity for FRF analysis. Available since version 9.7.
I_EVT_FRF_VZ = 1785	VZ velocity for FRF analysis. Available since version 9.7.
I_EVT_FOOTFALL_A = 1854	Overall maximum acceleration. Available since version 9.7.
I_EVT_FOOTFALL_RF_RESONANT = 1855	Maximum (resonant) response factor. Available since version 9.7.
I_EVT_FOOTFALL_RF_TRANSIENT = 1856	Maximum (transient) response factor. Available since version 9.7.
I_EVT_FOOTFALL_RF_OVERALL = 1857	Maximum (overall) response factor. Available since version 9.7.
I_EVT_FOOTFALL_VRMS = 1858	Available since version 9.7.
I_EVT_FOOTFALL_VRMQ = 1859	Available since version 9.7.
I_EVT_FOOTFALL_FREQUENCY = 1862	Available since version 9.7.
I_EVT_FOOTFALL_EXCITATION_NODE = 1861	Available since version 9.7.
I_EVT_MODAL_FREQUENCY = 404	Available since version 11.
I_EVT_MODAL_PERIOD = 405	Available since version 11.
I_EVT_MODAL_PULSATION = 406	Available since version 11.
I_EVT_MODAL_CURMASS_UX = 184	Available since version 11.
I_EVT_MODAL_CURMASS_UY = 185	Available since version 11.
I_EVT_MODAL_CURMASS_UZ = 186	Available since version 11.
I_EVT_MODAL_CURMASS_RX = 187	Available since version 11.
I_EVT_MODAL_CURMASS_RY = 188	Available since version 11.
I_EVT_MODAL_CURMASS_RZ = 189	Available since version 11.
I_EVT_MODAL_SUMMASS_UX = 190	Available since version 11.
I_EVT_MODAL_SUMMASS_UY = 191	Available since version 11.
I_EVT_MODAL_SUMMASS_UZ = 192	Available since version 11.
I_EVT_MODAL_SUMMASS_RX = 193	Available since version 11.
I_EVT_MODAL_SUMMASS_RY = 194	Available since version 11.
I_EVT_MODAL_SUMMASS_RZ = 195	Available since version 11.

I_EVT_MODAL_TOTMASS_UX = 196	Available since version 11.
I_EVT_MODAL_TOTMASS_UY = 197	Available since version 11.
I_EVT_MODAL_TOTMASS_UZ = 198	Available since version 11.
I_EVT_MODAL_TOTMASS_RX = 199	Available since version 11.
I_EVT_MODAL_TOTMASS_RY = 200	Available since version 11.
I_EVT_MODAL_TOTMASS_RZ = 201	Available since version 11.
I_EVT_MODAL_PRECISION = 209	Available since version 11.
I_EVT_MODAL_DAMPING = 984	Available since version 11.
I_EVT_MODAL_ENERGY = 985	Available since version 11.
I_EVT_MODAL_AVERAGE_PARTICIPATION_COEFF = 986	Available since version 11.
I_EVT_MODAL_EIGENVALUE = 208	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UX = 228	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UY = 229	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UZ = 230	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RX = 231	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RY = 232	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RZ = 233	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UX_1 = 234	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UY_1 = 235	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_UZ_1 = 236	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RX_1 = 237	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RY_1 = 238	Available since version 11.
I_EVT_MODAL_EIGENVECTOR_RZ_1 = 239	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_UX = 202	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_UY = 203	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_UZ = 204	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_RX = 205	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_RY = 206	Available since version 11.
I_EVT_MODAL_PARTICIPATION_COEFF_RZ = 207	Available since version 11.
I_EVT_MODAL_MASSES_X = 1767	Available since version 11.
I_EVT_MODAL_MASSES_Y = 1768	Available since version 11.
I_EVT_MODAL_MASSES_Z = 1769	Available since version 11.
I_EVT_MODAL_NODE_MASSES_X = 1770	Available since version 11.
I_EVT_MODAL_NODE_MASSES_Y = 1771	Available since version 11.
I_EVT_MODAL_NODE_MASSES_Z = 1772	Available since version 11.

**Description**

Set of results for which the extreme values are made accessible. .

**Version**

Available since version 3.5.

**III.3.13 IRobotFRFResults****Class Hierarchy****C++**

```
interface IRobotFRFResults : IDispatch;
```

**C#**

```
public interface IRobotFRFResults;
```

**Visual Basic**

```
Public Interface IRobotFRFResults
```

**Description**

Set of results for FRF analysis.

**Version**

Available since version 9.7.

**III.3.13.1 IRobotFRFResults Members**

The following tables list the members exposed by IRobotFRFResults.

**Public Fields**

	Name	Description
◆	Frequency ( <a href="#">see page 914</a> )	A frequency value that corresponds to the specified step .
◆	VX ( <a href="#">see page 914</a> )	Value of VX velocity.
◆	VY ( <a href="#">see page 914</a> )	Value of VY velocity.
◆	VZ ( <a href="#">see page 915</a> )	Value of VZ velocity.

**III.3.13.2 IRobotFRFResults Fields**

The fields of the IRobotFRFResults class are listed here.

**Public Fields**

	Name	Description
◆	Frequency ( <a href="#">see page 914</a> )	A frequency value that corresponds to the specified step .
◆	VX ( <a href="#">see page 914</a> )	Value of VX velocity.
◆	VY ( <a href="#">see page 914</a> )	Value of VY velocity.
◆	VZ ( <a href="#">see page 915</a> )	Value of VZ velocity.

**III.3.13.2.1 Frequency****C++**

```
HRESULT get_Frequency(double*);
```

**C#**

```
public double Frequency { get; }
```

**Visual Basic**

```
Public ReadOnly Frequency As double
```

**Description**

A frequency value that corresponds to the specified step .

**Version**

Available since version 9.7.

**III.3.13.2.2 VX****C++**

```
HRESULT get_VX(double*);
```

**C#**

```
public double VX { get; }
```

**Visual Basic**

```
Public ReadOnly VX As double
```

**Description**

Value of VX velocity.

**Version**

Available since version 9.7.

**III.3.13.2.3 VY****C++**

```
HRESULT get_VY(double*);
```

**C#**

```
public double VY { get; }
```

**Visual Basic**

```
Public ReadOnly VY As double
```

**Description**

Value of VY velocity.

**Version**

Available since version 9.7.

**III.3.13.2.4 VZ****C++**

```
HRESULT get_VZ(double*);
```

**C#**

```
public double VZ { get; }
```

**Visual Basic**

```
Public ReadOnly VZ As double
```

**Description**

Value of VZ velocity.

**Version**

Available since version 9.7.

**III.3.14 IRobotFootfallResults****Class Hierarchy****C++**

```
interface IRobotFootfallResults : IDispatch;
```

**C#**

```
public interface IRobotFootfallResults;
```

**Visual Basic**

```
Public Interface IRobotFootfallResults
```

## Description

Set of results for Footfall analysis.

## Version

Available since version 9.7.

### III.3.14.1 IRobotFootfallResults Members

The following tables list the members exposed by IRobotFootfallResults.

#### Public Fields

	Name	Description
◆	A (see page 916)	Overall maximum acceleration.
◆	ExcitationNode (see page 916)	Excitation node.
◆	Frequency (see page 917)	Frequency.
◆	RF_Overall (see page 917)	Maximum (overall) response factor.
◆	RF_Resonant (see page 917)	Maximum (resonant) response factor.
◆	RF_Transient (see page 918)	Maximum (transient) response factor.
◆	VRMQ (see page 918)	VRMQ (Root Mean Quad Velocity).
◆	VRMS (see page 918)	VRMS (Root Mean Square Velocity).

### III.3.14.2 IRobotFootfallResults Fields

The fields of the IRobotFootfallResults class are listed here.

#### Public Fields

	Name	Description
◆	A (see page 916)	Overall maximum acceleration.
◆	ExcitationNode (see page 916)	Excitation node.
◆	Frequency (see page 917)	Frequency.
◆	RF_Overall (see page 917)	Maximum (overall) response factor.
◆	RF_Resonant (see page 917)	Maximum (resonant) response factor.
◆	RF_Transient (see page 918)	Maximum (transient) response factor.
◆	VRMQ (see page 918)	VRMQ (Root Mean Quad Velocity).
◆	VRMS (see page 918)	VRMS (Root Mean Square Velocity).

### III.3.14.2.1 A

#### C++

```
HRESULT get_A(double*);
```

#### C#

```
public double A { get; }
```

#### Visual Basic

```
Public ReadOnly A As Double
```

#### Description

Overall maximum acceleration.

#### Version

Available since version 9.7.

### III.3.14.2.2 ExcitationNode

#### C++

```
HRESULT get_ExcitationNode(long*);
```

#### C#

```
public long ExcitationNode { get; }
```

#### Visual Basic

```
Public ReadOnly ExcitationNode As long
```

#### Description

Excitation node.

#### Version

Available since version 9.7.

### III.3.14.2.3 Frequency

#### C++

```
HRESULT get_Frequency(double*);
```

#### C#

```
public double Frequency { get; }
```

#### Visual Basic

```
Public ReadOnly Frequency As double
```

#### Description

Frequency.

#### Version

Available since version 9.7.

### III.3.14.2.4 RF\_Overall

#### C++

```
HRESULT get_RF_Overall(double*);
```

#### C#

```
public double RF_Overall { get; }
```

#### Visual Basic

```
Public ReadOnly RF_Overall As double
```

#### Description

Maximum (overall) response factor.

#### Version

Available since version 9.7.

### III.3.14.2.5 RF\_Resonant

#### C++

```
HRESULT get_RF_Resonant(double*);
```

**C#**

```
public double RF_Resonant { get; }
```

**Visual Basic**

```
Public ReadOnly RF_Resonant As double
```

**Description**

Maximum (resonant) response factor.

**Version**

Available since version 9.7.

### III.3.14.2.6 RF\_Transient

**C++**

```
HRESULT get_RF_Transient(double*);
```

**C#**

```
public double RF_Transient { get; }
```

**Visual Basic**

```
Public ReadOnly RF_Transient As double
```

**Description**

Maximum (transient) response factor.

**Version**

Available since version 9.7.

### III.3.14.2.7 VRMQ

**C++**

```
HRESULT get_VRMQ(double*);
```

**C#**

```
public double VRMQ { get; }
```

**Visual Basic**

```
Public ReadOnly VRMQ As double
```

**Description**

VRMQ (Root Mean Quad Velocity).

**Version**

Available since version 9.7.

### III.3.14.2.8 VRMS

**C++**

```
HRESULT get_VRMS(double*);
```

**C#**

```
public double VRMS { get; }
```

**Visual Basic**

```
Public ReadOnly VRMS As double
```

**Description**

VRMS (Root Mean Square Velocity).

**Version**

Available since version 9.7.

## III.4 Calculation results for dynamic analyses

**Interfaces**

	<b>Name</b>	<b>Description</b>
↳	IRobotAdvancedResultServer (↗ see page 919)	
↳	IRobotEigenvaluesServer (↗ see page 922)	Server providing access to eigenvalues.
↳	IRobotMassSumServer (↗ see page 923)	
↳	IRobotSpectralCoefficients (↗ see page 925)	Server provides access to spectral coefficient values. .
↳	IRobotEigenvectorsServer (↗ see page 927)	Server providing access to eigenvectors. .
↳	IRobotTimeHistoryResultServer (↗ see page 928)	Result server for time history analysis.
↳	IRobotFRFResultServer (↗ see page 929)	Result server for FRF analysis.

### III.4.1 IRobotAdvancedResultServer

**Class Hierarchy****C++**

```
interface IRobotAdvancedResultServer : IDispatch;
```

**C#**

```
public interface IRobotAdvancedResultServer;
```

**Visual Basic**

```
Public Interface IRobotAdvancedResultServer
```

#### III.4.1.1 IRobotAdvancedResultServer Members

The following tables list the members exposed by IRobotAdvancedResultServer.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Eigenvalues (↗ see page 920)	Available since version 1.7.
❖	Eigenvectors (↗ see page 920)	Server providing access to eigenvector values Available since version 1.7.
❖	FRF (↗ see page 920)	Result server for FRF analysis.
❖	MassSum (↗ see page 921)	Server providing access to sum of masses Available since version 1.7.
❖	SpectralCoeffs (↗ see page 921)	Server providing access to spectral coefficients Available since version 1.7.
❖	TimeHistory (↗ see page 921)	Server providing access to results of time history analysis.

## Public Methods

	Name	Description
FootfallValue (see page 922)		Function returns a set of Footfall analysis results for the specified node and load case.

### III.4.1.2 IRobotAdvancedResultServer Fields

The fields of the IRobotAdvancedResultServer class are listed here.

#### Public Fields

	Name	Description
Eigenvalues (see page 920)		Available since version 1.7.
Eigenvectors (see page 920)		Server providing access to eigenvector values Available since version 1.7.
FRF (see page 920)		Result server for FRF analysis.
MassSum (see page 921)		Server providing access to sum of masses Available since version 1.7.
SpectralCoeffs (see page 921)		Server providing access to spectral coefficients Available since version 1.7.
TimeHistory (see page 921)		Server providing access to results of time history analysis.

#### III.4.1.2.1 Eigenvalues

##### C++

```
HRESULT get_Eigenvalues(IRobotEigenvaluesServer**);
```

##### C#

```
public IRobotEigenvaluesServer Eigenvalues { get; }
```

##### Visual Basic

```
Public ReadOnly Eigenvalues As IRobotEigenvaluesServer
```

##### Description

Available since version 1.7.

#### III.4.1.2.2 Eigenvectors

##### C++

```
HRESULT get_Eigenvectors(IRobotEigenvectorsServer**);
```

##### C#

```
public IRobotEigenvectorsServer Eigenvectors { get; }
```

##### Visual Basic

```
Public ReadOnly Eigenvectors As IRobotEigenvectorsServer
```

##### Description

Server providing access to eigenvector values Available since version 1.7.

#### III.4.1.2.3 FRF

##### C++

```
HRESULT get_FRF(IRobotFRFResultServer**);
```

##### C#

```
public IRobotFRFResultServer FRF { get; }
```

**Visual Basic**

```
Public ReadOnly FRF As IRobotFRFResultServer
```

**Description**

Result server for FRF analysis.

**Version**

Available since version 9.7.

### III.4.1.2.4 MassSum

**C++**

```
HRESULT get_MassSum(IRobotMassSumServer**);
```

**C#**

```
public IRobotMassSumServer MassSum { get; }
```

**Visual Basic**

```
Public ReadOnly MassSum As IRobotMassSumServer
```

**Description**

Server providing access to sum of masses Available since version 1.7.

### III.4.1.2.5 SpectralCoeffs

**C++**

```
HRESULT get_SpectralCoeffs(IRobotSpectralCoefficients**);
```

**C#**

```
public IRobotSpectralCoefficients SpectralCoeffs { get; }
```

**Visual Basic**

```
Public ReadOnly SpectralCoeffs As IRobotSpectralCoefficients
```

**Description**

Server providing access to spectral coefficients Available since version 1.7.

### III.4.1.2.6 TimeHistory

**C++**

```
HRESULT get_TimeHistory(IRobotTimeHistoryResultServer**);
```

**C#**

```
public IRobotTimeHistoryResultServer TimeHistory { get; }
```

**Visual Basic**

```
Public ReadOnly TimeHistory As IRobotTimeHistoryResultServer
```

**Description**

Server providing access to results of time history analysis.

**Version**

Available since version 3.

### III.4.1.3 IRobotAdvancedResultServer Methods

The methods of the IRobotAdvancedResultServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
FootfallValue (see page 922)		Function returns a set of Footfall analysis results for the specified node and load case.

**III.4.1.3.1 FootfallValue****C++**

```
HRESULT FootfallValue(long _node, long _case, IRobotFootfallResults** ret);
```

**C#**

```
public IRobotFootfallResults FootfallValue(long _node, long _case);
```

**Visual Basic**

```
Public Function FootfallValue(_node As long, _case As long) As IRobotFootfallResults
```

**Description**

Function returns a set of Footfall analysis results for the specified node and load case.

**Version**

Available since version 9.7.

**III.4.2 IRobotEigenvaluesServer****Class Hierarchy****C++**

```
interface IRobotEigenvaluesServer : IDispatch;
```

**C#**

```
public interface IRobotEigenvaluesServer;
```

**Visual Basic**

```
Public Interface IRobotEigenvaluesServer
```

**Description**

Server providing access to eigenvalues.

**III.4.2.1 IRobotEigenvaluesServer Members**

The following tables list the members exposed by IRobotEigenvaluesServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
CombValue (see page 923)		Available since version 1.7.
Value (see page 923)		Available since version 1.7.

**III.4.2.2 IRobotEigenvaluesServer Methods**

The methods of the IRobotEigenvaluesServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
CombValue (see page 923)		Available since version 1.7.
Value (see page 923)		Available since version 1.7.

### III.4.2.2.1 CombValue

**C++**

```
HRESULT CombValue(long _case_num, IRobotModeCombinationType _mode_cmb, IRobotEigenvalues** ret);
```

**C#**

```
public IRobotEigenvalues CombValue(long _case_num, IRobotModeCombinationType _mode_cmb);
```

**Visual Basic**

```
Public Function CombValue(_case_num As long, _mode_cmb As IRobotModeCombinationType) As IRobotEigenvalues
```

**Description**

Available since version 1.7.

### III.4.2.2.2 Value

**C++**

```
HRESULT Value(long _case_num, long _mode_num, IRobotEigenvalues** ret);
```

**C#**

```
public IRobotEigenvalues Value(long _case_num, long _mode_num);
```

**Visual Basic**

```
Public Function Value(_case_num As long, _mode_num As long) As IRobotEigenvalues
```

**Description**

Available since version 1.7.

## III.4.3 IRobotMassSumServer

**Class Hierarchy**

**C++**

```
interface IRobotMassSumServer : IDispatch;
```

**C#**

```
public interface IRobotMassSumServer;
```

**Visual Basic**

```
Public Interface IRobotMassSumServer
```

### III.4.3.1 IRobotMassSumServer Members

The following tables list the members exposed by IRobotMassSumServer.

**Public Methods**

	Name	Description
✳️	Current (see page 924)	Function returns values of participation masses expressed in percentages in the current mode for a specified case and mode, for individual directions. Available since version 1.7.
✳️	PartCoeff (see page 924)	Function returns values of modal participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.

	Relative (see page 925)	Function returns values of participation mass sum expressed in percentages from mode 1 to the current mode for a specified case and mode, for individual directions. Available since version 1.7.
	Total (see page 925)	Function returns values of total participation mass sum in a specified case for a specified mode, for individual directions. Available since version 1.7.

### III.4.3.2 IRobotMassSumServer Methods

The methods of the IRobotMassSumServer class are listed here.

#### Public Methods

	Name	Description
	Current (see page 924)	Function returns values of participation masses expressed in percentages in the current mode for a specified case and mode, for individual directions. Available since version 1.7.
	PartCoeff (see page 924)	Function returns values of modal participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.
	Relative (see page 925)	Function returns values of participation mass sum expressed in percentages from mode 1 to the current mode for a specified case and mode, for individual directions. Available since version 1.7.
	Total (see page 925)	Function returns values of total participation mass sum in a specified case for a specified mode, for individual directions. Available since version 1.7.

#### III.4.3.2.1 Current

##### C++

```
HRESULT Current(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData Current(long _case_num, long _mode_num);
```

##### Visual Basic

```
Public Function Current(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

##### Description

Function returns values of participation masses expressed in percentages in the current mode for a specified case and mode, for individual directions. Available since version 1.7.

#### III.4.3.2.2 PartCoeff

##### C++

```
HRESULT PartCoeff(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData PartCoeff(long _case_num, long _mode_num);
```

##### Visual Basic

```
Public Function PartCoeff(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

##### Description

Function returns values of modal participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.

### III.4.3.2.3 Relative

**C++**

```
HRESULT Relative(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData Relative(long _case_num, long _mode_num);
```

**Visual Basic**

```
Public Function Relative(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

**Description**

Function returns values of participation mass sum expressed in percentages from mode 1 to the current mode for a specified case and mode, for individual directions. Available since version 1.7.

### III.4.3.2.4 Total

**C++**

```
HRESULT Total(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData Total(long _case_num, long _mode_num);
```

**Visual Basic**

```
Public Function Total(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

**Description**

Function returns values of total participation mass sum in a specified case for a specified mode, for individual directions. Available since version 1.7.

## III.4.4 IRobotSpectralCoefficients

**Class Hierarchy**

**C++**

```
interface IRobotSpectralCoefficients : IDispatch;
```

**C#**

```
public interface IRobotSpectralCoefficients;
```

**Visual Basic**

```
Public Interface IRobotSpectralCoefficients
```

**Description**

Server provides access to spectral coefficient values..

### III.4.4.1 IRobotSpectralCoefficients Members

The following tables list the members exposed by IRobotSpectralCoefficients.

**Public Methods**

	Name	Description
!	ModeCoef (see page 926)	Function returns values of spectral mode coefficients in a specified case and mode for individual directions. Available since version 1.7.

	PartCoef (see page 926)	Function returns values of spectral participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.
	SpectrCoef (see page 927)	Function returns spectral coefficient value for a specified case and mode. Available since version 1.7.

### III.4.4.2 IRobotSpectralCoefficients Methods

The methods of the IRobotSpectralCoefficients class are listed here.

#### Public Methods

	Name	Description
	ModeCoef (see page 926)	Function returns values of spectral mode coefficients in a specified case and mode for individual directions. Available since version 1.7.
	PartCoef (see page 926)	Function returns values of spectral participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.
	SpectrCoef (see page 927)	Function returns spectral coefficient value for a specified case and mode. Available since version 1.7.

#### III.4.4.2.1 ModeCoef

##### C++

```
HRESULT ModeCoef(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData ModeCoef(long _case_num, long _mode_num);
```

##### Visual Basic

```
Public Function ModeCoef(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

##### Description

Function returns values of spectral mode coefficients in a specified case and mode for individual directions. Available since version 1.7.

#### III.4.4.2.2 PartCoef

##### C++

```
HRESULT PartCoef(long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

##### C#

```
public IRobotDisplacementData PartCoef(long _case_num, long _mode_num);
```

##### Visual Basic

```
Public Function PartCoef(_case_num As long, _mode_num As long) As IRobotDisplacementData
```

##### Description

Function returns values of spectral participation coefficients in a specified case for a defined mode, for individual directions. Available since version 1.7.

#### III.4.4.2.3 SpectrCoef

##### C++

```
HRESULT SpectrCoef(long _case_num, long _mode_num, double* ret);
```

**C#**

```
public double SpectrCoef(long _case_num, long _mode_num);
```

**Visual Basic**

```
Public Function SpectrCoef(_case_num As Long, _mode_num As Long) As Double
```

**Description**

Function returns spectral coefficient value for a specified case and mode. Available since version 1.7.

**III.4.5 IRobotEigenvectorsServer****Class Hierarchy****C++**

```
interface IRobotEigenvectorsServer : IDispatch;
```

**C#**

```
public interface IRobotEigenvectorsServer;
```

**Visual Basic**

```
Public Interface IRobotEigenvectorsServer
```

**Description**

Server providing access to eigenvectors. .

**III.4.5.1 IRobotEigenvectorsServer Members**

The following tables list the members exposed by IRobotEigenvectorsServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	CombValue (see page 928)	Function returns eigenvector for a specified node, load case and mode combination. Available since version 1.7.
💡	Value (see page 928)	Function returns eigenvector for a specified node, load case and mode. Available since version 1.7.

**III.4.5.2 IRobotEigenvectorsServer Methods**

The methods of the IRobotEigenvectorsServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	CombValue (see page 928)	Function returns eigenvector for a specified node, load case and mode combination. Available since version 1.7.
💡	Value (see page 928)	Function returns eigenvector for a specified node, load case and mode. Available since version 1.7.

**III.4.5.2.1 CombValue****C++**

```
HRESULT CombValue(long _node_num, long _case_num, IRobotModeCombinationType _mode_cmb,
IRobotDisplacementData** ret);
```

**C#**

```
public IRobotDisplacementData CombValue(long _node_num, long _case_num,
IRobotModeCombinationType _mode_cmb);
```

## Visual Basic

```
Public Function CombValue(_node_num As long, _case_num As long, _mode_cmb As
IRobotModeCombinationType) As IRobotDisplacementData
```

### Description

Function returns eigenvector for a specified node, load case and mode combination. Available since version 1.7.

## III.4.5.2.2 Value

### C++

```
HRESULT Value(long _node_num, long _case_num, long _mode_num, IRobotDisplacementData** ret);
```

### C#

```
public IRobotDisplacementData Value(long _node_num, long _case_num, long _mode_num);
```

## Visual Basic

```
Public Function Value(_node_num As long, _case_num As long, _mode_num As long) As
IRobotDisplacementData
```

### Description

Function returns eigenvector for a specified node, load case and mode. Available since version 1.7.

## III.4.6 IRobotTimeHistoryResultServer

### Class Hierarchy

### C++

```
interface IRobotTimeHistoryResultServer : IDispatch;
```

### C#

```
public interface IRobotTimeHistoryResultServer;
```

## Visual Basic

```
Public Interface IRobotTimeHistoryResultServer
```

### Description

Result server for time history analysis.

### Version

Available since version 3.

## III.4.6.1 IRobotTimeHistoryResultServer Members

The following tables list the members exposed by IRobotTimeHistoryResultServer.

### Public Methods

	Name	Description
	Value (see page 929)	Function returns a set of time history analysis results for a given node, case and step.

## III.4.6.2 IRobotTimeHistoryResultServer Methods

The methods of the IRobotTimeHistoryResultServer class are listed here.

## Public Methods

	Name	Description
⌚	Value (⌚ see page 929)	Function returns a set of time history analysis results for a given node, case and step.

### III.4.6.2.1 Value

#### C++

```
HRESULT Value(long _node, long _case, long _step, IRobotTimeHistoryResults** ret);
```

#### C#

```
public IRobotTimeHistoryResults Value(long _node, long _case, long _step);
```

#### Visual Basic

```
Public Function Value(_node As long, _case As long, _step As long) As  
IRobotTimeHistoryResults
```

#### Description

Function returns a set of time history analysis results for a given node, case and step.

#### Version

Available since version 3.

## III.4.7 IRobotFRFResultServer

### Class Hierarchy

#### C++

```
interface IRobotFRFResultServer : IDispatch;
```

#### C#

```
public interface IRobotFRFResultServer;
```

#### Visual Basic

```
Public Interface IRobotFRFResultServer
```

#### Description

Result server for FRF analysis.

#### Version

Available since version 9.7.

### III.4.7.1 IRobotFRFResultServer Members

The following tables list the members exposed by IRobotFRFResultServer.

## Public Methods

	Name	Description
⌚	Value (⌚ see page 930)	Function returns a set of FRF analysis results for the specified node, load case and step.

### III.4.7.2 IRobotFRFResultServer Methods

The methods of the IRobotFRFResultServer class are listed here.

## Public Methods

	Name	Description
	Value (see page 930)	Function returns a set of FRF analysis results for the specified node, load case and step.

### III.4.7.2.1 Value

#### C++

```
HRESULT Value(long _node, long _case, long _step, IRobotFRFResults** ret);
```

#### C#

```
public IRobotFRFResults Value(long _node, long _case, long _step);
```

#### Visual Basic

```
Public Function Value(_node As long, _case As long, _step As long) As IRobotFRFResults
```

#### Description

Function returns a set of FRF analysis results for the specified node, load case and step.

#### Version

Available since version 9.7.

## III.5 Calculation results for finite elements

Available since version 2.5.

### Enumerations

	Name	Description
	IRobotFeLayerType (see page 932)	Available designations of layers for which calculation results for planar finite elements may be obtained. .
	IRobotReinforceCalcMethods (see page 951)	Types of the reinforcement calculation hypotheses.
	IRobotFeResultReducedCutPosition (see page 972)	Available positions of a panel cut.
	IRobotPanelCutDefinitionType (see page 978)	Types of panel cut definition. .
	IRobotFeResultType (see page 979)	Designation of the type of result calculated for finite elements.
	IRobotFeResultSmoothing (see page 980)	Available methods of averaging (smoothing) results.
	IRobotFeResultKind (see page 981)	

## Interfaces

	Name	Description
↪	IRobotFeResultParams (see page 932)	Structure defining parameters necessary to take calculation results for finite elements. The type of returned values is strictly determined by the setting configuration of the fields: Panel (see page 936), Node (see page 936), Element (see page 935). The following variants are possible: <ol style="list-style-type: none"> <li>1. the Node (see page 936) node number is set globally averaged results at a node are returned</li> <li>2. the Node (see page 936) node number and the Panel (see page 936) panel number are set results at a node averaged within a given panel are the results returned</li> <li>3. the Node (see page 936) node number and the Element (see page 935) finite element number are set exact results at a node are returned</li> <li>4. the Element (see page 935) finite element number is... more (see page 932)</li> </ol>
↪	IRobotFeResultDetailed (see page 938)	Detailed result vector.
↪	IRobotFeResultServer (see page 945)	Result server for planar finite elements.
↪	IRobotFeResultPrincipal (see page 951)	Principal result vector.
↪	IRobotFeResultComplex (see page 958)	Complex result vector.
↪	IRobotFeResultReinforcement (see page 961)	Result vector for reinforcement.
↪	IRobotFeResultReduced (see page 966)	Reduced results for panels.
↪	IRobotPanelCutMngr (see page 972)	Manager of panel cuts. .
↪	IRobotPanelCut (see page 976)	Interface defining a cut through panel. .
↪	IRobotFeExtremeParams (see page 981)	Parameters of searching extreme results for finite elements. The list of nodes to be considered while searching extreme values is determined based on the settings in the fields NodeSel (see page 985), ElementSel (see page 984) and PanelSel (see page 985). If a list of nodes is set in the NodeSel (see page 985) field, then this list exactly will be considered in searching extreme values. If the NodeSel (see page 985) field is empty and the ElementSel (see page 984) field is set, then all the nodes belonging to the indicated finite elements will be considered. If the fields NodeSel (see page 985) and ElementSel (see page 984) are empty and the PanelSel (see page 985) field is set, then all the nodes belonging... more (see page 981)
↪	IRobotFeExtremeValue (see page 988)	Interface providing access to information about the extreme value. .
↪	IRobotFeMultiResultType (see page 993)	Interface which enables indicating simultaneously - a great number of result types for finite elements.
↪	IRobotFeMultiExtremeValue (see page 995)	Interface providing access to a great number of min./max. results for finite elements.

### III.5.1 IRobotFeLayerType

C++

```
enum IRobotFeLayerType;
```

**C#**

```
public enum IRobotFeLayerType;
```

**Visual Basic**

```
Public Enum IRobotFeLayerType
```

**Members**

Members	Description
I_FLT_UPPER = 1	Top layer. Available since version 2.5.
I_FLT_MIDDLE = 2	Middle layer. Available since version 2.5.
I_FLT_LOWER = 3	Bottom layer. Available since version 2.5.
I_FLT_ARBITRARY = 4	Relative layer width defined arbitrarily. Available since version 2.5.
I_FLT_MAXIMUM = 5	.
I_FLT_MINIMUM = 6	.
I_FLT_ABSOLUTE_MAXIMUM = 7	.

**Description**

Available designations of layers for which calculation results for planar finite elements may be obtained. .

**Version**

Available since version 2.5.

### III.5.2 IRobotFeResultParams

**Class Hierarchy****C++**

```
interface IRobotFeResultParams : IDispatch;
```

**C#**

```
public interface IRobotFeResultParams;
```

**Visual Basic**

```
Public Interface IRobotFeResultParams
```

**Description**

Structure defining parameters necessary to take calculation results for finite elements. The type of returned values is strictly determined by the setting configuration of the fields: Panel ( see page 936), Node ( see page 936), Element ( see page 935). The following variants are possible:

1. the Node ( see page 936) node number is set  
globally averaged results at a node are returned
2. the Node ( see page 936) node number and the Panel ( see page 936) panel number are set  
results at a node averaged within a given panel are the results returned
3. the Node ( see page 936) node number and the Element ( see page 935) finite element number are set

exact results at a node are returned  
 4. the Element (see page 935) finite element number is set  
 results at a center of the indicated element are the results returned

## Version

Available since version 2.5.

### III.5.2.1 IRobotFeResultParams Members

The following tables list the members exposed by IRobotFeResultParams.

#### Public Fields

	Name	Description
◆	CalcMethod (see page 934)	Type of reinforcement calculation method.
◆	Case (see page 934)	Load case number.
◆	CaseCmpnt (see page 934)	Component number for a complex component case or mode number for a dynamic case.
◆	Element (see page 935)	Finite element number.
◆	Layer (see page 935)	Layer.
◆	LayerArbitraryValue (see page 935)	Layer (see page 935) thickness.
◆	ModeCmb (see page 936)	Type of mode combination for a dynamic case .
◆	Node (see page 936)	Node number.
◆	Panel (see page 936)	Panel number.

#### Public Methods

	Name	Description
◆	GetDirX (see page 937)	Function takes the direction of the local X axis of the coordinate system in which the results will be presented. The values read are the coordinates of the direction vector of the X axis for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. In case when values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns a type of the system direction definition.. .
◆	SetDirX (see page 937)	The function enables defining the X axis direction of the system in which the results will be presented. Activation of the function should take place only when the vector of detailed results is being taken. Function parameters determine the coordinates of the direction vector of the X axis in the Cartesian system or the coordinates of the central point in the polar system. In case when all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined.. .

### III.5.2.2 IRobotFeResultParams Fields

The fields of the IRobotFeResultParams class are listed here.

#### Public Fields

	Name	Description
◆	CalcMethod (see page 934)	Type of reinforcement calculation method.
◆	Case (see page 934)	Load case number.
◆	CaseCmpnt (see page 934)	Component number for a complex component case or mode number for a dynamic case.
◆	Element (see page 935)	Finite element number.
◆	Layer (see page 935)	Layer.

◆	LayerArbitraryValue (see page 935)	Layer (see page 935) thickness.
◆	ModeCmb (see page 936)	Type of mode combination for a dynamic case .
◆	Node (see page 936)	Node number.
◆	Panel (see page 936)	Panel number.

### III.5.2.2.1 CalcMethod

#### C++

```
HRESULT get_CalcMethod(IRobotReinforceCalcMethods* );
HRESULT put_CalcMethod(IRobotReinforceCalcMethods);
```

#### C#

```
public IRobotReinforceCalcMethods CalcMethod { get; set; }
```

#### Visual Basic

```
Public CalcMethod As IRobotReinforceCalcMethods
```

#### Description

Type of reinforcement calculation method.

#### Version

Available since version 2.5.

### III.5.2.2.2 Case

#### C++

```
HRESULT get_Case( long* );
HRESULT put_Case( long );
```

#### C#

```
public long Case { get; set; }
```

#### Visual Basic

```
Public Case As long
```

#### Description

Load case number.

#### Version

Available since version 2.5.

### III.5.2.2.3 CaseCmpnt

#### C++

```
HRESULT get_CaseCmpnt( long* );
HRESULT put_CaseCmpnt( long );
```

#### C#

```
public long CaseCmpnt { get; set; }
```

#### Visual Basic

```
Public CaseCmpnt As long
```

#### Description

Component number for a complex component case or mode number for a dynamic case.

**Version**

Available since version 2.5.

**III.5.2.2.4 Element****C++**

```
HRESULT get_Element( long* );
HRESULT put_Element( long );
```

**C#**

```
public long Element { get; set; }
```

**Visual Basic**

```
Public Element As Long
```

**Description**

Finite element number.

**Version**

Available since version 2.5.

**III.5.2.2.5 Layer****C++**

```
HRESULT get_Layer( IRobotFeLayerType* );
HRESULT put_Layer( IRobotFeLayerType );
```

**C#**

```
public IRobotFeLayerType Layer { get; set; }
```

**Visual Basic**

```
Public Layer As IRobotFeLayerType
```

**Description**

Layer.

**Version**

Available since version 2.5.

**III.5.2.2.6 LayerArbitraryValue****C++**

```
HRESULT get_LayerArbitraryValue( double* );
HRESULT put_LayerArbitraryValue( double );
```

**C#**

```
public double LayerArbitraryValue { get; set; }
```

**Visual Basic**

```
Public LayerArbitraryValue As Double
```

**Description**

Layer (see page 935) thickness.

**Version**

Available since version 2.5.

### III.5.2.2.7 ModeCmb

#### C++

```
HRESULT get_ModeCmb(IRobotModeCombinationType* );
HRESULT put_ModeCmb(IRobotModeCombinationType);
```

#### C#

```
public IRobotModeCombinationType ModeCmb { get; set; }
```

#### Visual Basic

```
Public ModeCmb As IRobotModeCombinationType
```

#### Description

Type of mode combination for a dynamic case .

#### Version

Available since version 2.5.

### III.5.2.2.8 Node

#### C++

```
HRESULT get_Node( long* );
HRESULT put_Node( long );
```

#### C#

```
public long Node { get; set; }
```

#### Visual Basic

```
Public Node As long
```

#### Description

Node number.

#### Version

Available since version 2.5.

### III.5.2.2.9 Panel

#### C++

```
HRESULT get_Panel( long* );
HRESULT put_Panel( long );
```

#### C#

```
public long Panel { get; set; }
```

#### Visual Basic

```
Public Panel As long
```

#### Description

Panel number.

#### Version

Available since version 2.5.

### III.5.2.3 IRobotFeResultParams Methods

The methods of the IRobotFeResultParams class are listed here.

## Public Methods

	Name	Description
◆	GetDirX (see page 937)	Function takes the direction of the local X axis of the coordinate system in which the results will be presented. The values read are the coordinates of the direction vector of the X axis for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. In case when values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns a type of the system direction definition.. .
◆	SetDirX (see page 937)	The function enables defining the X axis direction of the system in which the results will be presented. Activation of the function should take place only when the vector of detailed results is being taken. Function parameters determine the coordinates of the direction vector of the X axis in the Cartesian system or the coordinates of the central point in the polar system. In case when all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined. .

### III.5.2.3.1 GetDirX

#### C++

```
HRESULT GetDirX(double* _x, double* _y, double* _z, IRobotObjLocalXDirDefinitionType* ret);
```

#### C#

```
public IRobotObjLocalXDirDefinitionType GetDirX(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetDirX(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As IRobotObjLocalXDirDefinitionType
```

#### Description

Function takes the direction of the local X axis of the coordinate system in which the results will be presented. The values read are the coordinates of the direction vector of the X axis for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. In case when values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns a type of the system direction definition. .

#### Version

Available since version 2.5.

### III.5.2.3.2 SetDirX

#### C++

```
HRESULT SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

#### C#

```
public void SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetDirX(_def_type As IRobotObjLocalXDirDefinitionType, _x As double, _y As double, _z As double)
```

#### Description

The function enables defining the X axis direction of the system in which the results will be presented. Activation of the function should take place only when the vector of detailed results is being taken. Function parameters determine the

coordinates of the direction vector of the X axis in the Cartesian system or the coordinates of the central point in the polar system. In case when all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined. .

## Version

Available since version 2.5.

### III.5.3 IRobotFeResultDetailed

#### Class Hierarchy

#### C++

```
interface IRobotFeResultDetailed : IDispatch;
```

#### C#

```
public interface IRobotFeResultDetailed;
```

#### Visual Basic

```
Public Interface IRobotFeResultDetailed
```

#### Description

Detailed result vector.

## Version

Available since version 2.5.

### III.5.3.1 IRobotFeResultDetailed Members

The following tables list the members exposed by IRobotFeResultDetailed.

#### Public Fields

	Name	Description
❖	MXX ( <a href="#">see page 939</a> )	MXX moment value.
❖	MXY ( <a href="#">see page 940</a> )	MXY moment value .
❖	MYY ( <a href="#">see page 940</a> )	MYY moment value.
❖	NXX ( <a href="#">see page 940</a> )	NXX membrane force value.
❖	NXY ( <a href="#">see page 940</a> )	NXY membrane force value .
❖	NYY ( <a href="#">see page 941</a> )	NYY membrane force value .
❖	PNorm ( <a href="#">see page 941</a> )	PNorm soil reaction value.
❖	QXX ( <a href="#">see page 941</a> )	QXX shear force value.
❖	QYY ( <a href="#">see page 942</a> )	QYY shear force value.
❖	RNorm ( <a href="#">see page 942</a> )	RNorm rotation.
❖	RXX ( <a href="#">see page 942</a> )	RXX rotation.
❖	RYY ( <a href="#">see page 943</a> )	RYY rotation.
❖	SXX ( <a href="#">see page 943</a> )	SXX stress value.
❖	SXY ( <a href="#">see page 943</a> )	SXY stress value.
❖	SYY ( <a href="#">see page 943</a> )	SYY stress value.
❖	TXX ( <a href="#">see page 944</a> )	TXX shear stress value.
❖	TYY ( <a href="#">see page 944</a> )	TYY shear stress value.
❖	UXX ( <a href="#">see page 944</a> )	UXX displacement.
❖	UYY ( <a href="#">see page 945</a> )	UYY displacement .
❖	WNorm ( <a href="#">see page 945</a> )	WNorm displacement .

### III.5.3.2 IRobotFeResultDetailed Fields

The fields of the IRobotFeResultDetailed class are listed here.

#### Public Fields

	Name	Description
◆	MXX ( <a href="#">see page 939</a> )	MXX moment value.
◆	MXY ( <a href="#">see page 940</a> )	MXY moment value .
◆	MYY ( <a href="#">see page 940</a> )	MYY moment value.
◆	NXX ( <a href="#">see page 940</a> )	NXX membrane force value.
◆	NXY ( <a href="#">see page 940</a> )	NXY membrane force value .
◆	NYY ( <a href="#">see page 941</a> )	NYY membrane force value .
◆	PNorm ( <a href="#">see page 941</a> )	PNorm soil reaction value.
◆	QXX ( <a href="#">see page 941</a> )	QXX shear force value.
◆	QYY ( <a href="#">see page 942</a> )	QYY shear force value.
◆	RNorm ( <a href="#">see page 942</a> )	RNorm rotation.
◆	RXX ( <a href="#">see page 942</a> )	RXX rotation.
◆	RYY ( <a href="#">see page 943</a> )	RYY rotation.
◆	SXX ( <a href="#">see page 943</a> )	SXX stress value.
◆	SXY ( <a href="#">see page 943</a> )	SXY stress value.
◆	SYY ( <a href="#">see page 943</a> )	SYY stress value.
◆	TXX ( <a href="#">see page 944</a> )	TXX shear stress value.
◆	TYY ( <a href="#">see page 944</a> )	TYY shear stress value.
◆	UXX ( <a href="#">see page 944</a> )	UXX displacement.
◆	UYY ( <a href="#">see page 945</a> )	UYY displacement .
◆	WNorm ( <a href="#">see page 945</a> )	WNorm displacement .

### III.5.3.2.1 MXX

#### C++

```
HRESULT get_MXX( double* );
```

#### C#

```
public double MXX { get; }
```

#### Visual Basic

```
Public ReadOnly MXX As Double
```

#### Description

MXX moment value.

#### Version

Available since version 2.5.

### III.5.3.2.2 MXY

#### C++

```
HRESULT get_MXY( double* );
```

#### C#

```
public double MXY { get; }
```

#### Visual Basic

```
Public ReadOnly MXY As Double
```

**Description**

MXY moment value .

**Version**

Available since version 2.5.

**III.5.3.2.3 MYY****C++**

```
HRESULT get_MYY( double* );
```

**C#**

```
public double MYY { get; }
```

**Visual Basic**

```
Public ReadOnly MYY As double
```

**Description**

MYY moment value.

**Version**

Available since version 2.5.

**III.5.3.2.4 NXX****C++**

```
HRESULT get_NXX( double* );
```

**C#**

```
public double NXX { get; }
```

**Visual Basic**

```
Public ReadOnly NXX As double
```

**Description**

NXX membrane force value.

**Version**

Available since version 2.5.

**III.5.3.2.5 NXY****C++**

```
HRESULT get_NXY( double* );
```

**C#**

```
public double NXY { get; }
```

**Visual Basic**

```
Public ReadOnly NXY As double
```

**Description**

NXY membrane force value .

**Version**

Available since version 2.5.

### III.5.3.2.6 NYY

**C++**

```
HRESULT get_NYY( double* );
```

**C#**

```
public double NYY { get; }
```

**Visual Basic**

```
Public ReadOnly NYY As double
```

**Description**

NYY membrane force value.

**Version**

Available since version 2.5.

### III.5.3.2.7 PNorm

**C++**

```
HRESULT get_PNorm( double* );
```

**C#**

```
public double PNorm { get; }
```

**Visual Basic**

```
Public ReadOnly PNorm As double
```

**Description**

PNorm soil reaction value.

**Version**

Available since version 2.5.

### III.5.3.2.8 QXX

**C++**

```
HRESULT get_QXX( double* );
```

**C#**

```
public double QXX { get; }
```

**Visual Basic**

```
Public ReadOnly QXX As double
```

**Description**

QXX shear force value.

**Version**

Available since version 2.5.

### III.5.3.2.9 QYY

**C++**

```
HRESULT get_QYY( double* );
```

**C#**

```
public double QYY { get; }
```

**Visual Basic**

```
Public ReadOnly QYY As Double
```

**Description**

QYY shear force value.

**Version**

Available since version 2.5.

### III.5.3.2.10 RNorm

**C++**

```
HRESULT get_RNorm( Double* );
```

**C#**

```
public double RNorm { get; }
```

**Visual Basic**

```
Public ReadOnly RNorm As Double
```

**Description**

RNorm rotation.

**Version**

Available since version 2.5.

### III.5.3.2.11 RXX

**C++**

```
HRESULT get_RXX( Double* );
```

**C#**

```
public double RXX { get; }
```

**Visual Basic**

```
Public ReadOnly RXX As Double
```

**Description**

RXX rotation.

**Version**

Available since version 2.5.

### III.5.3.2.12 RYY

**C++**

```
HRESULT get_RYY( Double* );
```

**C#**

```
public double RYY { get; }
```

**Visual Basic**

```
Public ReadOnly RYY As Double
```

**Description**

RYY rotation.

**Version**

Available since version 2.5.

**III.5.3.2.13 SXX****C++**

```
HRESULT get_SXX( double* );
```

**C#**

```
public double SXX { get; }
```

**Visual Basic**

```
Public ReadOnly SXX As double
```

**Description**

SXX stress value.

**Version**

Available since version 2.5.

**III.5.3.2.14 SXY****C++**

```
HRESULT get_SXY( double* );
```

**C#**

```
public double SXY { get; }
```

**Visual Basic**

```
Public ReadOnly SXY As double
```

**Description**

SXY stress value.

**Version**

Available since version 2.5.

**III.5.3.2.15 SYY****C++**

```
HRESULT get_SYY( double* );
```

**C#**

```
public double SYY { get; }
```

**Visual Basic**

```
Public ReadOnly SYY As double
```

**Description**

SYY stress value.

**Version**

Available since version 2.5.

### III.5.3.2.16 TXX

#### C++

```
HRESULT get_TXX( double* );
```

#### C#

```
public double TXX { get; }
```

#### Visual Basic

```
Public ReadOnly TXX As double
```

#### Description

TXX shear stress value.

#### Version

Available since version 2.5.

### III.5.3.2.17 TYY

#### C++

```
HRESULT get_TYY( double* );
```

#### C#

```
public double TYY { get; }
```

#### Visual Basic

```
Public ReadOnly TYY As double
```

#### Description

TYY shear stress value.

#### Version

Available since version 2.5.

### III.5.3.2.18 UXX

#### C++

```
HRESULT get_UXX( double* );
```

#### C#

```
public double UXX { get; }
```

#### Visual Basic

```
Public ReadOnly UXX As double
```

#### Description

UXX displacement.

#### Version

Available since version 2.5.

### III.5.3.2.19 UYY

#### C++

```
HRESULT get_UYY( double* );
```

**C#**

```
public double UYY { get; }
```

**Visual Basic**

```
Public ReadOnly UYY As Double
```

**Description**

UYY displacement .

**Version**

Available since version 2.5.

**III.5.3.2.20 WNorm****C++**

```
HRESULT get_WNorm( double* );
```

**C#**

```
public double WNorm { get; }
```

**Visual Basic**

```
Public ReadOnly WNorm As Double
```

**Description**

WNorm displacement .

**Version**

Available since version 2.5.

**III.5.4 IRobotFeResultServer****Class Hierarchy****C++**

```
interface IRobotFeResultServer : IDispatch;
```

**C#**

```
public interface IRobotFeResultServer;
```

**Visual Basic**

```
Public Interface IRobotFeResultServer
```

**Description**

Result server for planar finite elements.

**Version**

Available since version 2.5.

**III.5.4.1 IRobotFeResultServer Members**

The following tables list the members exposed by IRobotFeResultServer.

**Public Fields**

	Name	Description
!	PanelCuts ( see page 946)	Manager of panel cuts .

## Public Methods

	Name	Description
⇨	Complex (⇨ see page 947)	Function returns a set of complex results corresponding to the parameters defined in the argument.
⇨	Detailed (⇨ see page 948)	Function returns the detailed result vector corresponding to the parameters defined in the argument. .
⇨	MaxValue (⇨ see page 948)	
⇨	MinValue (⇨ see page 948)	
⇨	MultiMaxValue (⇨ see page 949)	Function searches and returns the maximal values for all the specified types of results. Parameters of search for the maximal values are common for all the indicated types of results.
⇨	MultiMinValue (⇨ see page 949)	Function searches and returns the minimal values for all the specified types of results. Parameters of search for the minimal values are common for all the indicated types of results.
⇨	Principal (⇨ see page 949)	Function returns the principal result vector corresponding to the parameters defined in the argument. .
⇨	Reduced (⇨ see page 950)	Function returns the vector of reduced results for panels corresponding to the specified parameters.
⇨	ReducedEx (⇨ see page 950)	Function returns a vector of reduced results for panels, corresponding to the specified parameters.
⇨	Reinforcement (⇨ see page 950)	Function returns a result vector for the reinforcement corresponding to the specified parameters. There are two variants of obtaining results by the appropriate setting of arguments: <ol style="list-style-type: none"> <li>first argument determines number of a panel, second argument determines node number</li> <li>first argument determines number of a finite element, while second argument is set as zero.</li> </ol>

## III.5.4.2 IRobotFeResultServer Fields

The fields of the IRobotFeResultServer class are listed here.

### Public Fields

	Name	Description
⇨	PanelCuts (⇨ see page 946)	Manager of panel cuts .

## III.5.4.2.1 PanelCuts

### C++

```
HRESULT get_PanelCuts( IRobotPanelCutMngr** );
```

### C#

```
public IRobotPanelCutMngr PanelCuts { get; }
```

### Visual Basic

```
Public ReadOnly PanelCuts As IRobotPanelCutMngr
```

### Description

Manager of panel cuts .

### Version

Available since version 3.

## III.5.4.3 IRobotFeResultServer Methods

The methods of the IRobotFeResultServer class are listed here.

## Public Methods

	Name	Description
Complex (see page 947)	Function returns a set of complex results corresponding to the parameters defined in the argument.	
Detailed (see page 948)	Function returns the detailed result vector corresponding to the parameters defined in the argument.	
MaxValue (see page 948)		
MinValue (see page 948)		
MultiMaxValue (see page 949)	Function searches and returns the maximal values for all the specified types of results. Parameters of search for the maximal values are common for all the indicated types of results.	
MultiMinValue (see page 949)	Function searches and returns the minimal values for all the specified types of results. Parameters of search for the minimal values are common for all the indicated types of results.	
Principal (see page 949)	Function returns the principal result vector corresponding to the parameters defined in the argument.	
Reduced (see page 950)	Function returns the vector of reduced results for panels corresponding to the specified parameters.	
ReducedEx (see page 950)	Function returns a vector of reduced results for panels, corresponding to the specified parameters.	
Reinforcement (see page 950)	Function returns a result vector for the reinforcement corresponding to the specified parameters. There are two variants of obtaining results by the appropriate setting of arguments: <ol style="list-style-type: none"> <li>first argument determines number of a panel, second argument determines node number</li> <li>first argument determines number of a finite element, while second argument is set as zero.</li> </ol>	

### III.5.4.3.1 Complex

#### C++

```
HRESULT Complex( IRobotFeResultParams* __param, IRobotFeResultComplex** ret );
```

#### C#

```
public IRobotFeResultComplex Complex( IRobotFeResultParams __param );
```

#### Visual Basic

```
Public Function Complex(ByRef __param As IRobotFeResultParams) As IRobotFeResultComplex
```

#### Description

Function returns a set of complex results corresponding to the parameters defined in the argument.

#### Version

Available since version 3.

### III.5.4.3.2 Detailed

#### C++

```
HRESULT Detailed( IRobotFeResultParams* __param, IRobotFeResultDetailed** ret );
```

#### C#

```
public IRobotFeResultDetailed Detailed( IRobotFeResultParams __param );
```

**Visual Basic**

```
Public Function Detailed(ByRef _param As IRobotFeResultParams) As IRobotFeResultDetailed
```

**Description**

Function returns the detailed result vector corresponding to the parameters defined in the argument. .

**Version**

Available since version 2.5.

**III.5.4.3.3 MaxValue****C++**

```
HRESULT MaxValue(IRobotFeExtremeParams* _params, IRobotFeExtremeValue** ret);
```

**C#**

```
public IRobotFeExtremeValue MaxValue(IRobotFeExtremeParams _params);
```

**Visual Basic**

```
Public Function MaxValue(ByRef _params As IRobotFeExtremeParams) As IRobotFeExtremeValue
```

**Version**

Available since version 4.

**III.5.4.3.4 MinValue****C++**

```
HRESULT MinValue(IRobotFeExtremeParams* _params, IRobotFeExtremeValue** ret);
```

**C#**

```
public IRobotFeExtremeValue MinValue(IRobotFeExtremeParams _params);
```

**Visual Basic**

```
Public Function MinValue(ByRef _params As IRobotFeExtremeParams) As IRobotFeExtremeValue
```

**Version**

Available since version 4.

**III.5.4.3.5 Multi.MaxValue****C++**

```
HRESULT Multi.MaxValue(IRobotFeExtremeParams* _params, IRobotFeMultiResultType* _result_ids,
IRobotFeMultiExtremeValue** ret);
```

**C#**

```
public IRobotFeMultiExtremeValue Multi.MaxValue(IRobotFeExtremeParams _params,
IRobotFeMultiResultType _result_ids);
```

**Visual Basic**

```
Public Function Multi.MaxValue(ByRef _params As IRobotFeExtremeParams, ByRef _result_ids As
IRobotFeMultiResultType) As IRobotFeMultiExtremeValue
```

**Description**

Function searches and returns the maximal values for all the specified types of results. Parameters of search for the maximal values are common for all the indicated types of results.

**Version**

Available since version 4.1.

**III.5.4.3.6 MultiMinValue****C++**

```
HRESULT MultiMinValue(IRobotFeExtremeParams* _params, IRobotFeMultiResultType* _result_ids,
IRobotFeMultiExtremeValue** ret);
```

**C#**

```
public IRobotFeMultiExtremeValue MultiMinValue(IRobotFeExtremeParams _params,
IRobotFeMultiResultType _result_ids);
```

**Visual Basic**

```
Public Function MultiMinValue(ByRef _params As IRobotFeExtremeParams, ByRef _result_ids As
IRobotFeMultiResultType) As IRobotFeMultiExtremeValue
```

**Description**

Function searches and returns the minimal values for all the specified types of results. Parameters of search for the minimal values are common for all the indicated types of results.

**Version**

Available since version 4.1.

**III.5.4.3.7 Principal****C++**

```
HRESULT Principal(IRobotFeResultParams* _param, IRobotFeResultPrincipal** ret);
```

**C#**

```
public IRobotFeResultPrincipal Principal(IRobotFeResultParams _param);
```

**Visual Basic**

```
Public Function Principal(ByRef _param As IRobotFeResultParams) As IRobotFeResultPrincipal
```

**Description**

Function returns the principal result vector corresponding to the parameters defined in the argument. .

**Version**

Available since version 2.5.

**III.5.4.3.8 Reduced****C++**

```
HRESULT Reduced(long _panel, IRobotFeResultReducedCutPosition _cut, long _case,
IRobotFeResultReduced** ret);
```

**C#**

```
public IRobotFeResultReduced Reduced(long _panel, IRobotFeResultReducedCutPosition _cut,
long _case);
```

**Visual Basic**

```
Public Function Reduced(_panel As long, _cut As IRobotFeResultReducedCutPosition, _case As
long) As IRobotFeResultReduced
```

## Description

Function returns the vector of reduced results for panels corresponding to the specified parameters.

## Version

Available since version 3.

### III.5.4.3.9 ReducedEx

#### C++

```
HRESULT ReducedEx(long _panel, long _part_idx, IRobotFeResultReducedCutPosition _cut_pos,
long _case, long _case_cmpnt = 1, IRobotFeResultReduced** ret);
```

#### C#

```
public IRobotFeResultReduced ReducedEx(long _panel, long _part_idx,
IRobotFeResultReducedCutPosition _cut_pos, long _case, long _case_cmpnt = 1);
```

## Visual Basic

```
Public Function ReducedEx(_panel As long, _part_idx As long, _cut_pos As
IRobotFeResultReducedCutPosition, _case As long, Optional _case_cmpnt As long = 1) As
IRobotFeResultReduced
```

## Description

Function returns a vector of reduced results for panels, corresponding to the specified parameters.

## Version

Available since version 5.5.

### III.5.4.3.10 Reinforcement

#### C++

```
HRESULT Reinforcement(long _elem_or_panel, long _node = 0, IRobotFeResultReinforcement** ret);
```

#### C#

```
public IRobotFeResultReinforcement Reinforcement(long _elem_or_panel, long _node = 0);
```

## Visual Basic

```
Public Function Reinforcement(_elem_or_panel As long, Optional _node As long = 0) As
IRobotFeResultReinforcement
```

## Description

Function returns a result vector for the reinforcement corresponding to the specified parameters. There are two variants of obtaining results by the appropriate setting of arguments:

1. first argument determines number of a panel, second argument determines node number
2. first argument determines number of a finite element, while second argument is set as zero.

## Version

Available since version 3.

### III.5.5 IRobotReinforceCalcMethods

#### C++

```
enum IRobotReinforceCalcMethods;
```

**C#**

```
public enum IRobotReinforceCalcMethods;
```

**Visual Basic**

```
Public Enum IRobotReinforceCalcMethods
```

**Members**

Members	Description
I_RCM_NEN = 1	NEN hypothesis. Available since version 2.5.
I_RCM_WOOD_ARMER = 2	WOOD&ARMER hypothesis. Available since version 2.5.
I_RCM_ANALYTICAL = 3	Available since version 5.5.

**Description**

Types of the reinforcement calculation hypotheses.

**Version**

Available since version 2.5.

**III.5.6 IRobotFeResultPrincipal****Class Hierarchy****C++**

```
interface IRobotFeResultPrincipal : IDispatch;
```

**C#**

```
public interface IRobotFeResultPrincipal;
```

**Visual Basic**

```
Public Interface IRobotFeResultPrincipal
```

**Description**

Principal result vector.

**Version**

Available since version 2.5.

**III.5.6.1 IRobotFeResultPrincipal Members**

The following tables list the members exposed by IRobotFeResultPrincipal.

**Public Fields**

	Name	Description
◆	M1 (☞ see page 953)	Value of M1 moment.
◆	M1_2 (☞ see page 953)	Value of M(1-2) moment .
◆	M2 (☞ see page 953)	Value of M2 moment .
◆	MAL (☞ see page 953)	Mal angle.
◆	N1 (☞ see page 954)	Value of N1 membrane force .
◆	N1_2 (☞ see page 954)	Value of N(1-2) membrane force .
◆	N2 (☞ see page 954)	Value of N2 membrane force .
◆	NAL (☞ see page 955)	Nal angle.
◆	Q1_2 (☞ see page 955)	Value of Q(1-2) shear force .

◆	S1 (see page 955)	Value of s1 stress .
◆	S1_2 (see page 956)	Value of s(1-2) stress .
◆	S2 (see page 956)	Value of s2 stress .
◆	SAL (see page 956)	Sal angle.
◆	T1_2 (see page 956)	Value of T(1-2) shear stress .
◆	U (see page 957)	Value of U total displacement.
◆	UGX (see page 957)	Value of UGX global displacement.
◆	UGY (see page 957)	Value of UGY global displacement.
◆	UGZ (see page 958)	Value of UGZ global displacement.

### III.5.6.2 IRobotFeResultPrincipal Fields

The fields of the IRobotFeResultPrincipal class are listed here.

#### Public Fields

	Name	Description
◆	M1 (see page 953)	Value of M1 moment.
◆	M1_2 (see page 953)	Value of M(1-2) moment .
◆	M2 (see page 953)	Value of M2 moment .
◆	MAL (see page 953)	Mal angle.
◆	N1 (see page 954)	Value of N1 membrane force .
◆	N1_2 (see page 954)	Value of N(1-2) membrane force .
◆	N2 (see page 954)	Value of N2 membrane force .
◆	NAL (see page 955)	Nal angle.
◆	Q1_2 (see page 955)	Value of Q(1-2) shear force .
◆	S1 (see page 955)	Value of s1 stress .
◆	S1_2 (see page 956)	Value of s(1-2) stress .
◆	S2 (see page 956)	Value of s2 stress .
◆	SAL (see page 956)	Sal angle.
◆	T1_2 (see page 956)	Value of T(1-2) shear stress .
◆	U (see page 957)	Value of U total displacement.
◆	UGX (see page 957)	Value of UGX global displacement.
◆	UGY (see page 957)	Value of UGY global displacement.
◆	UGZ (see page 958)	Value of UGZ global displacement.

#### III.5.6.2.1 M1

##### C++

```
HRESULT get_M1( double* );
```

##### C#

```
public double M1 { get; }
```

##### Visual Basic

```
Public ReadOnly M1 As Double
```

##### Description

Value of M1 moment.

##### Version

Available since version 2.5.

### III.5.6.2.2 M1\_2

**C++**

```
HRESULT get_M1_2( double* );
```

**C#**

```
public double M1_2 { get; }
```

**Visual Basic**

```
Public ReadOnly M1_2 As double
```

**Description**

Value of M(1-2) moment .

**Version**

Available since version 2.5.

### III.5.6.2.3 M2

**C++**

```
HRESULT get_M2( double* );
```

**C#**

```
public double M2 { get; }
```

**Visual Basic**

```
Public ReadOnly M2 As double
```

**Description**

Value of M2 moment .

**Version**

Available since version 2.5.

### III.5.6.2.4 MAL

**C++**

```
HRESULT get_MAL( double* );
```

**C#**

```
public double MAL { get; }
```

**Visual Basic**

```
Public ReadOnly MAL As double
```

**Description**

Mal angle.

**Version**

Available since version 2.5.

### III.5.6.2.5 N1

**C++**

```
HRESULT get_N1( double* );
```

**C#**

```
public double N1 { get; }
```

**Visual Basic**

```
Public ReadOnly N1 As Double
```

**Description**

Value of N1 membrane force .

**Version**

Available since version 2.5.

### III.5.6.2.6 N1\_2

**C++**

```
HRESULT get_N1_2( Double* );
```

**C#**

```
public double N1_2 { get; }
```

**Visual Basic**

```
Public ReadOnly N1_2 As Double
```

**Description**

Value of N(1-2) membrane force .

**Version**

Available since version 2.5.

### III.5.6.2.7 N2

**C++**

```
HRESULT get_N2( Double* );
```

**C#**

```
public double N2 { get; }
```

**Visual Basic**

```
Public ReadOnly N2 As Double
```

**Description**

Value of N2 membrane force .

**Version**

Available since version 2.5.

### III.5.6.2.8 NAL

**C++**

```
HRESULT get_NAL( Double* );
```

**C#**

```
public double NAL { get; }
```

**Visual Basic**

```
Public ReadOnly NAL As Double
```

**Description**

Nal angle.

**Version**

Available since version 2.5.

**III.5.6.2.9 Q1\_2****C++**

```
HRESULT get_Q1_2( double* );
```

**C#**

```
public double Q1_2 { get; }
```

**Visual Basic**

```
Public ReadOnly Q1_2 As Double
```

**Description**

Value of Q(1-2) shear force .

**Version**

Available since version 2.5.

**III.5.6.2.10 S1****C++**

```
HRESULT get_S1( double* );
```

**C#**

```
public double S1 { get; }
```

**Visual Basic**

```
Public ReadOnly S1 As Double
```

**Description**

Value of s1 stress .

**Version**

Available since version 2.5.

**III.5.6.2.11 S1\_2****C++**

```
HRESULT get_S1_2( double* );
```

**C#**

```
public double S1_2 { get; }
```

**Visual Basic**

```
Public ReadOnly S1_2 As Double
```

**Description**

Value of s(1-2) stress .

**Version**

Available since version 2.5.

### III.5.6.2.12 S2

**C++**

```
HRESULT get_S2( double* );
```

**C#**

```
public double S2 { get; }
```

**Visual Basic**

```
Public ReadOnly S2 As Double
```

**Description**

Value of s2 stress .

**Version**

Available since version 2.5.

### III.5.6.2.13 SAL

**C++**

```
HRESULT get_SAL( double* );
```

**C#**

```
public double SAL { get; }
```

**Visual Basic**

```
Public ReadOnly SAL As Double
```

**Description**

Sal angle.

**Version**

Available since version 2.5.

### III.5.6.2.14 T1\_2

**C++**

```
HRESULT get_T1_2( double* );
```

**C#**

```
public double T1_2 { get; }
```

**Visual Basic**

```
Public ReadOnly T1_2 As Double
```

**Description**

Value of T(1-2) shear stress .

**Version**

Available since version 2.5.

### III.5.6.2.15 U

**C++**

```
HRESULT get_U( double* );
```

**C#**

```
public double U { get; }
```

**Visual Basic**

```
Public ReadOnly U As Double
```

**Description**

Value of U total displacement.

**Version**

Available since version 2.5.

### III.5.6.2.16 UGX

**C++**

```
HRESULT get_UGX( Double* );
```

**C#**

```
public double UGX { get; }
```

**Visual Basic**

```
Public ReadOnly UGX As Double
```

**Description**

Value of UGX global displacement.

**Version**

Available since version 2.5.

### III.5.6.2.17 UGY

**C++**

```
HRESULT get_UGY( Double* );
```

**C#**

```
public double UGY { get; }
```

**Visual Basic**

```
Public ReadOnly UGY As Double
```

**Description**

Value of UGY global displacement.

**Version**

Available since version 2.5.

### III.5.6.2.18 UGZ

**C++**

```
HRESULT get_UGZ( Double* );
```

**C#**

```
public double UGZ { get; }
```

**Visual Basic**

```
Public ReadOnly UGZ As Double
```

**Description**

Value of UGZ global displacement.

**Version**

Available since version 2.5.

**III.5.7 IRobotFeResultComplex****Class Hierarchy****C++**

```
interface IRobotFeResultComplex : IDispatch;
```

**C#**

```
public interface IRobotFeResultComplex;
```

**Visual Basic**

```
Public Interface IRobotFeResultComplex
```

**Description**

Complex result vector.

**Version**

Available since version 3.

**III.5.7.1 IRobotFeResultComplex Members**

The following tables list the members exposed by IRobotFeResultComplex.

**Public Fields**

	Name	Description
❖	M_MISES (☞ see page 959)	Value of reduced moment for surface finite elements.
❖	MXX_BOTTOM (☞ see page 959)	Value of MXX designing moment of bottom reinforcement.
❖	MXX_TOP (☞ see page 960)	Value of MXX designing moment of top reinforcement.
❖	MYY_BOTTOM (☞ see page 960)	Value of MYY designing moment of bottom reinforcement.
❖	MYY_TOP (☞ see page 960)	Value of MYY designing moment of top reinforcement.
❖	N_MISES (☞ see page 961)	Value of reduced membrane force for surface finite elements .
❖	S_MISES (☞ see page 961)	Reduced stress value for surface finite elements.

**III.5.7.2 IRobotFeResultComplex Fields**

The fields of the IRobotFeResultComplex class are listed here.

**Public Fields**

	Name	Description
❖	M_MISES (☞ see page 959)	Value of reduced moment for surface finite elements.
❖	MXX_BOTTOM (☞ see page 959)	Value of MXX designing moment of bottom reinforcement.
❖	MXX_TOP (☞ see page 960)	Value of MXX designing moment of top reinforcement.
❖	MYY_BOTTOM (☞ see page 960)	Value of MYY designing moment of bottom reinforcement.
❖	MYY_TOP (☞ see page 960)	Value of MYY designing moment of top reinforcement.
❖	N_MISES (☞ see page 961)	Value of reduced membrane force for surface finite elements .
❖	S_MISES (☞ see page 961)	Reduced stress value for surface finite elements.

### III.5.7.2.1 M\_MISES

#### C++

```
HRESULT get_M_MISES( double* );
```

#### C#

```
public double M_MISES { get; }
```

#### Visual Basic

```
Public ReadOnly M_MISES As double
```

#### Description

Value of reduced moment for surface finite elements.

#### Version

Available since version 3.

### III.5.7.2.2 MXX\_BOTTOM

#### C++

```
HRESULT get_MXX_BOTTOM( double* );
```

#### C#

```
public double MXX_BOTTOM { get; }
```

#### Visual Basic

```
Public ReadOnly MXX_BOTTOM As double
```

#### Description

Value of MXX designing moment of bottom reinforcement.

#### Version

Available since version 3.

### III.5.7.2.3 MXX\_TOP

#### C++

```
HRESULT get_MXX_TOP( double* );
```

#### C#

```
public double MXX_TOP { get; }
```

#### Visual Basic

```
Public ReadOnly MXX_TOP As double
```

#### Description

Value of MXX designing moment of top reinforcement.

#### Version

Available since version 3.

### III.5.7.2.4 MYY\_BOTTOM

#### C++

```
HRESULT get_MYY_BOTTOM( double* );
```

**C#**

```
public double MYY_BOTTOM { get; }
```

**Visual Basic**

```
Public ReadOnly MYY_BOTTOM As Double
```

**Description**

Value of MYY designing moment of bottom reinforcement.

**Version**

Available since version 3.

### III.5.7.2.5 MYY\_TOP

**C++**

```
HRESULT get_MYY_TOP( Double* );
```

**C#**

```
public double MYY_TOP { get; }
```

**Visual Basic**

```
Public ReadOnly MYY_TOP As Double
```

**Description**

Value of MYY designing moment of top reinforcement.

**Version**

Available since version 3.

### III.5.7.2.6 N\_MISES

**C++**

```
HRESULT get_N_MISES( Double* );
```

**C#**

```
public double N_MISES { get; }
```

**Visual Basic**

```
Public ReadOnly N_MISES As Double
```

**Description**

Value of reduced membrane force for surface finite elements .

**Version**

Available since version 3.

### III.5.7.2.7 S\_MISES

**C++**

```
HRESULT get_S_MISES( Double* );
```

**C#**

```
public double S_MISES { get; }
```

**Visual Basic**

```
Public ReadOnly S_MISES As Double
```

**Description**

Reduced stress value for surface finite elements.

**Version**

Available since version 3.

**III.5.8 IRobotFeResultReinforcement****Class Hierarchy****C++**

```
interface IRobotFeResultReinforcement : IDispatch;
```

**C#**

```
public interface IRobotFeResultReinforcement;
```

**Visual Basic**

```
Public Interface IRobotFeResultReinforcement
```

**Description**

Result vector for reinforcement.

**Version**

Available since version 3.

**III.5.8.1 IRobotFeResultReinforcement Members**

The following tables list the members exposed by IRobotFeResultReinforcement.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	A_MIN (☞ see page 962)	Minimum reinforcement area for one layer and one reinforcement direction.
❖	AX (☞ see page 963)	Cracking (X axis direction).
❖	AX_BOTTOM (☞ see page 963)	Bottom reinforcement area (X axis).
❖	AX_TOP (☞ see page 963)	Top reinforcement area (X axis).
❖	AY (☞ see page 964)	Cracking (Y axis direction).
❖	AY_BOTTOM (☞ see page 964)	Bottom reinforcement area (Y axis).
❖	AY_TOP (☞ see page 964)	Top reinforcement area (Y axis).
❖	CalcError (☞ see page 965)	
❖	E_AX_BOTTOM (☞ see page 965)	Bottom reinforcement spacing (X axis).
❖	E_AX_TOP (☞ see page 965)	Top reinforcement spacing (X axis).
❖	E_AY_BOTTOM (☞ see page 965)	Bottom reinforcement spacing (Y axis).
❖	E_AY_TOP (☞ see page 966)	Top reinforcement spacing (Y axis).
❖	F (☞ see page 966)	Deflection value.

**III.5.8.2 IRobotFeResultReinforcement Fields**

The fields of the IRobotFeResultReinforcement class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	A_MIN (☞ see page 962)	Minimum reinforcement area for one layer and one reinforcement direction.

◆	AX ( [ see page 963)	Cracking (X axis direction).
◆	AX_BOTTOM ( [ see page 963)	Bottom reinforcement area (X axis).
◆	AX_TOP ( [ see page 963)	Top reinforcement area (X axis).
◆	AY ( [ see page 964)	Cracking (Y axis direction).
◆	AY_BOTTOM ( [ see page 964)	Bottom reinforcement area (Y axis).
◆	AY_TOP ( [ see page 964)	Top reinforcement area (Y axis).
◆	CalcError ( [ see page 965)	
◆	E_AX_BOTTOM ( [ see page 965)	Bottom reinforcement spacing (X axis).
◆	E_AX_TOP ( [ see page 965)	Top reinforcement spacing (X axis).
◆	E_AY_BOTTOM ( [ see page 965)	Bottom reinforcement spacing (Y axis).
◆	E_AY_TOP ( [ see page 966)	Top reinforcement spacing (Y axis).
◆	F ( [ see page 966)	Deflection value.

### III.5.8.2.1 A\_MIN

#### C++

```
HRESULT get_A_MIN( double* );
```

#### C#

```
public double A_MIN { get; }
```

#### Visual Basic

```
Public ReadOnly A_MIN As Double
```

#### Description

Minimum reinforcement area for one layer and one reinforcement direction.

#### Version

Available since version 3.

### III.5.8.2.2 AX

#### C++

```
HRESULT get_AX( double* );
```

#### C#

```
public double AX { get; }
```

#### Visual Basic

```
Public ReadOnly AX As Double
```

#### Description

Cracking (X axis direction).

#### Version

Available since version 3.

### III.5.8.2.3 AX\_BOTTOM

#### C++

```
HRESULT get_AX_BOTTOM( double* );
```

#### C#

```
public double AX_BOTTOM { get; }
```

**Visual Basic**

```
Public ReadOnly AX_BOTTOM As Double
```

**Description**

Bottom reinforcement area (X axis).

**Version**

Available since version 3.

### III.5.8.2.4 AX\_TOP

**C++**

```
HRESULT get_AX_TOP( Double* );
```

**C#**

```
public double AX_TOP { get; }
```

**Visual Basic**

```
Public ReadOnly AX_TOP As Double
```

**Description**

Top reinforcement area (X axis).

**Version**

Available since version 3.

### III.5.8.2.5 AY

**C++**

```
HRESULT get_AY( Double* );
```

**C#**

```
public double AY { get; }
```

**Visual Basic**

```
Public ReadOnly AY As Double
```

**Description**

Cracking (Y axis direction).

**Version**

Available since version 3.

### III.5.8.2.6 AY\_BOTTOM

**C++**

```
HRESULT get_AY_BOTTOM( Double* );
```

**C#**

```
public double AY_BOTTOM { get; }
```

**Visual Basic**

```
Public ReadOnly AY_BOTTOM As Double
```

**Description**

Bottom reinforcement area (Y axis).

**Version**

Available since version 3.

**III.5.8.2.7 AY\_TOP****C++**

```
HRESULT get_AY_TOP( double* );
```

**C#**

```
public double AY_TOP { get; }
```

**Visual Basic**

```
Public ReadOnly AY_TOP As Double
```

**Description**

Top reinforcement area (Y axis).

**Version**

Available since version 3.

**III.5.8.2.8 CalcError****C++**

```
HRESULT get_CalcError(VARIANT_BOOL* );
```

**C#**

```
public bool CalcError { get; }
```

**Visual Basic**

```
Public ReadOnly CalcError As Boolean
```

**Version**

Available since version 3.

**III.5.8.2.9 E\_AX\_BOTTOM****C++**

```
HRESULT get_E_AX_BOTTOM( double* );
```

**C#**

```
public double E_AX_BOTTOM { get; }
```

**Visual Basic**

```
Public ReadOnly E_AX_BOTTOM As Double
```

**Description**

Bottom reinforcement spacing (X axis).

**Version**

Available since version 3.

**III.5.8.2.10 E\_AX\_TOP****C++**

```
HRESULT get_E_AX_TOP( double* );
```

**C#**

```
public double E_AX_TOP { get; }
```

**Visual Basic**

```
Public ReadOnly E_AX_TOP As Double
```

**Description**

Top reinforcement spacing (X axis).

**Version**

Available since version 3.

### III.5.8.2.11 E\_AY\_BOTTOM

**C++**

```
HRESULT get_E_AY_BOTTOM( double* );
```

**C#**

```
public double E_AY_BOTTOM { get; }
```

**Visual Basic**

```
Public ReadOnly E_AY_BOTTOM As Double
```

**Description**

Bottom reinforcement spacing (Y axis).

**Version**

Available since version 3.

### III.5.8.2.12 E\_AY\_TOP

**C++**

```
HRESULT get_E_AY_TOP( double* );
```

**C#**

```
public double E_AY_TOP { get; }
```

**Visual Basic**

```
Public ReadOnly E_AY_TOP As Double
```

**Description**

Top reinforcement spacing (Y axis).

**Version**

Available since version 3.

### III.5.8.2.13 F

**C++**

```
HRESULT get_F( double* );
```

**C#**

```
public double F { get; }
```

**Visual Basic**

```
Public ReadOnly F As Double
```

**Description**

Deflection value.

**Version**

Available since version 3.

**III.5.9 IRobotFeResultReduced****Class Hierarchy****C++**

```
interface IRobotFeResultReduced : IDispatch;
```

**C#**

```
public interface IRobotFeResultReduced;
```

**Visual Basic**

```
Public Interface IRobotFeResultReduced
```

**Description**

Reduced results for panels.

**Version**

Available since version 3.

**III.5.9.1 IRobotFeResultReduced Members**

The following tables list the members exposed by IRobotFeResultReduced.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CutPos ( <a href="#">see page 968</a> )	Definition of the panel cut.
◆	Height ( <a href="#">see page 968</a> )	Panel height.
◆	Length ( <a href="#">see page 968</a> )	Length of panel cut.
◆	MY ( <a href="#">see page 969</a> )	MRy reduced moment outside the panel plane .
◆	MZ ( <a href="#">see page 969</a> )	MRz reduced moment in the panel plane.
◆	NodeLeftBottom ( <a href="#">see page 969</a> )	
◆	NodeLeftTop ( <a href="#">see page 969</a> )	
◆	NodeRightBottom ( <a href="#">see page 970</a> )	
◆	NodeRightTop ( <a href="#">see page 970</a> )	
◆	NX ( <a href="#">see page 970</a> )	NRx reduced force in the panel plane.
◆	SE ( <a href="#">see page 970</a> )	SRe normal stress in the middle layer.
◆	SO ( <a href="#">see page 971</a> )	SRo normal stress in the middle layer.
◆	T ( <a href="#">see page 971</a> )	TR shear stress in the middle layer.
◆	TY ( <a href="#">see page 971</a> )	TRy reduced force in the panel plane.
◆	TZ ( <a href="#">see page 972</a> )	TRz shear force outside the panel plane.

**III.5.9.2 IRobotFeResultReduced Fields**

The fields of the IRobotFeResultReduced class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CutPos ( <a href="#">see page 968</a> )	Definition of the panel cut.

◆	Height ( [ see page 968)	Panel height.
◆	Length ( [ see page 968)	Length of panel cut.
◆	MY ( [ see page 969)	MRy reduced moment outside the panel plane .
◆	MZ ( [ see page 969)	MRz reduced moment in the panel plane.
◆	NodeLeftBottom ( [ see page 969)	
◆	NodeLeftTop ( [ see page 969)	
◆	NodeRightBottom ( [ see page 970)	
◆	NodeRightTop ( [ see page 970)	
◆	NX ( [ see page 970)	NRx reduced force in the panel plane.
◆	SE ( [ see page 970)	SRe normal stress in the middle layer.
◆	SO ( [ see page 971)	SRo normal stress in the middle layer.
◆	T ( [ see page 971)	TR shear stress in the middle layer.
◆	TY ( [ see page 971)	TRy reduced force in the panel plane.
◆	TZ ( [ see page 972)	TRz shear force outside the panel plane.

### III.5.9.2.1 CutPos

#### C++

```
HRESULT get_CutPos( IRobotFeResultReducedCutPosition* );
```

#### C#

```
public IRobotFeResultReducedCutPosition CutPos { get; }
```

#### Visual Basic

```
Public ReadOnly CutPos As IRobotFeResultReducedCutPosition
```

#### Description

Definition of the panel cut.

#### Version

Available since version 5.5.

### III.5.9.2.2 Height

#### C++

```
HRESULT get_Height( double* );
```

#### C#

```
public double Height { get; }
```

#### Visual Basic

```
Public ReadOnly Height As Double
```

#### Description

Panel height.

#### Version

Available since version 3.

### III.5.9.2.3 Length

#### C++

```
HRESULT get_Length( double* );
```

**C#**

```
public double Length { get; }
```

**Visual Basic**

```
Public ReadOnly Length As Double
```

**Description**

Length of panel cut.

**Version**

Available since version 3.

**III.5.9.2.4 MY****C++**

```
HRESULT get_MY( Double* );
```

**C#**

```
public double MY { get; }
```

**Visual Basic**

```
Public ReadOnly MY As Double
```

**Description**

MRy reduced moment outside the panel plane .

**Version**

Available since version 3.

**III.5.9.2.5 MZ****C++**

```
HRESULT get_MZ( Double* );
```

**C#**

```
public double MZ { get; }
```

**Visual Basic**

```
Public ReadOnly MZ As Double
```

**Description**

MRz reduced moment in the panel plane.

**Version**

Available since version 3.

**III.5.9.2.6 NodeLeftBottom****C++**

```
HRESULT get_NodeLeftBottom( Long* );
```

**C#**

```
public long NodeLeftBottom { get; }
```

**Visual Basic**

```
Public ReadOnly NodeLeftBottom As Long
```

**Version**

Available since version 5.5.

**III.5.9.2.7 NodeLeftTop****C++**

```
HRESULT get_NodeLeftTop( long* );
```

**C#**

```
public long NodeLeftTop { get; }
```

**Visual Basic**

```
Public ReadOnly NodeLeftTop As long
```

**Version**

Available since version 5.5.

**III.5.9.2.8 NodeRightBottom****C++**

```
HRESULT get_NodeRightBottom( long* );
```

**C#**

```
public long NodeRightBottom { get; }
```

**Visual Basic**

```
Public ReadOnly NodeRightBottom As long
```

**Version**

Available since version 5.5.

**III.5.9.2.9 NodeRightTop****C++**

```
HRESULT get_NodeRightTop( long* );
```

**C#**

```
public long NodeRightTop { get; }
```

**Visual Basic**

```
Public ReadOnly NodeRightTop As long
```

**Version**

Available since version 5.5.

**III.5.9.2.10 NX****C++**

```
HRESULT get_NX( double* );
```

**C#**

```
public double NX { get; }
```

**Visual Basic**

```
Public ReadOnly NX As double
```

**Description**

NRx reduced force in the panel plane.

**Version**

Available since version 3.

**III.5.9.2.11 SE****C++**

```
HRESULT get_SE( double* );
```

**C#**

```
public double SE { get; }
```

**Visual Basic**

```
Public ReadOnly SE As Double
```

**Description**

SRe normal stress in the middle layer.

**Version**

Available since version 3.

**III.5.9.2.12 SO****C++**

```
HRESULT get_SO( double* );
```

**C#**

```
public double SO { get; }
```

**Visual Basic**

```
Public ReadOnly SO As Double
```

**Description**

SRo normal stress in the middle layer.

**Version**

Available since version 3.

**III.5.9.2.13 T****C++**

```
HRESULT get_T( double* );
```

**C#**

```
public double T { get; }
```

**Visual Basic**

```
Public ReadOnly T As Double
```

**Description**

TR shear stress in the middle layer.

**Version**

Available since version 3.

### III.5.9.2.14 TY

**C++**

```
HRESULT get_TY( double* );
```

**C#**

```
public double TY { get; }
```

**Visual Basic**

```
Public ReadOnly TY As Double
```

**Description**

TRy reduced force in the panel plane.

**Version**

Available since version 3.

### III.5.9.2.15 TZ

**C++**

```
HRESULT get_TZ( double* );
```

**C#**

```
public double TZ { get; }
```

**Visual Basic**

```
Public ReadOnly TZ As Double
```

**Description**

TRz shear force outside the panel plane.

**Version**

Available since version 3.

## III.5.10 IRobotFeResultReducedCutPosition

**C++**

```
enum IRobotFeResultReducedCutPosition;
```

**C#**

```
public enum IRobotFeResultReducedCutPosition;
```

**Visual Basic**

```
Public Enum IRobotFeResultReducedCutPosition
```

**Members**

Members	Description
I_FRRCP_VERTICAL_LEFT = 4	Available since version 3.
I_FRRCP_HORIZONTAL_MIDDLE = 2	Available since version 3.
I_FRRCP_VERTICAL_RIGHT = 6	Available since version 3.
I_FRRCP_HORIZONTAL_TOP = 3	Available since version 3.
I_FRRCP_VERTICAL_MIDDLE = 5	Available since version 3.
I_FRRCP_HORIZONTAL_BOTTOM = 1	Available since version 3.

**Description**

Available positions of a panel cut.

**Version**

Available since version 3.

**III.5.11 IRobotPanelCutMngr****Class Hierarchy****C++**

```
interface IRobotPanelCutMngr : IDispatch;
```

**C#**

```
public interface IRobotPanelCutMngr;
```

**Visual Basic**

```
Public Interface IRobotPanelCutMngr
```

**Description**

Manager of panel cuts. .

**Version**

Available since version 3.

**III.5.11.1 IRobotPanelCutMngr Members**

The following tables list the members exposed by IRobotPanelCutMngr.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 973</a> )	Number of available cuts.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Create ( <a href="#">see page 974</a> )	Function creates and returns a new cut.
◆	Find ( <a href="#">see page 974</a> )	Function returns index of the cut of the specified name. .
◆	Get ( <a href="#">see page 974</a> )	Function returns the cut of the indicated index. .
◆	GetName ( <a href="#">see page 975</a> )	Function returns name of the cut of the indicated index. .
◆	Remove ( <a href="#">see page 975</a> )	Function deletes the cut of the indicated index. .
◆	Store ( <a href="#">see page 975</a> )	Function saves the cut under the specified name. .

**III.5.11.2 IRobotPanelCutMngr Fields**

The fields of the IRobotPanelCutMngr class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 973</a> )	Number of available cuts.

**III.5.11.2.1 Count****C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of available cuts.

**Version**

Available since version 3.

**III.5.11.3 IRobotPanelCutMngr Methods**

The methods of the IRobotPanelCutMngr class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Create (see page 974)	Function creates and returns a new cut.
≡	Find (see page 974)	Function returns index of the cut of the specified name..
≡	Get (see page 974)	Function returns the cut of the indicated index..
≡	GetName (see page 975)	Function returns name of the cut of the indicated index..
≡	Remove (see page 975)	Function deletes the cut of the indicated index..
≡	Store (see page 975)	Function saves the cut under the specified name..

**III.5.11.3.1 Create****C++**

```
HRESULT Create(IRobotPanelCut** ret);
```

**C#**

```
public IRobotPanelCut Create();
```

**Visual Basic**

```
Public Function Create() As IRobotPanelCut
```

**Description**

Function creates and returns a new cut.

**Version**

Available since version 3.

**III.5.11.3.2 Find****C++**

```
HRESULT Find(BSTR _cut_name, long* ret);
```

**C#**

```
public long Find(String _cut_name);
```

**Visual Basic**

```
Public Function Find(_cut_name As String) As Long
```

**Description**

Function returns index of the cut of the specified name..

**Version**

Available since version 3.

**III.5.11.3.3 Get****C++**

```
HRESULT Get(long _idx, IRobotPanelCut** ret);
```

**C#**

```
public IRobotPanelCut Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotPanelCut
```

**Description**

Function returns the cut of the indicated index..

**Version**

Available since version 3.

**III.5.11.3.4 GetName****C++**

```
HRESULT GetName(long _idx, BSTR* ret);
```

**C#**

```
public String GetName(long _idx);
```

**Visual Basic**

```
Public Function GetName(_idx As long) As String
```

**Description**

Function returns name of the cut of the indicated index..

**Version**

Available since version 3.

**III.5.11.3.5 Remove****C++**

```
HRESULT Remove(long _idx);
```

**C#**

```
public void Remove(long _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As long)
```

**Description**

Function deletes the cut of the indicated index..

**Version**

Available since version 3.

### III.5.11.3.6 Store

#### C++

```
HRESULT Store(IRobotPanelCut* _cut, BSTR _name);
```

#### C#

```
public void Store(IRobotPanelCut _cut, String _name);
```

#### Visual Basic

```
Public Sub Store(ByRef _cut As IRobotPanelCut, _name As String)
```

#### Description

Function saves the cut under the specified name. .

#### Version

Available since version 3.

## III.5.12 IRobotPanelCut

#### Class Hierarchy

#### C++

```
interface IRobotPanelCut : IDispatch;
```

#### C#

```
public interface IRobotPanelCut;
```

#### Visual Basic

```
Public Interface IRobotPanelCut
```

#### Description

Interface defining a cut through panel. .

#### Version

Available since version 3.

### III.5.12.1 IRobotPanelCut Members

The following tables list the members exposed by IRobotPanelCut.

#### Public Fields

	Name	Description
◆	Active ( [ see page 976) )	Flag indicating if a given cut is active.
◆	Color ( [ see page 977) )	Color applied while presenting a cut in graphical views .
◆	DefType ( [ see page 977) )	Type of a panel cut definition.
◆	Point1 ( [ see page 977) )	
◆	Point2 ( [ see page 978) )	
◆	Point3 ( [ see page 978) )	

### III.5.12.2 IRobotPanelCut Fields

The fields of the IRobotPanelCut class are listed here.

## Public Fields

	Name	Description
◆	Active ( [ see page 976)	Flag indicating if a given cut is active.
◆	Color ( [ see page 977)	Color applied while presenting a cut in graphical views .
◆	DefType ( [ see page 977)	Type of a panel cut definition.
◆	Point1 ( [ see page 977)	
◆	Point2 ( [ see page 978)	
◆	Point3 ( [ see page 978)	

### III.5.12.2.1 Active

#### C++

```
HRESULT get_Active(VARIANT_BOOL* );
HRESULT put_Active(VARIANT_BOOL);
```

#### C#

```
public bool Active { get; set; }
```

#### Visual Basic

```
Public Active As Boolean
```

#### Description

Flag indicating if a given cut is active.

#### Version

Available since version 3.

### III.5.12.2.2 Color

#### C++

```
HRESULT get_Color( long* );
HRESULT put_Color( long );
```

#### C#

```
public long Color { get; set; }
```

#### Visual Basic

```
Public Color As long
```

#### Description

Color applied while presenting a cut in graphical views .

#### Version

Available since version 3.

### III.5.12.2.3 DefType

#### C++

```
HRESULT get_DefType( IRobotPanelCutDefinitionType* );
HRESULT put_DefType( IRobotPanelCutDefinitionType );
```

#### C#

```
public IRobotPanelCutDefinitionType DefType { get; set; }
```

#### Visual Basic

```
Public DefType As IRobotPanelCutDefinitionType
```

**Description**

Type of a panel cut definition.

**Version**

Available since version 3.

**III.5.12.2.4 Point1****C++**

```
HRESULT get_Point1(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D Point1 { get; }
```

**Visual Basic**

```
Public ReadOnly Point1 As IRobotGeoPoint3D
```

**Version**

Available since version 3.

**III.5.12.2.5 Point2****C++**

```
HRESULT get_Point2(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D Point2 { get; }
```

**Visual Basic**

```
Public ReadOnly Point2 As IRobotGeoPoint3D
```

**Version**

Available since version 3.

**III.5.12.2.6 Point3****C++**

```
HRESULT get_Point3(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D Point3 { get; }
```

**Visual Basic**

```
Public ReadOnly Point3 As IRobotGeoPoint3D
```

**Version**

Available since version 3.

**III.5.13 IRobotPanelCutDefinitionType****C++**

```
enum IRobotPanelCutDefinitionType;
```

**C#**

```
public enum IRobotPanelCutDefinitionType;
```

**Visual Basic**

```
Public Enum IRobotPanelCutDefinitionType
```

**Members**

Members	Description
I_PCDT_FULL_PLANE = 1	Unlimited cut plane, determined by two indicated points. Available since version 3.
I_PCDT_LIMITED_PLANE = 2	A cut plane defined by a segment determined by two indicated points and perpendicular to XY plane. Available since version 3.

**Description**

Types of panel cut definition. .

**Version**

Available since version 3.

**III.5.14 IRobotFeResultType****C++**

```
enum IRobotFeResultType;
```

**C#**

```
public enum IRobotFeResultType;
```

**Visual Basic**

```
Public Enum IRobotFeResultType
```

**Members**

Members	Description
I_FRT_DETAILED_SXX = 483	Available since version 4.
I_FRT_DETAILED_SYY = 484	Available since version 4.
I_FRT_DETAILED_SXY = 485	Available since version 4.
I_FRT_DETAILED_NXX = 492	Available since version 4.
I_FRT_DETAILED_NYY = 493	Available since version 4.
I_FRT_DETAILED_NXY = 494	Available since version 4.
I_FRT_DETAILED_MXX = 501	Available since version 4.
I_FRT_DETAILED_MYY = 502	Available since version 4.
I_FRT_DETAILED_MXY = 503	Available since version 4.
I_FRT_DETAILED_TXX = 510	Available since version 4.
I_FRT_DETAILED_TYY = 511	Available since version 4.
I_FRT_DETAILED_QXX = 519	Available since version 4.
I_FRT_DETAILED_QYY = 520	Available since version 4.
I_FRT_DETAILED_UXX = 537	Available since version 4.
I_FRT_DETAILED_UYY = 538	Available since version 4.
I_FRT_DETAILED_WNORM = 531	Available since version 4.
I_FRT_DETAILED_RXX = 546	Available since version 4.
I_FRT_DETAILED_RYY = 547	Available since version 4.
I_FRT_DETAILED_RNORM = 549	Available since version 4.
I_FRT_DETAILED_PNORM = 558	Available since version 4.
I_FRT_PRINCIPAL_S1 = 487	Available since version 4.
I_FRT_PRINCIPAL_S2 = 488	Available since version 4.

I_FRT_PRINCIPAL_S1_2 = 490	Available since version 4.
I_FRT_PRINCIPAL_SAL = 489	Available since version 4.
I_FRT_PRINCIPAL_N1 = 496	Available since version 4.
I_FRT_PRINCIPAL_N2 = 497	Available since version 4.
I_FRT_PRINCIPAL_N1_2 = 499	Available since version 4.
I_FRT_PRINCIPAL_NAL = 498	Available since version 4.
I_FRT_PRINCIPAL_M1 = 505	Available since version 4.
I_FRT_PRINCIPAL_M2 = 506	Available since version 4.
I_FRT_PRINCIPAL_M1_2 = 508	Available since version 4.
I_FRT_PRINCIPAL_MAL = 507	Available since version 4.
I_FRT_PRINCIPAL_T1_2 = 517	Available since version 4.
I_FRT_PRINCIPAL_Q1_2 = 526	Available since version 4.
I_FRT_PRINCIPAL_UGX = 1302	Available since version 4.
I_FRT_PRINCIPAL_UGY = 1303	Available since version 4.
I_FRT_PRINCIPAL_UGZ = 1304	Available since version 4.
I_FRT_PRINCIPAL_U = 1305	Available since version 4.
I_FRT_COMPLEX_S_MISES = 491	Available since version 4.
I_FRT_COMPLEX_N_MISES = 500	Available since version 4.
I_FRT_COMPLEX_M_MISES = 509	Available since version 4.
I_FRT_COMPLEX_MXX_TOP_WA = 889	Available since version 4.
I_FRT_COMPLEX_MXX_BOTTOM_WA = 890	Available since version 4.
I_FRT_COMPLEX_MYY_TOP_WA = 891	Available since version 4.
I_FRT_COMPLEX_MYY_BOTTOM_WA = 892	Available since version 4.
I_FRT_COMPLEX_MXX_TOP_NEN = 893	Available since version 4.
I_FRT_COMPLEX_MXX_BOTTOM_NEN = 894	Available since version 4.
I_FRT_COMPLEX_MYY_TOP_NEN = 895	Available since version 4.
I_FRT_COMPLEX_MYY_BOTTOM_NEN = 896	Available since version 4.
I_FRT_REINF_AX_TOP = 0	Available since version 4.
I_FRT_REINF_AX_BOTTOM = 1	Available since version 4.
I_FRT_REINF_AY_TOP = 2	Available since version 4.
I_FRT_REINF_AY_BOTTOM = 3	Available since version 4.
I_FRT_REINF_E_AX_TOP = 4	Available since version 4.
I_FRT_REINF_E_AX_BOTTOM = 5	Available since version 4.
I_FRT_REINF_E_AY_TOP = 6	Available since version 4.
I_FRT_REINF_E_AY_BOTTOM = 7	Available since version 4.
I_FRT_REINF_A_MIN = 8	Available since version 4.
I_FRT_REINF_AX = 9	Available since version 4.
I_FRT_REINF_AY = 10	Available since version 4.
I_FRT_REINF_F = 11	Available since version 4.
I_FRT_REDUCED_NX = 1217	Available since version 4.
I_FRT_REDUCED_MZ = 1218	Available since version 4.
I_FRT_REDUCED_TY = 1219	Available since version 4.
I_FRT_REDUCED_SO = 1220	Available since version 4.
I_FRT_REDUCED_SE = 1221	Available since version 4.
I_FRT_REDUCED_T = 1222	Available since version 4.
I_FRT_REDUCED_TZ = 1223	Available since version 4.
I_FRT_REDUCED_MY = 1224	Available since version 4.
I_FRT_REDUCED_LENGTH = 1225	Available since version 4.
I_FRT_REDUCED_HEIGHT = 1226	Available since version 4.

**Description**

Designation of the type of result calculated for finite elements.

**Version**

Available since version 4.

**III.5.15 IRobotFeResultSmoothing****C++**

```
enum IRobotFeResultSmoothing;
```

**C#**

```
public enum IRobotFeResultSmoothing;
```

**Visual Basic**

```
Public Enum IRobotFeResultSmoothing
```

**Members**

Members	Description
I_FRS_NO_SMOOTHING = 0	Exact results in nodes. Available since version 4.
I_FRS_GLOBAL_SMOOTHING = 1	Globally averaged results in nodes. Available since version 4.
I_FRS_SMOOTHING_WITHIN_A_PANEL = 2	Results in nodes - averaged within a panel. Available since version 4.
I_FRS_SMOOTHING_ACCORDING_TO_SELECTION = 3	Results in nodes averaged according to the selection of finite elements to which the nodes belong. Available since version 4.
I_FRS_IN_ELEMENT_CENTER = 4	Averaged results in finite element centers. Available since version 4.

**Description**

Available methods of averaging (smoothing) results.

**Version**

Available since version 4.

**III.5.16 IRobotFeResultKind****C++**

```
enum IRobotFeResultKind;
```

**C#**

```
public enum IRobotFeResultKind;
```

**Visual Basic**

```
Public Enum IRobotFeResultKind
```

**Members**

Members	Description
I_FRK_PANEL_NODE = 0	Available since version 4.
I_FRK_ELEMENT_CENTER = 1	Available since version 4.
I_FRK_ELEMENT_NODE = 2	Available since version 4.

## Version

Available since version 4.

### III.5.17 IRobotFeExtremeParams

#### Class Hierarchy

#### C++

```
interface IRobotFeExtremeParams : IDispatch;
```

#### C#

```
public interface IRobotFeExtremeParams;
```

#### Visual Basic

```
Public Interface IRobotFeExtremeParams
```

#### Description

Parameters of searching extreme results for finite elements.

The list of nodes to be considered while searching extreme values is determined based on the settings in the fields NodeSel (see page 985), ElementSel (see page 984) and PanelSel (see page 985). If a list of nodes is set in the NodeSel (see page 985) field, then this list exactly will be considered in searching extreme values. If the NodeSel (see page 985) field is empty and the ElementSel (see page 984) field is set, then all the nodes belonging to the indicated finite elements will be considered. If the fields NodeSel (see page 985) and ElementSel (see page 984) are empty and the PanelSel (see page 985) field is set, then all the nodes belonging to the indicated panels will be considered. If the fields NodeSel (see page 985), PanelSel (see page 985) and ElementSel (see page 984) are empty, then all the structure nodes will be considered.

For results in finite element centers, the list of elements to be considered is determined based on the ElementSel (see page 984) field. If the ElementSel (see page 984) list is empty and the PanelSel (see page 985) field is set, then all the finite elements belonging to the indicated panels will be considered. If the fields ElementSel (see page 984) and PanelSel (see page 985) are empty, then all the finite elements of the structure will be considered.

## Version

Available since version 4.

### III.5.17.1 IRobotFeExtremeParams Members

The following tables list the members exposed by IRobotFeExtremeParams.

#### Public Fields

	Name	Description
◆	CaseCmpnt (see page 983)	Number of a load case component or mode number.
◆	CaseSel (see page 983)	List of load cases to be considered .
◆	ElementSel (see page 984)	List of finite elements to be considered.
◆	Layer (see page 984)	Layer.
◆	LayerArbitraryValue (see page 984)	Layer (see page 984) thickness.
◆	ModeCmb (see page 985)	Mode combination.
◆	NodeSel (see page 985)	List of nodes to be considered .
◆	PanelSel (see page 985)	List of panels to be considered.
◆	ReducedCutPos (see page 986)	
◆	ResultId (see page 986)	Designation of the result type.

	Smoothing (see page 986)	Method of result averaging.
-----------------------------------------------------------------------------------	--------------------------	-----------------------------

## Public Methods

	Name	Description
	GetDirX (see page 987)	
	Reset (see page 987)	
	SetDirX (see page 987)	

### III.5.17.2 IRobotFeExtremeParams Fields

The fields of the IRobotFeExtremeParams class are listed here.

## Public Fields

	Name	Description
	CaseCmpnt (see page 983)	Number of a load case component or mode number.
	CaseSel (see page 983)	List of load cases to be considered .
	ElementSel (see page 984)	List of finite elements to be considered.
	Layer (see page 984)	Layer.
	LayerArbitraryValue (see page 984)	Layer (see page 984) thickness.
	ModeCmb (see page 985)	Mode combination.
	NodeSel (see page 985)	List of nodes to be considered .
	PanelSel (see page 985)	List of panels to be considered.
	ReducedCutPos (see page 986)	
	ResultId (see page 986)	Designation of the result type.
	Smoothing (see page 986)	Method of result averaging.

### III.5.17.2.1 CaseCmpnt

#### C++

```
HRESULT get_CaseCmpnt(long* );
HRESULT put_CaseCmpnt(long);
```

#### C#

```
public long CaseCmpnt { get; set; }
```

#### Visual Basic

```
Public CaseCmpnt As long
```

#### Description

Number of a load case component or mode number.

#### Version

Available since version 4.1.

### III.5.17.2.2 CaseSel

#### C++

```
HRESULT get_CaseSel(BSTR* );
HRESULT put_CaseSel(BSTR);
```

#### C#

```
public String CaseSel { get; set; }
```

**Visual Basic**

```
Public CaseSel As String
```

**Description**

List of load cases to be considered.

**Version**

Available since version 4.

**III.5.17.2.3 ElementSel****C++**

```
HRESULT get_ElementSel(BSTR* );
HRESULT put_ElementSel(BSTR);
```

**C#**

```
public String ElementSel { get; set; }
```

**Visual Basic**

```
Public ElementSel As String
```

**Description**

List of finite elements to be considered.

**Version**

Available since version 4.

**III.5.17.2.4 Layer****C++**

```
HRESULT get_Layer(IRobotFeLayerType* );
HRESULT put_Layer(IRobotFeLayerType);
```

**C#**

```
public IRobotFeLayerType Layer { get; set; }
```

**Visual Basic**

```
Public Layer As IRobotFeLayerType
```

**Description**

Layer.

**Version**

Available since version 4.

**III.5.17.2.5 LayerArbitraryValue****C++**

```
HRESULT get_LayerArbitraryValue(double* );
HRESULT put_LayerArbitraryValue(double);
```

**C#**

```
public double LayerArbitraryValue { get; set; }
```

**Visual Basic**

```
Public LayerArbitraryValue As double
```

**Description**

Layer (see page 984) thickness.

**Version**

Available since version 4.

**III.5.17.2.6 ModeCmb****C++**

```
HRESULT get_ModeCmb(IRobotModeCombinationType* );
HRESULT put_ModeCmb(IRobotModeCombinationType);
```

**C#**

```
public IRobotModeCombinationType ModeCmb { get; set; }
```

**Visual Basic**

```
Public ModeCmb As IRobotModeCombinationType
```

**Description**

Mode combination.

**Version**

Available since version 4.1.

**III.5.17.2.7 NodeSel****C++**

```
HRESULT get_NodeSel(BSTR* );
HRESULT put_NodeSel(BSTR);
```

**C#**

```
public String NodeSel { get; set; }
```

**Visual Basic**

```
Public NodeSel As String
```

**Description**

List of nodes to be considered .

**Version**

Available since version 4.

**III.5.17.2.8 PanelSel****C++**

```
HRESULT get_PanelSel(BSTR* );
HRESULT put_PanelSel(BSTR);
```

**C#**

```
public String PanelSel { get; set; }
```

**Visual Basic**

```
Public PanelSel As String
```

**Description**

List of panels to be considered.

**Version**

Available since version 4.

**III.5.17.2.9 ReducedCutPos****C++**

```
HRESULT get_ReducedCutPos(IRobotFeResultReducedCutPosition* );
HRESULT put_ReducedCutPos(IRobotFeResultReducedCutPosition);
```

**C#**

```
public IRobotFeResultReducedCutPosition ReducedCutPos { get; set; }
```

**Visual Basic**

```
Public ReducedCutPos As IRobotFeResultReducedCutPosition
```

**Version**

Available since version 4.

**III.5.17.2.10 ResultId****C++**

```
HRESULT get_ResultId(IRobotFeResultType* );
HRESULT put_ResultId(IRobotFeResultType);
```

**C#**

```
public IRobotFeResultType ResultId { get; set; }
```

**Visual Basic**

```
Public ResultId As IRobotFeResultType
```

**Description**

Designation of the result type.

**Version**

Available since version 4.

**III.5.17.2.11 Smoothing****C++**

```
HRESULT get_Smoothing(IRobotFeResultSmoothing* );
HRESULT put_Smoothing(IRobotFeResultSmoothing);
```

**C#**

```
public IRobotFeResultSmoothing Smoothing { get; set; }
```

**Visual Basic**

```
Public Smoothing As IRobotFeResultSmoothing
```

**Description**

Method of result averaging.

**Version**

Available since version 4.

**III.5.17.3 IRobotFeExtremeParams Methods**

The methods of the IRobotFeExtremeParams class are listed here.

## Public Methods

	Name	Description
⊕	GetDirX ( [ see page 987] )	
⊕	Reset ( [ see page 987] )	
⊕	SetDirX ( [ see page 987] )	

### III.5.17.3.1 GetDirX

#### C++

```
HRESULT GetDirX(double* _x, double* _y, double* _z, IRobotObjLocalXDirDefinitionType* ret);
```

#### C#

```
public IRobotObjLocalXDirDefinitionType GetDirX(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetDirX(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As IRobotObjLocalXDirDefinitionType
```

#### Version

Available since version 4.

### III.5.17.3.2 Reset

#### C++

```
HRESULT Reset();
```

#### C#

```
public void Reset();
```

#### Visual Basic

```
Public Sub Reset()
```

#### Version

Available since version 4.

### III.5.17.3.3 SetDirX

#### C++

```
HRESULT SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

#### C#

```
public void SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub SetDirX(_def_type As IRobotObjLocalXDirDefinitionType, _x As double, _y As double, _z As double)
```

#### Version

Available since version 4.

### III.5.18 IRobotFeExtremeValue

#### Class Hierarchy

#### C++

```
interface IRobotFeExtremeValue : IDispatch;
```

#### C#

```
public interface IRobotFeExtremeValue;
```

#### Visual Basic

```
Public Interface IRobotFeExtremeValue
```

#### Description

Interface providing access to information about the extreme value. .

#### Version

Available since version 4.

### III.5.18.1 IRobotFeExtremeValue Members

The following tables list the members exposed by IRobotFeExtremeValue.

#### Public Fields

	Name	Description
❖	Case (see page 989)	
❖	CaseCmpnt (see page 990)	Additional number indicating a load case component <ul style="list-style-type: none"> <li>• for a code combination it is the number of its component</li> <li>• for a modal analysis case it is the number of a mode</li> <li>• for results of time history analysis it is the number of a step.</li> </ul>
❖	Element (see page 990)	
❖	IsAvailable (see page 990)	Flag indicating if the extreme value has been taken correctly from the result server.
❖	Layer (see page 990)	
❖	LayerArbitraryValue (see page 991)	
❖	ModeCmb (see page 991)	
❖	Node (see page 991)	
❖	Panel (see page 991)	
❖	ReducedCutPos (see page 992)	
❖	ResultId (see page 992)	
❖	Smoothing (see page 992)	
❖	Value (see page 992)	

#### Public Methods

	Name	Description
❖	GetDirX (see page 993)	Function takes the direction of the local X axis of the coordinate system, in which the specified extreme value has been obtained. The values read are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns the type of the system direction definition. .

### III.5.18.2 IRobotFeExtremeValue Fields

The fields of the IRobotFeExtremeValue class are listed here.

#### Public Fields

	Name	Description
◆	Case (see page 989)	
◆	CaseCmpnt (see page 990)	Additional number indicating a load case component <ul style="list-style-type: none"> <li>• for a code combination it is the number of its component</li> <li>• for a modal analysis case it is the number of a mode</li> <li>• for results of time history analysis it is the number of a step.</li> </ul>
◆	Element (see page 990)	
◆	IsAvailable (see page 990)	Flag indicating if the extreme value has been taken correctly from the result server.
◆	Layer (see page 990)	
◆	LayerArbitraryValue (see page 991)	
◆	ModeCmb (see page 991)	
◆	Node (see page 991)	
◆	Panel (see page 991)	
◆	ReducedCutPos (see page 992)	
◆	ResultId (see page 992)	
◆	Smoothing (see page 992)	
◆	Value (see page 992)	

#### III.5.18.2.1 Case

##### C++

```
HRESULT get_Case(long*);
```

##### C#

```
public long Case { get; }
```

##### Visual Basic

```
Public ReadOnly Case As long
```

##### Version

Available since version 4.

#### III.5.18.2.2 CaseCmpnt

##### C++

```
HRESULT get_CaseCmpnt(long*);
```

##### C#

```
public long CaseCmpnt { get; }
```

##### Visual Basic

```
Public ReadOnly CaseCmpnt As long
```

##### Description

Additional number indicating a load case component

- for a code combination it is the number of its component

- for a modal analysis case it is the number of a mode
- for results of time history analysis it is the number of a step.

**Version**

Available since version 4.

**III.5.18.2.3 Element****C++**

```
HRESULT get_Element(long*);
```

**C#**

```
public long Element { get; }
```

**Visual Basic**

```
Public ReadOnly Element As long
```

**Version**

Available since version 4.

**III.5.18.2.4 IsAvailable****C++**

```
HRESULT get_IsAvailable(VARIANT_BOOL*);
```

**C#**

```
public bool IsAvailable { get; }
```

**Visual Basic**

```
Public ReadOnly IsAvailable As Boolean
```

**Description**

Flag indicating if the extreme value has been taken correctly from the result server.

**Version**

Available since version 4.

**III.5.18.2.5 Layer****C++**

```
HRESULT get_Layer(IRobotFeLayerType*);
```

**C#**

```
public IRobotFeLayerType Layer { get; }
```

**Visual Basic**

```
Public ReadOnly Layer As IRobotFeLayerType
```

**Version**

Available since version 4.

**III.5.18.2.6 LayerArbitraryValue****C++**

```
HRESULT get_LayerArbitraryValue(double*);
```

**C#**

```
public double LayerArbitraryValue { get; }
```

**Visual Basic**

```
Public ReadOnly LayerArbitraryValue As Double
```

**Version**

Available since version 4.

### III.5.18.2.7 ModeCmb

**C++**

```
HRESULT get_ModeCmb(IRobotModeCombinationType* );
```

**C#**

```
public IRobotModeCombinationType ModeCmb { get; }
```

**Visual Basic**

```
Public ReadOnly ModeCmb As IRobotModeCombinationType
```

**Version**

Available since version 4.

### III.5.18.2.8 Node

**C++**

```
HRESULT get_Node(long* );
```

**C#**

```
public long Node { get; }
```

**Visual Basic**

```
Public ReadOnly Node As Long
```

**Version**

Available since version 4.

### III.5.18.2.9 Panel

**C++**

```
HRESULT get_Panel(long* );
```

**C#**

```
public long Panel { get; }
```

**Visual Basic**

```
Public ReadOnly Panel As Long
```

**Version**

Available since version 4.

### III.5.18.2.10 ReducedCutPos

**C++**

```
HRESULT get_ReducedCutPos(IRobotFeResultReducedCutPosition* );
```

**C#**

```
public IRobotFeResultReducedCutPosition ReducedCutPos { get; }
```

**Visual Basic**

```
Public ReadOnly ReducedCutPos As IRobotFeResultReducedCutPosition
```

**Version**

Available since version 4.

### III.5.18.2.11 ResultId

**C++**

```
HRESULT get_ResultId(IRobotFeResultType*);
```

**C#**

```
public IRobotFeResultType ResultId { get; }
```

**Visual Basic**

```
Public ReadOnly ResultId As IRobotFeResultType
```

**Version**

Available since version 4.

### III.5.18.2.12 Smoothing

**C++**

```
HRESULT get_Smoothing(IRobotFeResultSmoothing*);
```

**C#**

```
public IRobotFeResultSmoothing Smoothing { get; }
```

**Visual Basic**

```
Public ReadOnly Smoothing As IRobotFeResultSmoothing
```

**Version**

Available since version 4.

### III.5.18.2.13 Value

**C++**

```
HRESULT get_Value(double*);
```

**C#**

```
public double Value { get; }
```

**Visual Basic**

```
Public ReadOnly Value As double
```

**Version**

Available since version 4.

### III.5.18.3 IRobotFeExtremeValue Methods

The methods of the IRobotFeExtremeValue class are listed here.

## Public Methods

	Name	Description
💡	GetDirX (see page 993)	Function takes the direction of the local X axis of the coordinate system, in which the specified extreme value has been obtained. The values read are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns the type of the system direction definition. .

### III.5.18.3.1 GetDirX

#### C++

```
HRESULT GetDirX(double* _x, double* _y, double* _z, IRobotObjLocalXDirDefinitionType* ret);
```

#### C#

```
public IRobotObjLocalXDirDefinitionType GetDirX(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetDirX(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As IRobotObjLocalXDirDefinitionType
```

#### Description

Function takes the direction of the local X axis of the coordinate system, in which the specified extreme value has been obtained. The values read are the coordinates of the X axis direction vector for definition in the Cartesian system or the coordinates of the central point for definition in the polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the panel local system. In addition, the function returns the type of the system direction definition. .

#### Version

Available since version 4.

## III.5.19 IRobotFeMultiResultType

#### Class Hierarchy

#### C++

```
interface IRobotFeMultiResultType : IDispatch;
```

#### C#

```
public interface IRobotFeMultiResultType;
```

#### Visual Basic

```
Public Interface IRobotFeMultiResultType
```

#### Description

Interface which enables indicating simultaneously - a great number of result types for finite elements.

#### Version

Available since version 4.1.

### III.5.19.1 IRobotFeMultiResultType Members

The following tables list the members exposed by IRobotFeMultiResultType.

## Public Fields

	Name	Description
◆	Count (see page 994)	

## Public Methods

	Name	Description
◆	Add (see page 994)	
◆	Get (see page 995)	
◆	Remove (see page 995)	

### III.5.19.2 IRobotFeMultiResultType Fields

The fields of the IRobotFeMultiResultType class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 994)	

### III.5.19.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Version

Available since version 4.1.

### III.5.19.3 IRobotFeMultiResultType Methods

The methods of the IRobotFeMultiResultType class are listed here.

## Public Methods

	Name	Description
◆	Add (see page 994)	
◆	Get (see page 995)	
◆	Remove (see page 995)	

### III.5.19.3.1 Add

#### C++

```
HRESULT Add(IRobotFeResultType _result_type);
```

#### C#

```
public void Add(IRobotFeResultType _result_type);
```

#### Visual Basic

```
Public Sub Add(_result_type As IRobotFeResultType)
```

#### Version

Available since version 4.1.

### III.5.19.3.2 Get

**C++**

```
HRESULT Get(long _idx, IRobotFeResultType* ret);
```

**C#**

```
public IRobotFeResultType Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotFeResultType
```

**Version**

Available since version 4.1.

### III.5.19.3.3 Remove

**C++**

```
HRESULT Remove(IRobotFeResultType _result_type);
```

**C#**

```
public void Remove(IRobotFeResultType _result_type);
```

**Visual Basic**

```
Public Sub Remove(_result_type As IRobotFeResultType)
```

**Version**

Available since version 4.1.

## III.5.20 IRobotFeMultiExtremeValue

**Class Hierarchy**

**C++**

```
interface IRobotFeMultiExtremeValue : IDispatch;
```

**C#**

```
public interface IRobotFeMultiExtremeValue;
```

**Visual Basic**

```
Public Interface IRobotFeMultiExtremeValue
```

**Description**

Interface providing access to a great number of min./max. results for finite elements.

**Version**

Available since version 4.1.

### III.5.20.1 IRobotFeMultiExtremeValue Members

The following tables list the members exposed by IRobotFeMultiExtremeValue.

**Public Fields**

	Name	Description
◆	Count (↗ see page 996)	

## Public Methods

	Name	Description
💡	Get (🔗 see page 996)	
💡	GetByResType (🔗 see page 997)	Function returns the indicated min./max. result.

### III.5.20.2 IRobotFeMultiExtremeValue Fields

The fields of the IRobotFeMultiExtremeValue class are listed here.

#### Public Fields

	Name	Description
💡	Count (🔗 see page 996)	

#### III.5.20.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As long
```

##### Version

Available since version 4.1.

### III.5.20.3 IRobotFeMultiExtremeValue Methods

The methods of the IRobotFeMultiExtremeValue class are listed here.

#### Public Methods

	Name	Description
💡	Get (🔗 see page 996)	
💡	GetByResType (🔗 see page 997)	Function returns the indicated min./max. result.

#### III.5.20.3.1 Get

##### C++

```
HRESULT Get(long _idx, IRobotFeExtremeValue** ret);
```

##### C#

```
public IRobotFeExtremeValue Get(long _idx);
```

##### Visual Basic

```
Public Function Get(_idx As long) As IRobotFeExtremeValue
```

##### Version

Available since version 4.1.

#### III.5.20.3.2 GetByResType

##### C++

```
HRESULT GetByResType(IRobotFeResultType _result_type, IRobotFeExtremeValue** ret);
```

**C#**

```
public IRobotFeExtremeValue GetByResType(IRobotFeResultType _result_type);
```

**Visual Basic**

```
Public Function GetByResType(_result_type As IRobotFeResultType) As IRobotFeExtremeValue
```

**Description**

Function returns the indicated min./max. result.

**Version**

Available since version 4.1.

## III.6 IRobotResultServer

**Class Hierarchy****C++**

```
interface IRobotResultServer : IDispatch;
```

**C#**

```
public interface IRobotResultServer;
```

**Visual Basic**

```
Public Interface IRobotResultServer
```

**Description**

The server makes available all types of results obtained by performing structure calculations. .

### III.6.1 IRobotResultServer Members

The following tables list the members exposed by IRobotResultServer.

**Public Fields**

	Name	Description
❖	Advanced (see page 998)	Available since version 1.7.
❖	Any (see page 998)	Access to any calculation result .
❖	Bars (see page 999)	Server of all results for bars .
❖	CalculationResume (see page 999)	.
❖	Extremes (see page 999)	Server providing access to some extreme values .
❖	FiniteElems (see page 1000)	Result server for finite elements .
❖	Nodes (see page 1000)	Server of all results for nodes .
❖	Status (see page 1000)	Results status.
❖	Storeys (see page 1000)	Results for stories.
❖	Total (see page 1001)	Physical properties of a whole structure.

**Public Methods**

	Name	Description
❖	Query (see page 1001)	Function fills out the specified set with results which correspond to the query parameters. Function returns information if all results have been got. To get all results, it is necessary to repeat calling up the function until the value I_RQRT_DONE value is returned.

## III.6.2 IRobotResultServer Fields

The fields of the IRobotResultServer class are listed here.

### Public Fields

	Name	Description
◆	Advanced ( [ see page 998 )	Available since version 1.7.
◆	Any ( [ see page 998 )	Access to any calculation result .
◆	Bars ( [ see page 999 )	Server of all results for bars .
◆	CalculationResume ( [ see page 999 )	.
◆	Extremes ( [ see page 999 )	Server providing access to some extreme values .
◆	FiniteElems ( [ see page 1000 )	Result server for finite elements .
◆	Nodes ( [ see page 1000 )	Server of all results for nodes .
◆	Status ( [ see page 1000 )	Results status.
◆	Storeys ( [ see page 1000 )	Results for stories.
◆	Total ( [ see page 1001 )	Physical properties of a whole structure.

### III.6.2.1 Advanced

#### C++

```
HRESULT get_Advanced(IRobotAdvancedResultServer**);
```

#### C#

```
public IRobotAdvancedResultServer Advanced { get; }
```

#### Visual Basic

```
Public ReadOnly Advanced As IRobotAdvancedResultServer
```

#### Description

Available since version 1.7.

### III.6.2.2 Any

#### C++

```
HRESULT get_Any(IRobotUniversalResultAccess**);
```

#### C#

```
public IRobotUniversalResultAccess Any { get; }
```

#### Visual Basic

```
Public ReadOnly Any As IRobotUniversalResultAccess
```

#### Description

Access to any calculation result .

#### Version

Available since version 3.5.

### III.6.2.3 Bars

#### C++

```
HRESULT get_Bars(IRobotBarResultServer**);
```

**C#**

```
public IRobotBarResultServer Bars { get; }
```

**Visual Basic**

```
Public ReadOnly Bars As IRobotBarResultServer
```

**Description**

Server of all results for bars .

### III.6.2.4 CalculationResume

**C++**

```
HRESULT get_CalculationResume(IRobotCalculationResume**);
```

**C#**

```
public IRobotCalculationResume CalculationResume { get; }
```

**Visual Basic**

```
Public ReadOnly CalculationResume As IRobotCalculationResume
```

**Description****Version**

Available since version 15.

### III.6.2.5 Extremes

**C++**

```
HRESULT get_Extremes(IRobotExtremeResultServer**);
```

**C#**

```
public IRobotExtremeResultServer Extremes { get; }
```

**Visual Basic**

```
Public ReadOnly Extremes As IRobotExtremeResultServer
```

**Description**

Server providing access to some extreme values .

**Version**

Available since version 3.5.

### III.6.2.6 FiniteElems

**C++**

```
HRESULT get_FiniteElems(IRobotFeResultServer**);
```

**C#**

```
public IRobotFeResultServer FiniteElems { get; }
```

**Visual Basic**

```
Public ReadOnly FiniteElems As IRobotFeResultServer
```

**Description**

Result server for finite elements .

**Version**

Available since version 2.5.

**III.6.2.7 Nodes****C++**

```
HRESULT get_Nodes(IRobotNodeResultServer**);
```

**C#**

```
public IRobotNodeResultServer Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As IRobotNodeResultServer
```

**Description**

Server of all results for nodes .

**III.6.2.8 Status****C++**

```
HRESULT get_Status(IRobotResultStatusType*);
```

**C#**

```
public IRobotResultStatusType Status { get; }
```

**Visual Basic**

```
Public ReadOnly Status As IRobotResultStatusType
```

**Description**

Results status.

**Version**

Available since version 14.

**III.6.2.9 Storeys****C++**

```
HRESULT get_Storeys(IRobotStoreyResultServer**);
```

**C#**

```
public IRobotStoreyResultServer Storeys { get; }
```

**Visual Basic**

```
Public ReadOnly Storeys As IRobotStoreyResultServer
```

**Description**

Results for stories.

**Version**

Available since version 9.7.

**III.6.2.10 Total****C++**

```
HRESULT get_Total(IRobotStructureValues**);
```

**C#**

```
public IRobotStructureValues Total { get; }
```

**Visual Basic**

```
Public ReadOnly Total As IRobotStructureValues
```

**Description**

Physical properties of a whole structure.

**Version**

Available since version 7.5.

### III.6.3 IRobotResultServer Methods

The methods of the IRobotResultServer class are listed here.

**Public Methods**

	Name	Description
💡	Query (see page 1001)	Function fills out the specified set with results which correspond to the query parameters. Function returns information if all results have been got. To get all results, it is necessary to repeat calling up the function until the value I_RQRT_DONE value is returned.

#### III.6.3.1 Query

**C++**

```
HRESULT Query(IRobotResultQueryParams* _params, IRobotResultRowSet* _ret_results,
IRobotResultQueryReturnType* ret);
```

**C#**

```
public IRobotResultQueryReturnType Query(IRobotResultQueryParams _params,
IRobotResultRowSet _ret_results);
```

**Visual Basic**

```
Public Function Query(ByRef _params As IRobotResultQueryParams, ByRef _ret_results As
IRobotResultRowSet) As IRobotResultQueryReturnType
```

**Description**

Function fills out the specified set with results which correspond to the query parameters. Function returns information if all results have been got. To get all results, it is necessary to repeat calling up the function until the value I\_RQRT\_DONE value is returned.

**Version**

Available since version 8.2.

### III.7 IRobotUniversalResultAccess

**Class Hierarchy****C++**

```
interface IRobotUniversalResultAccess : IDispatch;
```

**C#**

```
public interface IRobotUniversalResultAccess;
```

## Visual Basic

```
Public Interface IRobotUniversalResultAccess
```

### Description

Universal access interface for calculation results. In order to obtain the relevant result the object should be first appropriately parametrized by determining what result type, for which load case and for which structure element is to be obtained, and then ResultValue (see page 1009) should be taken. .

### III.7.1 IRobotUniversalResultAccess Members

The following tables list the members exposed by IRobotUniversalResultAccess.

#### Public Fields

	Name	Description
◆	Available (see page 1004)	Flag indicating if the result for defined parameters can be obtained .
◆	Bar (see page 1004)	Bar number for which the result is to be taken .
◆	CalcPoint (see page 1004)	
◆	DivCount (see page 1005)	Number of bar division points (only for the results obtained in the defined point of bar division) .
◆	DivPoint (see page 1005)	Number of bar division point (only for the results obtained in the defined point of bar division) .
◆	Element (see page 1005)	Number of the finite element .
◆	Layer (see page 1005)	Layer.
◆	LayerArbitraryValue (see page 1006)	Layer (see page 1005) thickness.
◆	LinearSupports (see page 1006)	Linear supports.
◆	LoadCase (see page 1006)	Number of the load case for which the result is to be taken .
◆	LoadCaseCmpnt (see page 1007)	Number of the load case component (significant for cases including more than one component, e.g. code combination) .
◆	Mode (see page 1007)	Number of the load case mode (significant for cases for which many modes are generated) .
◆	ModeCmb (see page 1007)	Type of mode combination .
◆	Node (see page 1007)	Node number for which the result is to be taken .
◆	Panel (see page 1008)	Panel number.
◆	ReducedCutPos (see page 1008)	Position of the panel cut (for reduced results).
◆	ReinforceCalcMethod (see page 1008)	Calculation method for reinforcement .
◆	RelativePoint (see page 1009)	Relative point on the bar (significant only for the results obtained in the defined bar point) .
◆	ResultId (see page 1009)	Identifier determining the result type which is to be taken .
◆	ResultType (see page 1009)	Result type.
◆	ResultValue (see page 1009)	Result value for defined parameters (provided that the Available (see page 1004) flag assumes value different from zero - True) .
◆	Storey (see page 1010)	Story name.

#### Public Methods

	Name	Description
◆	GetDirX (see page 1010)	Function takes direction of X axis of the local coordinate system in which results will be presented. The values read are the coordinates of the direction vector of X axis for definition in the Cartesian system or the coordinates of the central point for definition in the Polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the local panel system. In addition, the function returns a type of system direction definition. .

	Reset ( <a href="#">see page 1011</a> )	Function sets as zero all the settings concerning the result type and the manner of its taking. .
	ResultValue3D ( <a href="#">see page 1011</a> )	Function returns a result in a form of a triple of numbers.
	SetDirX ( <a href="#">see page 1011</a> )	Function allows defining direction of X axis of the system in which results will be presented. The function should be activated only when the vector of detailed results is being taken. Function parameters determine coordinates of the direction vector of X axis in the Cartesian system or coordinates of the central point in the Polar system. If all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined. .

### III.7.2 IRobotUniversalResultAccess Fields

The fields of the IRobotUniversalResultAccess class are listed here.

#### Public Fields

	Name	Description
	Available ( <a href="#">see page 1004</a> )	Flag indicating if the result for defined parameters can be obtained .
	Bar ( <a href="#">see page 1004</a> )	Bar number for which the result is to be taken .
	CalcPoint ( <a href="#">see page 1004</a> )	
	DivCount ( <a href="#">see page 1005</a> )	Number of bar division points (only for the results obtained in the defined point of bar division) .
	DivPoint ( <a href="#">see page 1005</a> )	Number of bar division point (only for the results obtained in the defined point of bar division) .
	Element ( <a href="#">see page 1005</a> )	Number of the finite element .
	Layer ( <a href="#">see page 1005</a> )	Layer.
	LayerArbitraryValue ( <a href="#">see page 1006</a> )	Layer ( <a href="#">see page 1005</a> ) thickness.
	LinearSupports ( <a href="#">see page 1006</a> )	Linear supports.
	LoadCase ( <a href="#">see page 1006</a> )	Number of the load case for which the result is to be taken .
	LoadCaseCmpnt ( <a href="#">see page 1007</a> )	Number of the load case component (significant for cases including more than one component, e.g. code combination) .
	Mode ( <a href="#">see page 1007</a> )	Number of the load case mode (significant for cases for which many modes are generated) .
	ModeCmb ( <a href="#">see page 1007</a> )	Type of mode combination .
	Node ( <a href="#">see page 1007</a> )	Node number for which the result is to be taken .
	Panel ( <a href="#">see page 1008</a> )	Panel number.
	ReducedCutPos ( <a href="#">see page 1008</a> )	Position of the panel cut (for reduced results).
	ReinforceCalcMethod ( <a href="#">see page 1008</a> )	Calculation method for reinforcement .
	RelativePoint ( <a href="#">see page 1009</a> )	Relative point on the bar (significant only for the results obtained in the defined bar point) .
	ResultId ( <a href="#">see page 1009</a> )	Identifier determining the result type which is to be taken .
	ResultType ( <a href="#">see page 1009</a> )	Result type.
	ResultValue ( <a href="#">see page 1009</a> )	Result value for defined parameters (provided that the Available ( <a href="#">see page 1004</a> ) flag assumes value different from zero - True) .
	Storey ( <a href="#">see page 1010</a> )	Story name.

#### III.7.2.1 Available

##### C++

```
HRESULT get_Available(VARIANT_BOOL* );
```

##### C#

```
public bool Available { get; }
```

**Visual Basic**

```
Public ReadOnly Available As Boolean
```

**Description**

Flag indicating if the result for defined parameters can be obtained .

**III.7.2.2 Bar****C++**

```
HRESULT get_Bar(long* );
HRESULT put_Bar(long);
```

**C#**

```
public long Bar { get; set; }
```

**Visual Basic**

```
Public Bar As long
```

**Description**

Bar number for which the result is to be taken .

**III.7.2.3 CalcPoint****C++**

```
HRESULT get_CalcPoint(long* );
HRESULT put_CalcPoint(long);
```

**C#**

```
public long CalcPoint { get; set; }
```

**Visual Basic**

```
Public CalcPoint As long
```

**Version**

Available (see page 1004) since version 8.

**III.7.2.4 DivCount****C++**

```
HRESULT get_DivCount(long* );
HRESULT put_DivCount(long);
```

**C#**

```
public long DivCount { get; set; }
```

**Visual Basic**

```
Public DivCount As long
```

**Description**

Number of bar division points (only for the results obtained in the defined point of bar division) .

**III.7.2.5 DivPoint****C++**

```
HRESULT get_DivPoint(long* );
HRESULT put_DivPoint(long);
```

**C#**

```
public long DivPoint { get; set; }
```

**Visual Basic**

```
Public DivPoint As Long
```

**Description**

Number of bar division point (only for the results obtained in the defined point of bar division) .

**III.7.2.6 Element****C++**

```
HRESULT get_Element(long* );
HRESULT put_Element(long);
```

**C#**

```
public long Element { get; set; }
```

**Visual Basic**

```
Public Element As Long
```

**Description**

Number of the finite element .

**Version**

Available (see page 1004) since version 3.5.

**III.7.2.7 Layer****C++**

```
HRESULT get_Layer(IRobotFeLayerType* );
HRESULT put_Layer(IRobotFeLayerType);
```

**C#**

```
public IRobotFeLayerType Layer { get; set; }
```

**Visual Basic**

```
Public Layer As IRobotFeLayerType
```

**Description**

Layer.

**Version**

Available (see page 1004) since version 3.5.

**III.7.2.8 LayerArbitraryValue****C++**

```
HRESULT get_LayerArbitraryValue(double* );
HRESULT put_LayerArbitraryValue(double);
```

**C#**

```
public double LayerArbitraryValue { get; set; }
```

**Visual Basic**

```
Public LayerArbitraryValue As Double
```

**Description**

Layer (see page 1005) thickness.

**Version**

Available (see page 1004) since version 3.5.

### III.7.2.9 LinearSupports

**C++**

```
HRESULT get_LinearSupports(VARIANT_BOOL* );
HRESULT put_LinearSupports(VARIANT_BOOL);
```

**C#**

```
public bool LinearSupports { get; set; }
```

**Visual Basic**

```
Public LinearSupports As Boolean
```

**Description**

Linear supports.

**Version**

Available (see page 1004) since version 8.1.

### III.7.2.10 LoadCase

**C++**

```
HRESULT get_LoadCase(long* );
HRESULT put_LoadCase(long);
```

**C#**

```
public long LoadCase { get; set; }
```

**Visual Basic**

```
Public LoadCase As long
```

**Description**

Number of the load case for which the result is to be taken .

### III.7.2.11 LoadCaseCmpnt

**C++**

```
HRESULT get_LoadCaseCmpnt(long* );
HRESULT put_LoadCaseCmpnt(long);
```

**C#**

```
public long LoadCaseCmpnt { get; set; }
```

**Visual Basic**

```
Public LoadCaseCmpnt As long
```

**Description**

Number of the load case component (significant for cases including more than one component, e.g. code combination) .

### III.7.2.12 Mode

#### C++

```
HRESULT get_Mode(long*);  
HRESULT put_Mode(long);
```

#### C#

```
public long Mode { get; set; }
```

#### Visual Basic

```
Public Mode As long
```

#### Description

Number of the load case mode (significant for cases for which many modes are generated).

### III.7.2.13 ModeCmb

#### C++

```
HRESULT get_ModeCmb(IRobotModeCombinationType*);  
HRESULT put_ModeCmb(IRobotModeCombinationType);
```

#### C#

```
public IRobotModeCombinationType ModeCmb { get; set; }
```

#### Visual Basic

```
Public ModeCmb As IRobotModeCombinationType
```

#### Description

Type of mode combination.

### III.7.2.14 Node

#### C++

```
HRESULT get_Node(long*);  
HRESULT put_Node(long);
```

#### C#

```
public long Node { get; set; }
```

#### Visual Basic

```
Public Node As long
```

#### Description

Node number for which the result is to be taken.

### III.7.2.15 Panel

#### C++

```
HRESULT get_Panel(long*);  
HRESULT put_Panel(long);
```

#### C#

```
public long Panel { get; set; }
```

#### Visual Basic

```
Public Panel As long
```

**Description**

Panel number.

**Version**

Available (see page 1004) since version 3.5.

**III.7.2.16 ReducedCutPos****C++**

```
HRESULT get_ReducedCutPos(IRobotFeResultReducedCutPosition* );
HRESULT put_ReducedCutPos(IRobotFeResultReducedCutPosition);
```

**C#**

```
public IRobotFeResultReducedCutPosition ReducedCutPos { get; set; }
```

**Visual Basic**

```
Public ReducedCutPos As IRobotFeResultReducedCutPosition
```

**Description**

Position of the panel cut (for reduced results).

**Version**

Available (see page 1004) since version 3.5.

**III.7.2.17 ReinforceCalcMethod****C++**

```
HRESULT get_ReinforceCalcMethod(IRobotReinforceCalcMethods* );
HRESULT put_ReinforceCalcMethod(IRobotReinforceCalcMethods);
```

**C#**

```
public IRobotReinforceCalcMethods ReinforceCalcMethod { get; set; }
```

**Visual Basic**

```
Public ReinforceCalcMethod As IRobotReinforceCalcMethods
```

**Description**

Calculation method for reinforcement .

**Version**

Available (see page 1004) since version 3.5.

**III.7.2.18 RelativePoint****C++**

```
HRESULT get_RelativePoint(double* );
HRESULT put_RelativePoint(double);
```

**C#**

```
public double RelativePoint { get; set; }
```

**Visual Basic**

```
Public RelativePoint As Double
```

**Description**

Relative point on the bar (significant only for the results obtained in the defined bar point) .

### III.7.2.19 ResultId

#### C++

```
HRESULT get_ResultId(long* );
HRESULT put_ResultId(long);
```

#### C#

```
public long ResultId { get; set; }
```

#### Visual Basic

```
Public ResultId As long
```

#### Description

Identifier determining the result type which is to be taken .

### III.7.2.20 ResultType

#### C++

```
HRESULT get_ResultType(IRobotUniversalResultType* );
```

#### C#

```
public IRobotUniversalResultType ResultType { get; }
```

#### Visual Basic

```
Public ReadOnly ResultType As IRobotUniversalResultType
```

#### Description

Result type.

### III.7.2.21 ResultValue

#### C++

```
HRESULT get_ResultValue(double* );
```

#### C#

```
public double ResultValue { get; }
```

#### Visual Basic

```
Public ReadOnly ResultValue As double
```

#### Description

Result value for defined parameters (provided that the Available ( see page 1004) flag assumes value different from zero - True) .

### III.7.2.22 Storey

#### C++

```
HRESULT get_Storey(BSTR* );
HRESULT put_Storey(BSTR);
```

#### C#

```
public String Storey { get; set; }
```

#### Visual Basic

```
Public Storey As String
```

## Description

Story name.

### III.7.3 IRobotUniversalResultAccess Methods

The methods of the IRobotUniversalResultAccess class are listed here.

#### Public Methods

	Name	Description
✳	GetDirX (see page 1010)	Function takes direction of X axis of the local coordinate system in which results will be presented. The values read are the coordinates of the direction vector of X axis for definition in the Cartesian system or the coordinates of the central point for definition in the Polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the local panel system. In addition, the function returns a type of system direction definition. .
✳	Reset (see page 1011)	Function sets as zero all the settings concerning the result type and the manner of its taking. .
✳	ResultValue3D (see page 1011)	Function returns a result in a form of a triple of numbers.
✳	SetDirX (see page 1011)	Function allows defining direction of X axis of the system in which results will be presented. The function should be activated only when the vector of detailed results is being taken. Function parameters determine coordinates of the direction vector of X axis in the Cartesian system or coordinates of the central point in the Polar system. If all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined..

#### III.7.3.1 GetDirX

##### C++

```
HRESULT GetDirX(double* _x, double* _y, double* _z, IRobotObjLocalXDirDefinitionType* ret);
```

##### C#

```
public IRobotObjLocalXDirDefinitionType GetDirX(double* _x, double* _y, double* _z);
```

##### Visual Basic

```
Public Function GetDirX(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*) As IRobotObjLocalXDirDefinitionType
```

#### Description

Function takes direction of X axis of the local coordinate system in which results will be presented. The values read are the coordinates of the direction vector of X axis for definition in the Cartesian system or the coordinates of the central point for definition in the Polar system. If values of all the arguments: x, y, z are set as zeros, then the system direction is compatible with the local panel system. In addition, the function returns a type of system direction definition. .

#### Version

Available (see page 1004) since version 3.5.

### III.7.3.2 Reset

##### C++

```
HRESULT Reset();
```

##### C#

```
public void Reset();
```

**Visual Basic**

```
Public Sub Reset()
```

**Description**

Function sets as zero all the settings concerning the result type and the manner of its taking. .

**III.7.3.3 ResultValue3D****C++**

```
HRESULT ResultValue3D(double* _val1, double* _val2, double* _val3);
```

**C#**

```
public void ResultValue3D(double* _val1, double* _val2, double* _val3);
```

**Visual Basic**

```
Public Sub ResultValue3D(ByRef _val1 As double*, ByRef _val2 As double*, ByRef _val3 As double*)
```

**Description**

Function returns a result in a form of a triple of numbers.

**III.7.3.4 SetDirX****C++**

```
HRESULT SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

**C#**

```
public void SetDirX(IRobotObjLocalXDirDefinitionType _def_type, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetDirX(_def_type As IRobotObjLocalXDirDefinitionType, _x As double, _y As double, _z As double)
```

**Description**

Function allows defining direction of X axis of the system in which results will be presented. The function should be activated only when the vector of detailed results is being taken. Function parameters determine coordinates of the direction vector of X axis in the Cartesian system or coordinates of the central point in the Polar system. If all the values: x, y, z are set as zeros, a direction compatible with the local system of the selected panel will be defined. .

**Version**

Available (see page 1004) since version 3.5.

**III.8 IRobotExtremeResultServer****Class Hierarchy****C++**

```
interface IRobotExtremeResultServer : IDispatch;
```

**C#**

```
public interface IRobotExtremeResultServer;
```

**Visual Basic**

```
Public Interface IRobotExtremeResultServer
```

**Description**

Server of extreme results.

**Version**

Available since version 3.5.

**III.8.1 IRobotExtremeResultServer Members**

The following tables list the members exposed by IRobotExtremeResultServer.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	MaxValue (see page 1012)	Function returns the maximal value calculated for the parameters given. .
💡	MinValue (see page 1013)	Function returns the minimal value calculated for the parameters given. .

**III.8.2 IRobotExtremeResultServer Methods**

The methods of the IRobotExtremeResultServer class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	MaxValue (see page 1012)	Function returns the maximal value calculated for the parameters given. .
💡	MinValue (see page 1013)	Function returns the minimal value calculated for the parameters given. .

**III.8.2.1 MaxValue****C++**

```
HRESULT MaxValue(IRobotExtremeParams* __params, IRobotExtremeValue** ret);
```

**C#**

```
public IRobotExtremeValue MaxValue(IRobotExtremeParams __params);
```

**Visual Basic**

```
Public Function MaxValue(ByRef __params As IRobotExtremeParams) As IRobotExtremeValue
```

**Description**

Function returns the maximal value calculated for the parameters given. .

**Version**

Available since version 3.5.

**III.8.2.2 MinValue****C++**

```
HRESULT MinValue(IRobotExtremeParams* __params, IRobotExtremeValue** ret);
```

**C#**

```
public IRobotExtremeValue MinValue(IRobotExtremeParams __params);
```

**Visual Basic**

```
Public Function MinValue(ByRef __params As IRobotExtremeParams) As IRobotExtremeValue
```

**Description**

Function returns the minimal value calculated for the parameters given. .

**Version**

Available since version 3.5.

## III.9 IRobotUniversalResultType

**C++**

```
enum IRobotUniversalResultType;
```

**C#**

```
public enum IRobotUniversalResultType;
```

**Visual Basic**

```
Public Enum IRobotUniversalResultType
```

**Members**

Members	Description
I_URT_VALUE = 1	Result has a form of a single value.
I_URT_VALUE_3D = 2	Result has a form of a triple of numbers.

**Description**

Types of results made available by the RobotUniversalResultAccess object.

## III.10 IRobotStructureValues

**Class Hierarchy****C++**

```
interface IRobotStructureValues : IDispatch;
```

**C#**

```
public interface IRobotStructureValues;
```

**Visual Basic**

```
Public Interface IRobotStructureValues
```

**Description**

Physical properties for the structure model.

**Version**

Available since version 7.5.

### III.10.1 IRobotStructureValues Members

The following tables list the members exposed by IRobotStructureValues.

**Public Methods**

	Name	Description
	GetEx2 (see page 1015)	Function returns additional eccentricity according to X axis for the given load case.

	GetEy2 (see page 1015)	Function returns additional eccentricity according to Y axis for the given load case.
	GetG (see page 1015)	Function returns coordinates of structure center of gravity for the given load case.
	GetIx (see page 1016)	Function returns mass moment of inertia according to X axis for the given load case.
	GetIy (see page 1016)	Function returns mass moment of inertia according to Y axis for the given load case.
	GetIz (see page 1016)	Function returns mass moment of inertia according to Z axis for the given load case.
	GetMass (see page 1017)	Function returns a mass of the whole structure for the given load case.
	GetT (see page 1017)	Function returns coordinates of structure center of torsion for the given load case.

### III.10.2 IRobotStructureValues Methods

The methods of the IRobotStructureValues class are listed here.

#### Public Methods

	Name	Description
	GetEx2 (see page 1015)	Function returns additional eccentricity according to X axis for the given load case.
	GetEy2 (see page 1015)	Function returns additional eccentricity according to Y axis for the given load case.
	GetG (see page 1015)	Function returns coordinates of structure center of gravity for the given load case.
	GetIx (see page 1016)	Function returns mass moment of inertia according to X axis for the given load case.
	GetIy (see page 1016)	Function returns mass moment of inertia according to Y axis for the given load case.
	GetIz (see page 1016)	Function returns mass moment of inertia according to Z axis for the given load case.
	GetMass (see page 1017)	Function returns a mass of the whole structure for the given load case.
	GetT (see page 1017)	Function returns coordinates of structure center of torsion for the given load case.

#### III.10.2.1 GetEx2

##### C++

```
HRESULT GetEx2(long _case_num, double* ret);
```

##### C#

```
public double GetEx2(long _case_num);
```

##### Visual Basic

```
Public Function GetEx2(_case_num As long) As double
```

##### Description

Function returns additional eccentricity according to X axis for the given load case.

##### Version

Available since version 7.5.

### III.10.2.2 GetEy2

#### C++

```
HRESULT GetEy2(long _case_num, double* ret);
```

#### C#

```
public double GetEy2(long _case_num);
```

#### Visual Basic

```
Public Function GetEy2(_case_num As long) As double
```

#### Description

Function returns additional eccentricity according to Y axis for the given load case.

#### Version

Available since version 7.5.

### III.10.2.3 GetG

#### C++

```
HRESULT GetG(long _case_num, IRobotGeoPoint3D** ret);
```

#### C#

```
public IRobotGeoPoint3D GetG(long _case_num);
```

#### Visual Basic

```
Public Function GetG(_case_num As long) As IRobotGeoPoint3D
```

#### Description

Function returns coordinates of structure center of gravity for the given load case.

#### Version

Available since version 7.5.

### III.10.2.4 GetIx

#### C++

```
HRESULT GetIx(long _case_num, double* ret);
```

#### C#

```
public double GetIx(long _case_num);
```

#### Visual Basic

```
Public Function GetIx(_case_num As long) As double
```

#### Description

Function returns mass moment of inertia according to X axis for the given load case.

#### Version

Available since version 7.5.

### III.10.2.5 GetIy

#### C++

```
HRESULT GetIy(long _case_num, double* ret);
```

#### C#

```
public double GetIy(long _case_num);
```

#### Visual Basic

```
Public Function GetIy(_case_num As long) As double
```

#### Description

Function returns mass moment of inertia according to Y axis for the given load case.

#### Version

Available since version 7.5.

### III.10.2.6 GetIz

#### C++

```
HRESULT GetIz(long _case_num, double* ret);
```

#### C#

```
public double GetIz(long _case_num);
```

#### Visual Basic

```
Public Function GetIz(_case_num As long) As double
```

#### Description

Function returns mass moment of inertia according to Z axis for the given load case.

#### Version

Available since version 7.5.

### III.10.2.7 GetMass

#### C++

```
HRESULT GetMass(long _case_num, double* ret);
```

#### C#

```
public double GetMass(long _case_num);
```

#### Visual Basic

```
Public Function GetMass(_case_num As long) As double
```

#### Description

Function returns a mass of the whole structure for the given load case.

#### Version

Available since version 7.5.

### III.10.2.8 GetT

#### C++

```
HRESULT GetT(long _case_num, IRobotGeoPoint3D** ret);
```

#### C#

```
public IRobotGeoPoint3D GetT(long _case_num);
```

#### Visual Basic

```
Public Function GetT(_case_num As long) As IRobotGeoPoint3D
```

#### Description

Function returns coordinates of structure center of torsion for the given load case.

#### Version

Available since version 7.5.

## III.11 IRobotResultRow

#### Class Hierarchy

#### C++

```
interface IRobotResultRow : IDispatch;
```

#### C#

```
public interface IRobotResultRow;
```

#### Visual Basic

```
Public Interface IRobotResultRow
```

#### Description

Row of results calculated for determined values of a parameter set.

#### Version

Available since version 8.2.

### III.11.1 IRobotResultRow Members

The following tables list the members exposed by IRobotResultRow.

#### Public Methods

	Name	Description
💡	GetParam (see page 1018)	Function returns values of determined parameters for which available results have been calculated.
💡	GetValue (see page 1018)	Function returns a value of the specified result.
💡	GetValueType (see page 1019)	Function returns a type of the value representing a specified result.
💡	IsAvailable (see page 1019)	Function checks if a specified result is available.

### III.11.2 IRobotResultRow Methods

The methods of the IRobotResultRow class are listed here.

## Public Methods

	Name	Description
⌚	GetParam (see page 1018)	Function returns values of determined parameters for which available results have been calculated.
⌚	GetValue (see page 1018)	Function returns a value of the specified result.
⌚	GetValueType (see page 1019)	Function returns a type of the value representing a specified result.
⌚	IsAvailable (see page 1019)	Function checks if a specified result is available.

### III.11.2.1 GetParam

#### C++

```
HRESULT GetParam(IRobotResultParamType _result_param_id, VARIANT* ret);
```

#### C#

```
public VARIANT GetParam(IRobotResultParamType _result_param_id);
```

#### Visual Basic

```
Public Function GetParam(_result_param_id As IRobotResultParamType) As VARIANT
```

#### Description

Function returns values of determined parameters for which available results have been calculated.

#### Version

Available since version 8.2.

### III.11.2.2 GetValue

#### C++

```
HRESULT GetValue(long _result_id, VARIANT* ret);
```

#### C#

```
public VARIANT GetValue(long _result_id);
```

#### Visual Basic

```
Public Function GetValue(_result_id As long) As VARIANT
```

#### Description

Function returns a value of the specified result.

#### Version

Available since version 8.2.

### III.11.2.3 GetValueType

#### C++

```
HRESULT GetValueType(long _result_id, IRobotUniversalResultType* ret);
```

#### C#

```
public IRobotUniversalResultType GetValueType(long _result_id);
```

#### Visual Basic

```
Public Function GetValueType(_result_id As long) As IRobotUniversalResultType
```

**Description**

Function returns a type of the value representing a specified result.

**Version**

Available since version 8.2.

**III.11.2.4 IsAvailable****C++**

```
HRESULT IsAvailable(long _result_id, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsAvailable(long _result_id);
```

**Visual Basic**

```
Public Function IsAvailable(_result_id As long) As Boolean
```

**Description**

Function checks if a specified result is available.

**Version**

Available since version 8.2.

**III.12 IRobotResultParamType****C++**

```
enum IRobotResultParamType;
```

**C#**

```
public enum IRobotResultParamType;
```

**Visual Basic**

```
Public Enum IRobotResultParamType
```

**Members**

<b>Members</b>	<b>Description</b>
I_RPT_LOAD_CASE = 1	Load case no. Available since version 8.2.
I_RPT_LOAD_CASE_CMPNT = 2	Load case component no. Available since version 8.2.
I_RPT_NODE = 3	Node no. Available since version 8.2.
I_RPT_BAR = 4	Bar no. Available since version 8.2.
I_RPT_BAR_DIV_COUNT = 5	Number of bar division points. Available since version 8.2.
I_RPT_BAR_DIV_POINT = 6	Bar division point no. Available since version 8.2.
I_RPT_BAR_RELATIVE_POINT = 7	Relative point position on the bar. Available since version 8.2.
I_RPT_MODE = 8	Load case mode no. Available since version 8.2.

I_RPT_MODE_CMB = 9	Mode combination type (assumes values of the type IRobotModeCombinationType (see page 1035)). Available since version 8.2.
I_RPT_PANEL = 10	Panel no. Available since version 8.2.
I_RPT_ELEMENT = 11	Finite element no. Available since version 8.2.
I_RPT_LAYER = 12	Layer designation (assumes values of the type IRobotFeLayerType (see page 932)). Available since version 8.2.
I_RPT_LAYER_ARBITRARY_VALUE = 13	Layer thickness. Available since version 8.2.
I_RPT_REINFORCE_CALC_METHOD = 14	Calculation method for reinforcement (assumes values of the type IRobotReinforceCalcMethods (see page 951)). Available since version 8.2.
I_RPT_REDUCED_CUT_POS = 15	Position of the cut through the panel for reduced results (assumes values of the type IRobotFeResultReducedCutPosition (see page 972)). Available since version 8.2.
I_RPT_STOREY = 16	Story name. Available since version 8.2.
I_RPT_CALC_POINT = 17	Available since version 8.2.
I_RPT_LINEAR_SUPPORTS = 18	Available since version 8.2.
I_RPT_DIR_X = 19	Direction of the X axis of the system in which results are presented - the table of three real numbers. Available since version 8.2.
I_RPT_PANEL_PART = 20	Panel component index. Available since version 8.2.
I_RPT_MAX_BUFFER_SIZE = 21	Maximal buffer size in bytes. Available since version 8.2.
I_RPT_MULTI_THREADS = 22	Flag indicating multithread access to results. Available since version 8.2.
I_RPT_BUFFER_CHUNK_SIZE = 23	Size of a single buffer fragment in bytes. Available since version 8.2.
I_RPT_THREAD_COUNT = 24	Number of threads which should be used for getting results. Available since version 8.2.
I_RPT_DIR_X_DEFTYPE = 25	Method of defining the X axis direction - value of the type IRobotObjLocalXDirDefinitionType (see page 788). Available since version 8.2.
I_RPT_SMOOTHING = 26	Method of result smoothing (values of the calculation type IRobotFeResultSmoothing (see page 980)). Available since version 8.2.
I_RPT_ITERATE_LOAD_CASES = 27	Parameter steering with getting the results; for results which do not depend on a load case the False value should be set. Available since version 8.7.
I_RPT_BAR_ELEMENT = 28	Available since version 12.
I_RPT_BAR_ELEMENT_DIV_COUNT = 29	Available since version 12.
I_RPT_BAR_ELEMENT_DIV_POINT = 30	Available since version 12.
I_RPT_RESULT_POINT_COORDINATES = 31	Set this on query to retrieve global coordinates of points where results are provided. Available since version 12.

I_RPT_BAR_ELEMENT_DIV_DISCONTINUITY = 32	Set this 1 (True) to offset division points on ends of an element in the case of discontinuity. The number of division points can be increased internally, so I_RPT_BAR_ELEMENT_DIV_COUNT should be read while reading results for a given element. Available since version 12.
I_RPT_BAR_ELEMENT_DIV_DISCONTINUITY_OFFSET = 33	Set a value for the relative offset used to shift the point of discontinuity, if you want it to be different from the default. Available since version 12.

**Description**

Set of parameters describing calculation results.

**Version**

Available since version 8.2.

## III.13 IRobotResultRowSet

**Class Hierarchy****C++**

```
interface IRobotResultRowSet : IDispatch;
```

**C#**

```
public interface IRobotResultRowSet;
```

**Visual Basic**

```
Public Interface IRobotResultRowSet
```

**Description**

Set of rows with calculation results.

**Version**

Available since version 8.2.

### III.13.1 IRobotResultRowSet Members

The following tables list the members exposed by IRobotResultRowSet.

**Public Fields**

	Name	Description
❖	CurrentRow (see page 1022)	Current row of the result set.
❖	ResultIds (see page 1022)	Collection of result identifiers available in the set.

**Public Methods**

	Name	Description
❖	Clear (see page 1023)	
❖	MoveFirst (see page 1023)	Function sets the current row on the first position in the set. If the set is blank, the function returns the False value.
❖	MoveNext (see page 1023)	Function moves the current row to the next position in the set. If the current row has been set earlier on the last position, or the set is blank, the function returns the False value.

### III.13.2 IRobotResultRowSet Fields

The fields of the IRobotResultRowSet class are listed here.

## Public Fields

	Name	Description
◆	CurrentRow (see page 1022)	Current row of the result set.
◆	ResultIds (see page 1022)	Collection of result identifiers available in the set.

### III.13.2.1 CurrentRow

#### C++

```
HRESULT get_CurrentRow(IRobotResultRow**);
```

#### C#

```
public IRobotResultRow CurrentRow { get; }
```

#### Visual Basic

```
Public ReadOnly CurrentRow As IRobotResultRow
```

#### Description

Current row of the result set.

#### Version

Available since version 8.2.

### III.13.2.2 ResultIds

#### C++

```
HRESULT get_ResultIds(IRobotNumbersCollection**);
```

#### C#

```
public IRobotNumbersCollection ResultIds { get; }
```

#### Visual Basic

```
Public ReadOnly ResultIds As IRobotNumbersCollection
```

#### Description

Collection of result identifiers available in the set.

#### Version

Available since version 8.2.

## III.13.3 IRobotResultRowSet Methods

The methods of the IRobotResultRowSet class are listed here.

### Public Methods

	Name	Description
◆	Clear (see page 1023)	
◆	MoveFirst (see page 1023)	Function sets the current row on the first position in the set. If the set is blank, the function returns the False value.
◆	MoveNext (see page 1023)	Function moves the current row to the next position in the set. If the current row has been set earlier on the last position, or the set is blank, the function returns the False value.

### III.13.3.1 Clear

#### C++

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Version**

Available since version 8.2.

### III.13.3.2 MoveFirst

**C++**

```
HRESULT MoveFirst(VARIANT_BOOL* ret);
```

**C#**

```
public bool MoveFirst();
```

**Visual Basic**

```
Public Function MoveFirst() As Boolean
```

**Description**

Function sets the current row on the first position in the set. If the set is blank, the function returns the False value.

**Version**

Available since version 8.2.

### III.13.3.3 MoveNext

**C++**

```
HRESULT MoveNext(VARIANT_BOOL* ret);
```

**C#**

```
public bool MoveNext();
```

**Visual Basic**

```
Public Function MoveNext() As Boolean
```

**Description**

Function moves the current row to the next position in the set. If the current row has been set earlier on the last position, or the set is blank, the function returns the False value.

**Version**

Available since version 8.2.

## III.14 IRobotResultQueryParams

**Class Hierarchy****C++**

```
interface IRobotResultQueryParams : IDispatch;
```

**C#**

```
public interface IRobotResultQueryParams;
```

**Visual Basic**

```
Public Interface IRobotResultQueryParams
```

**Description**

Definition of a query to the result server.

**Version**

Available since version 8.2.

**III.14.1 IRobotResultQueryParams Members**

The following tables list the members exposed by IRobotResultQueryParams.

**Public Fields**

	Name	Description
❖	ResultIds ( <a href="#">see page 1025</a> )	Table of identifiers of results to be got.
❖	Selection ( <a href="#">see page 1025</a> )	Selection of structure components for which results should be got.

**Public Methods**

	Name	Description
❖	GetParam ( <a href="#">see page 1026</a> )	Function returns a value of the determined parameter.
❖	IsParamSet ( <a href="#">see page 1026</a> )	Function allows checking if a value has been set for a given parameter.
❖	Reset ( <a href="#">see page 1026</a> )	Function assigns zero values to all attributes of the query.
❖	SetParam ( <a href="#">see page 1027</a> )	Function sets a determined value of the specified parameter.

**III.14.2 IRobotResultQueryParams Fields**

The fields of the IRobotResultQueryParams class are listed here.

**Public Fields**

	Name	Description
❖	ResultIds ( <a href="#">see page 1025</a> )	Table of identifiers of results to be got.
❖	Selection ( <a href="#">see page 1025</a> )	Selection of structure components for which results should be got.

**III.14.2.1 ResultIds****C++**

```
HRESULT get_ResultIds( IRobotNumbersArray** );
```

**C#**

```
public IRobotNumbersArray ResultIds { get; }
```

**Visual Basic**

```
Public ReadOnly ResultIds As IRobotNumbersArray
```

**Description**

Table of identifiers of results to be got.

**Version**

Available since version 8.2.

**III.14.2.2 Selection****C++**

```
HRESULT get_Selection( IRobotMultiSelection** );
```

**C#**

```
public IRobotMultiSelection Selection { get; }
```

**Visual Basic**

```
Public ReadOnly Selection As IRobotMultiSelection
```

**Description**

Selection of structure components for which results should be got.

**Version**

Available since version 8.2.

**III.14.3 IRobotResultQueryParams Methods**

The methods of the IRobotResultQueryParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	GetParam ( <a href="#">see page 1026</a> )	Function returns a value of the determined parameter.
≡	IsParamSet ( <a href="#">see page 1026</a> )	Function allows checking if a value has been set for a given parameter.
≡	Reset ( <a href="#">see page 1026</a> )	Function assigns zero values to all attributes of the query.
≡	SetParam ( <a href="#">see page 1027</a> )	Function sets a determined value of the specified parameter.

**III.14.3.1 GetParam****C++**

```
HRESULT GetParam( IRobotResultParamType _param_id, VARIANT* ret);
```

**C#**

```
public VARIANT GetParam( IRobotResultParamType _param_id);
```

**Visual Basic**

```
Public Function GetParam(_param_id As IRobotResultParamType) As VARIANT
```

**Description**

Function returns a value of the determined parameter.

**Version**

Available since version 8.2.

**III.14.3.2 IsParamSet****C++**

```
HRESULT IsParamSet( IRobotResultParamType _param_id, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsParamSet( IRobotResultParamType _param_id);
```

**Visual Basic**

```
Public Function IsParamSet(_param_id As IRobotResultParamType) As Boolean
```

**Description**

Function allows checking if a value has been set for a given parameter.

**Version**

Available since version 8.2.

**III.14.3.3 Reset****C++**

```
HRESULT Reset();
```

**C#**

```
public void Reset();
```

**Visual Basic**

```
Public Sub Reset()
```

**Description**

Function assigns zero values to all attributes of the query.

**Version**

Available since version 8.2.

**III.14.3.4 SetParam****C++**

```
HRESULT SetParam(IRobotResultParamType _param_id, VARIANT _param_value);
```

**C#**

```
public void SetParam(IRobotResultParamType _param_id, VARIANT _param_value);
```

**Visual Basic**

```
Public Sub SetParam(_param_id As IRobotResultParamType, _param_value As VARIANT)
```

**Description**

Function sets a determined value of the specified parameter.

**Version**

Available since version 8.2.

**III.15 IRobotResultQueryReturnType****C++**

```
enum IRobotResultQueryReturnType;
```

**C#**

```
public enum IRobotResultQueryReturnType;
```

**Visual Basic**

```
Public Enum IRobotResultQueryReturnType
```

**Members**

<b>Members</b>	<b>Description</b>
I_RQRT_DONE = 1	Getting the results has been completed, all results have been got. Available since version 8.2.

I_RQRT_MORE_AVAILABLE = 2	Function that gets the results has given access only to a part of them, it is necessary to call up the function with the same parameters again to get the next part of results. Available since version 8.2.
---------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Version**

Available since version 8.2.

## III.16 IRobotResultStatusType

**C++**

```
enum IRobotResultStatusType;
```

**C#**

```
public enum IRobotResultStatusType;
```

**Visual Basic**

```
Public Enum IRobotResultStatusType
```

**Members**

Members	Description
I_RST_NONE = 0	No results. Available since version 14.
I_RST_AVAILABLE = 1	Results are available and up-to-date. Available since version 14.
I_RST_OUT_OF_DATE = 2	Results are out-of-date. Available since version 14.

**Description**

Type of data describing status of results.

**Version**

Available since version 14.

## III.17 IRobotCalculationResume

**Class Hierarchy****C++**

```
interface IRobotCalculationResume : IDispatch;
```

**C#**

```
public interface IRobotCalculationResume;
```

**Visual Basic**

```
Public Interface IRobotCalculationResume
```

**Description****Version**

Available since version 15.

### III.17.1 IRobotCalculationResume Members

The following tables list the members exposed by IRobotCalculationResume.

#### Public Fields

	Name	Description
◆	DiagonalStiffnessMatrixMax (see page 1029)	.
◆	DiagonalStiffnessMatrixMin (see page 1029)	.
◆	DiagonalStiffnessMatrixPrecision (see page 1029)	.
◆	EquationSolvingMethodUsed (see page 1030)	It returns a method of solving equations which was used to calculate the results.

#### Public Methods

	Name	Description
≡	GetEnergy (see page 1030)	
≡	GetEnergyPrecision (see page 1031)	

### III.17.2 IRobotCalculationResume Fields

The fields of the IRobotCalculationResume class are listed here.

#### Public Fields

	Name	Description
◆	DiagonalStiffnessMatrixMax (see page 1029)	.
◆	DiagonalStiffnessMatrixMin (see page 1029)	.
◆	DiagonalStiffnessMatrixPrecision (see page 1029)	.
◆	EquationSolvingMethodUsed (see page 1030)	It returns a method of solving equations which was used to calculate the results.

#### III.17.2.1 DiagonalStiffnessMatrixMax

##### C++

```
HRESULT get_DiagonalStiffnessMatrixMax(double*);
```

##### C#

```
public double DiagonalStiffnessMatrixMax { get; }
```

##### Visual Basic

```
Public ReadOnly DiagonalStiffnessMatrixMax As double
```

#### Description

#### Version

Available since version 15.

### III.17.2.2 DiagonalStiffnessMatrixMin

#### C++

```
HRESULT get_DiagonalStiffnessMatrixMin(double*);
```

#### C#

```
public double DiagonalStiffnessMatrixMin { get; }
```

#### Visual Basic

```
Public ReadOnly DiagonalStiffnessMatrixMin As double
```

#### Description

#### Version

Available since version 15.

### III.17.2.3 DiagonalStiffnessMatrixPrecision

#### C++

```
HRESULT get_DiagonalStiffnessMatrixPrecision(double*);
```

#### C#

```
public double DiagonalStiffnessMatrixPrecision { get; }
```

#### Visual Basic

```
Public ReadOnly DiagonalStiffnessMatrixPrecision As double
```

#### Description

#### Version

Available since version 15.

### III.17.2.4 EquationSolvingMethodUsed

#### C++

```
HRESULT get_EquationSolvingMethodUsed(IRobotEquationSolvingMethod*);
```

#### C#

```
public IRobotEquationSolvingMethod EquationSolvingMethodUsed { get; }
```

#### Visual Basic

```
Public ReadOnly EquationSolvingMethodUsed As IRobotEquationSolvingMethod
```

#### Description

It returns a method of solving equations which was used to calculate the results.

#### Version

Available since version 16.

## III.17.3 IRobotCalculationResume Methods

The methods of the IRobotCalculationResume class are listed here.

## Public Methods

	Name	Description
»	GetEnergy ( [ see page 1030) )	
»	GetEnergyPrecision ( [ see page 1031) )	

### III.17.3.1 GetEnergy

#### C++

```
HRESULT GetEnergy(long _load_case, double* ret);
```

#### C#

```
public double GetEnergy(long _load_case);
```

#### Visual Basic

```
Public Function GetEnergy(_load_case As long) As double
```

#### Version

Available since version 15.

### III.17.3.2 GetEnergyPrecision

#### C++

```
HRESULT GetEnergyPrecision(long _load_case, double* ret);
```

#### C#

```
public double GetEnergyPrecision(long _load_case);
```

#### Visual Basic

```
Public Function GetEnergyPrecision(_load_case As long) As double
```

#### Version

Available since version 15.

## IV Selections

### Enumerations

	Name	Description
»	IRobotModeSelectionType ( [ see page 1035) )	A set of identifiers determining various manners of mode selection has been defined. .
»	IRobotModeCombinationType ( [ see page 1035) )	A set of identifiers is defined to determine different types of combinations of modes. .
»	IRobotPredefinedSelection ( [ see page 1035) )	

### Interfaces

	Name	Description
»	IRobotSelectionFactory ( [ see page 1031) )	Factory that creates objects representing selections. .
»	IRobotGroup ( [ see page 1036) )	A group of structure components of a given type.
»	IRobotGroupServer ( [ see page 1040) )	Object managing groups of structure elements.

## IV.1 IRobotSelectionFactory

### Class Hierarchy

#### C++

```
interface IRobotSelectionFactory : IDispatch;
```

#### C#

```
public interface IRobotSelectionFactory;
```

### Visual Basic

```
Public Interface IRobotSelectionFactory
```

### Description

Factory that creates objects representing selections. .

## IV.1.1 IRobotSelectionFactory Members

The following tables list the members exposed by IRobotSelectionFactory.

### Public Methods

	Name	Description
≡	Create (see page 1032)	The function creates and returns an object representing selections of structure components of an indicated type. The newly-created selection is empty. .
≡	CreateByLabel (see page 1033)	Function creates a selection of objects of the indicated type, which are assigned a label of the specified type and name. .
≡	CreateByStorey (see page 1033)	Function creates a selection of objects of the specified type assigned to the story with the specified name.
≡	CreateEdgeSelection (see page 1033)	This method creates an empty selection of edges.
≡	CreateFull (see page 1034)	Function creates selection containing all objects of the specified type. Available since version 1.7.
≡	CreateMulti (see page 1034)	The function creates and returns an empty object of the multi-selection type .
≡	CreatePredefined (see page 1034)	Available since version 1.7.
≡	Get (see page 1034)	The funcion gives the current selection of objects of a given type back.

## IV.1.2 IRobotSelectionFactory Methods

The methods of the IRobotSelectionFactory class are listed here.

### Public Methods

	Name	Description
≡	Create (see page 1032)	The function creates and returns an object representing selections of structure components of an indicated type. The newly-created selection is empty. .
≡	CreateByLabel (see page 1033)	Function creates a selection of objects of the indicated type, which are assigned a label of the specified type and name. .
≡	CreateByStorey (see page 1033)	Function creates a selection of objects of the specified type assigned to the story with the specified name.
≡	CreateEdgeSelection (see page 1033)	This method creates an empty selection of edges.

	CreateFull (see page 1034)	Function creates selection containing all objects of the specified type. Available since version 1.7.
	CreateMulti (see page 1034)	The function creates and returns an empty object of the multi-selection type .
	CreatePredefined (see page 1034)	Available since version 1.7.
	Get (see page 1034)	The funcion gives the current selection of objects of a given type back.

#### IV.1.2.1 Create

C++

```
HRESULT Create(IRobotObjectType _object_type, IRobotSelection** ret);
```

C#

```
public IRobotSelection Create(IRobotObjectType _object_type);
```

Visual Basic

```
Public Function Create(_object_type As IRobotObjectType) As IRobotSelection
```

Description

The function creates and returns an object representing selections of structure components of an indicated type. The newly-created selection is empty. .

#### IV.1.2.2 CreateByLabel

C++

```
HRESULT CreateByLabel(IRobotObjectType _obj_type, IRobotLabelType _label_type, BSTR _label_name, IRobotSelection** ret);
```

C#

```
public IRobotSelection CreateByLabel(IRobotObjectType _obj_type, IRobotLabelType _label_type, String _label_name);
```

Visual Basic

```
Public Function CreateByLabel(_obj_type As IRobotObjectType, _label_type As IRobotLabelType, _label_name As String) As IRobotSelection
```

Description

Function creates a selection of objects of the indicated type, which are assigned a label of the specified type and name. .

Version

Available since version 3.

#### IV.1.2.3 CreateByStorey

C++

```
HRESULT CreateByStorey(IRobotObjectType _obj_type, BSTR _storey, IRobotSelection** ret);
```

C#

```
public IRobotSelection CreateByStorey(IRobotObjectType _obj_type, String _storey);
```

Visual Basic

```
Public Function CreateByStorey(_obj_type As IRobotObjectType, _storey As String) As IRobotSelection
```

**Description**

Function creates a selection of objects of the specified type assigned to the story with the specified name.

**Version**

Available since version 9.5.

**IV.1.2.4 CreateEdgeSelection****C++**

```
HRESULT CreateEdgeSelection(IRobotObjEdgeSelection** ret);
```

**C#**

```
public IRobotObjEdgeSelection CreateEdgeSelection();
```

**Visual Basic**

```
Public Function CreateEdgeSelection() As IRobotObjEdgeSelection
```

**Description**

This method creates an empty selection of edges.

**Version**

Available since version 15.2.

**IV.1.2.5 CreateFull****C++**

```
HRESULT CreateFull(IRobotObjectType _obj_type, IRobotSelection** ret);
```

**C#**

```
public IRobotSelection CreateFull(IRobotObjectType _obj_type);
```

**Visual Basic**

```
Public Function CreateFull(_obj_type As IRobotObjectType) As IRobotSelection
```

**Description**

Function creates selection containing all objects of the specified type. Available since version 1.7.

**IV.1.2.6 CreateMulti****C++**

```
HRESULT CreateMulti(IRobotMultiSelection** ret);
```

**C#**

```
public IRobotMultiSelection CreateMulti();
```

**Visual Basic**

```
Public Function CreateMulti() As IRobotMultiSelection
```

**Description**

The function creates and returns an empty object of the multi-selection type .

#### IV.1.2.7 CreatePredefined

**C++**

```
HRESULT CreatePredefined(IRobotPredefinedSelection __predefined_sel, IRobotSelection** ret);
```

**C#**

```
public IRobotSelection CreatePredefined(IRobotPredefinedSelection __predefined_sel);
```

**Visual Basic**

```
Public Function CreatePredefined(__predefined_sel As IRobotPredefinedSelection) As IRobotSelection
```

**Description**

Available since version 1.7.

#### IV.1.2.8 Get

**C++**

```
HRESULT Get(IRobotObjectType _obj_type, IRobotSelection** ret);
```

**C#**

```
public IRobotSelection Get(IRobotObjectType _obj_type);
```

**Visual Basic**

```
Public Function Get(_obj_type As IRobotObjectType) As IRobotSelection
```

**Description**

The function gives the current selection of objects of a given type back.

### IV.2 IRobotModeSelectionType

**C++**

```
enum IRobotModeSelectionType;
```

**C#**

```
public enum IRobotModeSelectionType;
```

**Visual Basic**

```
Public Enum IRobotModeSelectionType
```

**Members**

Members	Description
I_MST_NONE = 1	Empty selection of modes.
I_MST_ALL = 2	Selection containing all modes .
I_MST_SINGLE = 3	Selection of one indicated mode .
I_MST_FIRST_N = 4	Selection of the first n modes .

**Description**

A set of identifiers determining various manners of mode selection has been defined. .

## IV.3 IRobotModeCombinationType

### C++

```
enum IRobotModeCombinationType;
```

### C#

```
public enum IRobotModeCombinationType;
```

### Visual Basic

```
Public Enum IRobotModeCombinationType
```

### Members

Members	Description
I_MCT_NONE = -1	No combinations .
I_MCT_SRSS = 0	SRSS type combination .
I_MCT_CQC = 1	CQC type combination .
I_MCT_10P = 2	10 percent.
I_MCT_2SM = 3	Double sum.
I_MCT_ALL = -2	Available since version 3.5.

### Description

A set of identifiers is defined to determine different types of combinations of modes. .

## IV.4 IRobotPredefinedSelection

### C++

```
enum IRobotPredefinedSelection;
```

### C#

```
public enum IRobotPredefinedSelection;
```

### Visual Basic

```
Public Enum IRobotPredefinedSelection
```

### Members

Members	Description
I_PS_CASE_SIMPLE_CASES = 2	Selection of all simple cases Available since version 1.7.
I_PS_CASE_COMBINATIONS = 3	Selection of all combination (but not code combinations) Available since version 1.7.
I_PS_CASE_CODE_COMBINATIONS = 4	Selection of all code combination Available since version 1.7.
I_PS_NODE_USER_NODES = 21	User-defined nodes. Available since version 2.5.
I_PS_NODE_CALC_NODES = 22	Nodes defined automatically during generation of a calculation model. Available since version 2.5.
I_PS_NODE_SUPPORTED = 23	Nodes at which supports have been defined. Available since version 2.5.
I_PS_BAR_INACTIVE = 31	Inactive bars. Available since version 4.5.

## IV.5 IRobotGroup

### Class Hierarchy

#### C++

```
interface IRobotGroup : IDispatch;
```

#### C#

```
public interface IRobotGroup;
```

### Visual Basic

```
Public Interface IRobotGroup
```

### Description

A group of structure components of a given type.

### Version

Available since version 3.

## IV.5.1 IRobotGroup Members

The following tables list the members exposed by IRobotGroup.

### Public Fields

	Name	Description
◆	Color (see page 1037)	Color assigned to a group.
◆	Name (see page 1037)	Group name.
◆	ObjectType (see page 1038)	Type of objects belonging to the group.
◆	SelList (see page 1038)	List describing the objects belonging to the group.

### Public Methods

	Name	Description
◆	CreateCollection (see page 1038)	Function creates and returns the collection containing the objects belonging to the group. .
◆	CreateSelection (see page 1039)	Function creates and returns the selection describing the objects belonging to the group. .
◆	Read (see page 1039)	Function reads group definition from the structure model.
◆	Store (see page 1039)	Function saves the group definition in the structure model. .

## IV.5.2 IRobotGroup Fields

The fields of the IRobotGroup class are listed here.

### Public Fields

	Name	Description
◆	Color (see page 1037)	Color assigned to a group.
◆	Name (see page 1037)	Group name.
◆	ObjectType (see page 1038)	Type of objects belonging to the group.
◆	SelList (see page 1038)	List describing the objects belonging to the group.

### IV.5.2.1 Color

#### C++

```
HRESULT get_Color(long*);
```

```

HRESULT put_Color(long);

C#
public long Color { get; set; }
```

**Visual Basic**

```
Public Color As long
```

**Description**

Color assigned to a group.

**Version**

Available since version 3.

**IV.5.2.2 Name****C++**

```

HRESULT get_Name(BSTR*);
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Group name.

**Version**

Available since version 3.

**IV.5.2.3 ObjectType****C++**

```

HRESULT get_ObjectType(IRobotObjectType* );
HRESULT put_ObjectType(IRobotObjectType* );
```

**C#**

```
public IRobotObjectType ObjectType { get; set; }
```

**Visual Basic**

```
Public ObjectType As IRobotObjectType
```

**Description**

Type of objects belonging to the group.

**Version**

Available since version 3.

**IV.5.2.4 SelList****C++**

```

HRESULT get_SelList(BSTR* );
HRESULT put_SelList(BSTR);
```

**C#**

```
public String SelList { get; set; }
```

**Visual Basic**

```
Public SelList As String
```

**Description**

List describing the objects belonging to the group.

**Version**

Available since version 3.

**IV.5.3 IRobotGroup Methods**

The methods of the IRobotGroup class are listed here.

**Public Methods**

	Name	Description
💡	CreateCollection (see page 1038)	Function creates and returns the collection containing the objects belonging to the group. .
💡	CreateSelection (see page 1039)	Function creates and returns the selection describing the objects belonging to the group. .
💡	Read (see page 1039)	Function reads group definition from the structure model.
💡	Store (see page 1039)	Function saves the group definition in the structure model. .

**IV.5.3.1 CreateCollection****C++**

```
HRESULT CreateCollection(IRobotCollection** ret);
```

**C#**

```
public IRobotCollection CreateCollection();
```

**Visual Basic**

```
Public Function CreateCollection() As IRobotCollection
```

**Description**

Function creates and returns the collection containing the objects belonging to the group. .

**Version**

Available since version 3.

**IV.5.3.2 CreateSelection****C++**

```
HRESULT CreateSelection(IRobotSelection** ret);
```

**C#**

```
public IRobotSelection CreateSelection();
```

**Visual Basic**

```
Public Function CreateSelection() As IRobotSelection
```

**Description**

Function creates and returns the selection describing the objects belonging to the group. .

**Version**

Available since version 3.

#### IV.5.3.3 Read

**C++**

```
HRESULT Read();
```

**C#**

```
public void Read();
```

**Visual Basic**

```
Public Sub Read()
```

**Description**

Function reads group definition from the structure model.

**Version**

Available since version 3.

#### IV.5.3.4 Store

**C++**

```
HRESULT Store();
```

**C#**

```
public void Store();
```

**Visual Basic**

```
Public Sub Store()
```

**Description**

Function saves the group definition in the structure model. .

**Version**

Available since version 3.

### IV.6 IRobotGroupServer

**Class Hierarchy**

**C++**

```
interface IRobotGroupServer : IDispatch;
```

**C#**

```
public interface IRobotGroupServer;
```

**Visual Basic**

```
Public Interface IRobotGroupServer
```

**Description**

Object managing groups of structure elements.

**Version**

Available since version 3.

## IV.6.1 IRobotGroupServer Members

The following tables list the members exposed by IRobotGroupServer.

### Public Methods

	Name	Description
➊	Create ( [ see page 1041) )	Function creates a new group containing objects of the indicated type with the specified name. Function returns the index of the created group. .
➋	Delete ( [ see page 1041) )	Function deletes the group containing a given object type of the specified index.
➌	Find ( [ see page 1041) )	Function returns index of the group of the given type of objects. If such a group does not exist, then zero value is returned. .
➍	Get ( [ see page 1042) )	Function returns the group containing a given type of objects of the specified index. Groups are indexed from 1 to the number of groups containing a given type of objects.
➎	GetCount ( [ see page 1042) )	Function returns the number of groups containing a given type of objects defined in the structure. .

## IV.6.2 IRobotGroupServer Methods

The methods of the IRobotGroupServer class are listed here.

### Public Methods

	Name	Description
➊	Create ( [ see page 1041) )	Function creates a new group containing objects of the indicated type with the specified name. Function returns the index of the created group. .
➋	Delete ( [ see page 1041) )	Function deletes the group containing a given object type of the specified index.
➌	Find ( [ see page 1041) )	Function returns index of the group of the given type of objects. If such a group does not exist, then zero value is returned. .
➍	Get ( [ see page 1042) )	Function returns the group containing a given type of objects of the specified index. Groups are indexed from 1 to the number of groups containing a given type of objects.
➎	GetCount ( [ see page 1042) )	Function returns the number of groups containing a given type of objects defined in the structure. .

### IV.6.2.1 Create

#### C++

```
HRESULT Create(IRobotObjectType _obj_type, BSTR _name, BSTR _sel_list = "", long _color = 0, long* ret);
```

#### C#

```
public long Create(IRobotObjectType _obj_type, String _name, String _sel_list = "", long _color = 0);
```

#### Visual Basic

```
Public Function Create(_obj_type As IRobotObjectType, _name As String, Optional _sel_list As String = "", Optional _color As Long = 0) As Long
```

#### Description

Function creates a new group containing objects of the indicated type with the specified name. Function returns the index of the created group. .

#### Version

Available since version 3.

#### IV.6.2.2 Delete

##### C++

```
HRESULT Delete(IRobotObjectType _obj_type, long _idx);
```

##### C#

```
public void Delete(IRobotObjectType _obj_type, long _idx);
```

##### Visual Basic

```
Public Sub Delete(_obj_type As IRobotObjectType, _idx As long)
```

##### Description

Function deletes the group containing a given object type of the specified index.

##### Version

Available since version 3.

#### IV.6.2.3 Find

##### C++

```
HRESULT Find(IRobotObjectType _obj_type, BSTR _grp_name, long* ret);
```

##### C#

```
public long Find(IRobotObjectType _obj_type, String _grp_name);
```

##### Visual Basic

```
Public Function Find(_obj_type As IRobotObjectType, _grp_name As String) As long
```

##### Description

Function returns index of the group of the given type of objects. If such a group does not exist, then zero value is returned. .

##### Version

Available since version 3.

#### IV.6.2.4 Get

##### C++

```
HRESULT Get(IRobotObjectType _obj_type, long _idx, IRobotGroup** ret);
```

##### C#

```
public IRobotGroup Get(IRobotObjectType _obj_type, long _idx);
```

##### Visual Basic

```
Public Function Get(_obj_type As IRobotObjectType, _idx As long) As IRobotGroup
```

##### Description

Function returns the group containing a given type of objects of the specified index. Groups are indexed from 1 to the number of groups containing a given type of objects.

##### Version

Available since version 3.

#### IV.6.2.5 GetCount

##### C++

```
HRESULT GetCount(IRobotObjectType _obj_type, long* ret);
```

##### C#

```
public long GetCount(IRobotObjectType _obj_type);
```

##### Visual Basic

```
Public Function GetCount(_obj_type As IRobotObjectType) As long
```

##### Description

Function returns the number of groups containing a given type of objects defined in the structure. .

##### Version

Available since version 3.

## V Edit operations

Available since version 3.5.

##### Enumerations

	Name	Description
	IRobotTranslateOptions (see page 1043)	

##### Interfaces

	Name	Description
	IRobotStructureEditTools (see page 1043)	Interface grouping edit functions that operate on the entire structure or its part. .

## V.1 IRobotTranslateOptions

##### C++

```
enum IRobotTranslateOptions;
```

##### C#

```
public enum IRobotTranslateOptions;
```

##### Visual Basic

```
Public Enum IRobotTranslateOptions
```

##### Members

Members	Description
I_TO_COPY = 0	Available since version 3.5.
I_TO_COPY_WITH_DRAG = 1	Available since version 3.5.
I_TO_MOVE = 2	Available since version 3.5.

##### Version

Available since version 3.5.

## V.2 IRobotStructureEditTools

### Class Hierarchy

#### C++

```
interface IRobotStructureEditTools : IDispatch;
```

#### C#

```
public interface IRobotStructureEditTools;
```

### Visual Basic

```
Public Interface IRobotStructureEditTools
```

#### Description

Interface grouping edit functions that operate on the entire structure or its part. .

#### Version

Available since version 3.5.

### V.2.1 IRobotStructureEditTools Members

The following tables list the members exposed by IRobotStructureEditTools.

#### Public Methods

	Name	Description
≡	DivideBar ( [ see page 1044 )	Function divides the indicated bar based on the provided table with positions of division points. Function returns tables with numbers of bars generated as a result of the division operation.. .
≡	SelMirror ( [ see page 1045 )	
≡	SelRotate ( [ see page 1045 )	Function performs the operation of rotation on the current selection. .
≡	SelScale ( [ see page 1045 )	Function performs the operation of scaling on the current selection. .
≡	SelTranslate ( [ see page 1046 )	Function performs the operation of translation on the current selection. .
≡	TranslateBar ( [ see page 1046 )	Function translates selected bar by the specified vector. Function returns a table with numbers of bars generated as a result of translation. .
≡	TranslateNode ( [ see page 1046 )	Function translates the selected node by the specified vector. Function returns a table with numbers of nodes generated as a result of translation. .

### V.2.2 IRobotStructureEditTools Methods

The methods of the IRobotStructureEditTools class are listed here.

#### Public Methods

	Name	Description
≡	DivideBar ( [ see page 1044 )	Function divides the indicated bar based on the provided table with positions of division points. Function returns tables with numbers of bars generated as a result of the division operation. .
≡	SelMirror ( [ see page 1045 )	
≡	SelRotate ( [ see page 1045 )	Function performs the operation of rotation on the current selection. .
≡	SelScale ( [ see page 1045 )	Function performs the operation of scaling on the current selection. .
≡	SelTranslate ( [ see page 1046 )	Function performs the operation of translation on the current selection. .
≡	TranslateBar ( [ see page 1046 )	Function translates selected bar by the specified vector. Function returns a table with numbers of bars generated as a result of translation. .

	TranslateNode (see page 1046)	Function translates the selected node by the specified vector. Function returns a table with numbers of nodes generated as a result of translation..
-----------------------------------------------------------------------------------	-------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

### V.2.2.1 DivideBar

#### C++

```
HRESULT DivideBar(long _bar_number, IRobotValuesArray* _div_points, VARIANT_BOOL _relative,
IRobotNumbersArray** ret);
```

#### C#

```
public IRobotNumbersArray DivideBar(long _bar_number, IRobotValuesArray _div_points, bool
_relative);
```

#### Visual Basic

```
Public Function DivideBar(_bar_number As long, ByRef _div_points As IRobotValuesArray,
_relative As Boolean) As IRobotNumbersArray
```

#### Description

Function divides the indicated bar based on the provided table with positions of division points. Function returns tables with numbers of bars generated as a result of the division operation..

#### Version

Available since version 3.5.

### V.2.2.2 SelMirror

#### C++

```
HRESULT SelMirror(double _p0x, double _p0y, double _p0z, double _plx, double _ply, double
_p1z, double _p2x, double _p2y, double _p2z, IRobotTranslateOptions _options, long
_repetitions);
```

#### C#

```
public void SelMirror(double _p0x, double _p0y, double _p0z, double _plx, double _ply,
double _p1z, double _p2x, double _p2y, double _p2z, IRobotTranslateOptions _options, long
_repetitions);
```

#### Visual Basic

```
Public Sub SelMirror(_p0x As double, _p0y As double, _p0z As double, _plx As double, _ply
As double, _p1z As double, _p2x As double, _p2y As double, _p2z As double, _options As
IRobotTranslateOptions, _repetitions As long)
```

#### Version

Available since version 4.

### V.2.2.3 SelRotate

#### C++

```
HRESULT SelRotate(double _p0x, double _p0y, double _p0z, double _plx, double _ply, double
_p1z, double _angle, IRobotTranslateOptions _options, long _repetitions);
```

#### C#

```
public void SelRotate(double _p0x, double _p0y, double _p0z, double _plx, double _ply,
double _p1z, double _angle, IRobotTranslateOptions _options, long _repetitions);
```

#### Visual Basic

```
Public Sub SelRotate(_p0x As double, _p0y As double, _p0z As double, _plx As double, _ply
```

```
As double, _plz As double, _angle As double, _options As IRobotTranslateOptions,
_repetitions As long)
```

**Description**

Function performs the operation of rotation on the current selection. .

**Version**

Available since version 4.

**V.2.2.4 SelScale****C++**

```
HRESULT SelScale(double _p0x, double _p0y, double _p0z, double _factor,
IRobotTranslateOptions _options, long _repetitions);
```

**C#**

```
public void SelScale(double _p0x, double _p0y, double _p0z, double _factor,
IRobotTranslateOptions _options, long _repetitions);
```

**Visual Basic**

```
Public Sub SelScale(_p0x As double, _p0y As double, _p0z As double, _factor As double,
_options As IRobotTranslateOptions, _repetitions As long)
```

**Description**

Function performs the operation of scaling on the current selection. .

**Version**

Available since version 4.

**V.2.2.5 SelTranslate****C++**

```
HRESULT SelTranslate(double _dx, double _dy, double _dz, IRobotTranslateOptions _options,
long _repetitions);
```

**C#**

```
public void SelTranslate(double _dx, double _dy, double _dz, IRobotTranslateOptions
_options, long _repetitions);
```

**Visual Basic**

```
Public Sub SelTranslate(_dx As double, _dy As double, _dz As double, _options As
IRobotTranslateOptions, _repetitions As long)
```

**Description**

Function performs the operation of translation on the current selection. .

**Version**

Available since version 4.

**V.2.2.6 TranslateBar****C++**

```
HRESULT TranslateBar(long _bar_number, double _dx, double _dy, double _dz,
IRobotTranslateOptions _trans_option, long _repetitions = 1, IRobotNumbersArray** ret);
```

**C#**

```
public IRobotNumbersArray TranslateBar(long _bar_number, double _dx, double _dy, double
```

```
_dz, IRobotTranslateOptions _trans_option, long _repetitions = 1);
```

### Visual Basic

```
Public Function TranslateBar(_bar_number As long, _dx As double, _dy As double, _dz As double, _trans_option As IRobotTranslateOptions, Optional _repetitions As long = 1) As IRobotNumbersArray
```

### Description

Function translates selected bar by the specified vector. Function returns a table with numbers of bars generated as a result of translation. .

### Version

Available since version 3.5.

## V.2.2.7 TranslateNode

### C++

```
HRESULT TranslateNode(long _node_number, double _dx, double _dy, double _dz,
IRobotTranslateOptions _trans_option, long _repetitions, IRobotNumbersArray** ret);
```

### C#

```
public IRobotNumbersArray TranslateNode(long _node_number, double _dx, double _dy, double
_dz, IRobotTranslateOptions _trans_option, long _repetitions);
```

### Visual Basic

```
Public Function TranslateNode(_node_number As long, _dx As double, _dy As double, _dz As
double, _trans_option As IRobotTranslateOptions, _repetitions As long) As IRobotNumbersArray
```

### Description

Function translates the selected node by the specified vector. Function returns a table with numbers of nodes generated as a result of translation. .

### Version

Available since version 3.5.

## VI Stories

### Interfaces

	Name	Description
↪	IRobotStorey (see page 1047)	Interface describing a story in a structure. .
↪	IRobotStoreyMngr (see page 1052)	Structure story manager.
↪	IRobotStoreySelection (see page 1056)	Interface that enables selecting stories. Stories are identified by name or number. A story no. is simultaneously the index of this story in RobotStoreyMngr. The object can also be used for selecting levels. Adding a specified story to the selection adds the top level of this story. Additionally, you can add the building base level to the selection specifying 0 as a story no.
↪	IRobotStoreyValues (see page 1059)	Bar properties.
↪	IRobotStoreyReducedForces (see page 1064)	
↪	IRobotStoreyDisplacements (see page 1069)	
↪	IRobotStoreyResultServer (see page 1072)	Server of results for stories.

## VI.1 IRobotStorey

### Class Hierarchy

#### C++

```
interface IRobotStorey : IDispatch;
```

#### C#

```
public interface IRobotStorey;
```

### Visual Basic

```
Public Interface IRobotStorey
```

#### Description

Interface describing a story in a structure. .

### VI.1.1 IRobotStorey Members

The following tables list the members exposed by IRobotStorey.

#### Public Fields

	Name	Description
◆	AutomaticSelection (see page 1048)	Flag indicating if selection of objects assigned to a story should be automatic .
◆	Color (see page 1049)	Color assigned to the story.
◆	Ex1 (see page 1049)	Accidental eccentricity along the X axis.
◆	Ey1 (see page 1049)	Accidental eccentricity along the Y axis.
◆	Height (see page 1049)	Story height.
◆	Lx (see page 1050)	Story dimension along the X axis.
◆	Ly (see page 1050)	Story dimension along the Y axis.
◆	Name (see page 1050)	Story name.
◆	Objects (see page 1051)	Selection of objects assigned to a story.
◆	TopLevel (see page 1051)	Top level of a story.

#### Public Methods

	Name	Description
◆	SetHeight (see page 1051)	Function changes the story height.
◆	SetTopLevel (see page 1052)	Function changes a position of the top level of a story.

### VI.1.2 IRobotStorey Fields

The fields of the IRobotStorey class are listed here.

#### Public Fields

	Name	Description
◆	AutomaticSelection (see page 1048)	Flag indicating if selection of objects assigned to a story should be automatic .
◆	Color (see page 1049)	Color assigned to the story.
◆	Ex1 (see page 1049)	Accidental eccentricity along the X axis.
◆	Ey1 (see page 1049)	Accidental eccentricity along the Y axis.
◆	Height (see page 1049)	Story height.
◆	Lx (see page 1050)	Story dimension along the X axis.
◆	Ly (see page 1050)	Story dimension along the Y axis.

◆	Name (see page 1050)	Story name.
◆	Objects (see page 1051)	Selection of objects assigned to a story.
◆	TopLevel (see page 1051)	Top level of a story.

### VI.1.2.1 AutomaticSelection

#### C++

```
HRESULT get_AutomaticSelection(VARIANT_BOOL* );
HRESULT put_AutomaticSelection(VARIANT_BOOL);
```

#### C#

```
public bool AutomaticSelection { get; set; }
```

#### Visual Basic

```
Public AutomaticSelection As Boolean
```

#### Description

Flag indicating if selection of objects assigned to a story should be automatic .

### VI.1.2.2 Color

#### C++

```
HRESULT get_Color(long* );
HRESULT put_Color(long);
```

#### C#

```
public long Color { get; set; }
```

#### Visual Basic

```
Public Color As long
```

#### Description

Color assigned to the story.

### VI.1.2.3 Ex1

#### C++

```
HRESULT get_Ex1(double* );
```

#### C#

```
public double Ex1 { get; }
```

#### Visual Basic

```
Public ReadOnly Ex1 As double
```

#### Description

Accidental eccentricity along the X axis.

#### Version

Available since version 9.7.

### VI.1.2.4 Ey1

#### C++

```
HRESULT get_Ey1(double* );
```

**C#**

```
public double Ey1 { get; }
```

**Visual Basic**

```
Public ReadOnly Ey1 As Double
```

**Description**

Accidental eccentricity along the Y axis.

**Version**

Available since version 9.7.

### VI.1.2.5 Height

**C++**

```
HRESULT get_Height(double*);  
HRESULT put_Height(double);
```

**C#**

```
public double Height { get; set; }
```

**Visual Basic**

```
Public Height As Double
```

**Description**

Story height.

### VI.1.2.6 Lx

**C++**

```
HRESULT get_Lx(double*);
```

**C#**

```
public double Lx { get; }
```

**Visual Basic**

```
Public ReadOnly Lx As Double
```

**Description**

Story dimension along the X axis.

**Version**

Available since version 9.7.

### VI.1.2.7 Ly

**C++**

```
HRESULT get_Ly(double*);
```

**C#**

```
public double Ly { get; }
```

**Visual Basic**

```
Public ReadOnly Ly As Double
```

**Description**

Story dimension along the Y axis.

**Version**

Available since version 9.7.

**VI.1.2.8 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Story name.

**VI.1.2.9 Objects****C++**

```
HRESULT get_Objects(BSTR* );
HRESULT put_Objects(BSTR);
```

**C#**

```
public String Objects { get; set; }
```

**Visual Basic**

```
Public Objects As String
```

**Description**

Selection of objects assigned to a story.

**VI.1.2.10 TopLevel****C++**

```
HRESULT get_TopLevel(double* );
HRESULT put_TopLevel(double);
```

**C#**

```
public double TopLevel { get; set; }
```

**Visual Basic**

```
Public TopLevel As double
```

**Description**

Top level of a story.

**Version**

Available since version 9.5.

**VI.1.3 IRobotStorey Methods**

The methods of the IRobotStorey class are listed here.

## Public Methods

	Name	Description
ESH	SetHeight (see page 1051)	Function changes the story height.
ESH	SetTopLevel (see page 1052)	Function changes a position of the top level of a story.

### VI.1.3.1 SetHeight

#### C++

```
HRESULT SetHeight(double _height, VARIANT_BOOL _affect_geometry = -1);
```

#### C#

```
public void SetHeight(double _height, bool _affect_geometry = -1);
```

#### Visual Basic

```
Public Sub SetHeight(_height As double, Optional _affect_geometry As Boolean = -1)
```

#### Description

Function changes the story height.

#### Version

Available since version 9.5.

### VI.1.3.2 SetTopLevel

#### C++

```
HRESULT SetTopLevel(double _top_level, VARIANT_BOOL _affect_geometry = -1);
```

#### C#

```
public void SetTopLevel(double _top_level, bool _affect_geometry = -1);
```

#### Visual Basic

```
Public Sub SetTopLevel(_top_level As double, Optional _affect_geometry As Boolean = -1)
```

#### Description

Function changes a position of the top level of a story.

#### Version

Available since version 9.5.

## VI.2 IRobotStoreyMngr

#### Class Hierarchy

#### C++

```
interface IRobotStoreyMngr : IDispatch;
```

#### C#

```
public interface IRobotStoreyMngr;
```

#### Visual Basic

```
Public Interface IRobotStoreyMngr
```

#### Description

Structure story manager.

## VI.2.1 IRobotStoreyMngr Members

The following tables list the members exposed by IRobotStoreyMngr.

### Public Fields

	Name	Description
◆	BaseLevel ( <a href="#">see page 1053</a> )	Building base level.
◆	Count ( <a href="#">see page 1053</a> )	Number of stories defined in a structure.
◆	DisregardedObjects ( <a href="#">see page 1054</a> )	Selection of objects that should be disregarded in automatic recognition of structure elements belonging to individual stories.
◆	FilterStorey ( <a href="#">see page 1054</a> )	Name of the active filter story to which editing operations are limited; an empty character string indicates that a filter story is not set.

### Public Methods

	Name	Description
◆	Create2 ( <a href="#">see page 1055</a> )	Function creates a new story with the specified parameters and returns its index. The created story is assigned a default automatic color and automatic object selection.
◆	Create2Ex ( <a href="#">see page 1055</a> )	Function creates a story with the specified parameters. Function returns the index of the created story.
◆	Delete ( <a href="#">see page 1055</a> )	Function deletes the story with the given index. .
◆	DeleteAll ( <a href="#">see page 1056</a> )	Function deletes all stories defined in a structure.
◆	Find ( <a href="#">see page 1056</a> )	Function searches for the story with the specified name and returns its index. If the story with the given name does not exist, function returns zero value. .
◆	Get ( <a href="#">see page 1056</a> )	Function returns definition of a story with the specified index. Stories are indexed with numbers from 1 to Count ( <a href="#">see page 1053</a> ).

## VI.2.2 IRobotStoreyMngr Fields

The fields of the IRobotStoreyMngr class are listed here.

### Public Fields

	Name	Description
◆	BaseLevel ( <a href="#">see page 1053</a> )	Building base level.
◆	Count ( <a href="#">see page 1053</a> )	Number of stories defined in a structure.
◆	DisregardedObjects ( <a href="#">see page 1054</a> )	Selection of objects that should be disregarded in automatic recognition of structure elements belonging to individual stories.
◆	FilterStorey ( <a href="#">see page 1054</a> )	Name of the active filter story to which editing operations are limited; an empty character string indicates that a filter story is not set.

### VI.2.2.1 BaseLevel

#### C++

```
HRESULT get_BaseLevel(double* );
HRESULT put_BaseLevel(double);
```

#### C#

```
public double BaseLevel { get; set; }
```

#### Visual Basic

```
Public BaseLevel As Double
```

#### Description

Building base level.

**Version**

Available since version 9.5.

**VI.2.2.2 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of stories defined in a structure.

**VI.2.2.3 DisregardedObjects****C++**

```
HRESULT get_DisregardedObjects(BSTR*);  
HRESULT put_DisregardedObjects(BSTR);
```

**C#**

```
public String DisregardedObjects { get; set; }
```

**Visual Basic**

```
Public DisregardedObjects As String
```

**Description**

Selection of objects that should be disregarded in automatic recognition of structure elements belonging to individual stories.

**VI.2.2.4 FilterStorey****C++**

```
HRESULT get_FilterStorey(BSTR*);  
HRESULT put_FilterStorey(BSTR);
```

**C#**

```
public String FilterStorey { get; set; }
```

**Visual Basic**

```
Public FilterStorey As String
```

**Description**

Name of the active filter story to which editing operations are limited; an empty character string indicates that a filter story is not set.

**Version**

Available since version 9.5.

**VI.2.3 IRobotStoreyMngr Methods**

The methods of the IRobotStoreyMngr class are listed here.

## Public Methods

	Name	Description
⊕	Create2 ( [ see page 1055 )	Function creates a new story with the specified parameters and returns its index. The created story is assigned a default automatic color and automatic object selection.
⊕	Create2Ex ( [ see page 1055 )	Function creates a story with the specified parameters. Function returns the index of the created story.
⊕	Delete ( [ see page 1055 )	Function deletes the story with the given index. .
⊕	DeleteAll ( [ see page 1056 )	Function deletes all stories defined in a structure.
⊕	Find ( [ see page 1056 )	Function searches for the story with the specified name and returns its index. If the story with the given name does not exist, function returns zero value..
⊕	Get ( [ see page 1056 )	Function returns definition of a story with the specified index. Stories are indexed with numbers from 1 to Count ( [ see page 1053 ).

### VI.2.3.1 Create2

#### C++

```
HRESULT Create2(BSTR _name, double _top_level, long* ret);
```

#### C#

```
public long Create2(String _name, double _top_level);
```

#### Visual Basic

```
Public Function Create2(_name As String, _top_level As double) As long
```

#### Description

Function creates a new story with the specified parameters and returns its index. The created story is assigned a default automatic color and automatic object selection.

#### Version

Available since version 9.5.

### VI.2.3.2 Create2Ex

#### C++

```
HRESULT Create2Ex(BSTR _name, double _top_level, BSTR _objects = "", long _color = -1, long* ret);
```

#### C#

```
public long Create2Ex(String _name, double _top_level, String _objects = "", long _color = -1);
```

#### Visual Basic

```
Public Function Create2Ex(_name As String, _top_level As double, Optional _objects As String = "", Optional _color As long = -1) As long
```

#### Description

Function creates a story with the specified parameters. Function returns the index of the created story.

#### Version

Available since version 9.5.

### VI.2.3.3 Delete

**C++**

```
HRESULT Delete(long _idx);
```

**C#**

```
public void Delete(long _idx);
```

**Visual Basic**

```
Public Sub Delete(_idx As long)
```

**Description**

Function deletes the story with the given index. .

### VI.2.3.4 DeleteAll

**C++**

```
HRESULT DeleteAll();
```

**C#**

```
public void DeleteAll();
```

**Visual Basic**

```
Public Sub DeleteAll()
```

**Description**

Function deletes all stories defined in a structure.

### VI.2.3.5 Find

**C++**

```
HRESULT Find(BSTR _story_name, long* ret);
```

**C#**

```
public long Find(String _story_name);
```

**Visual Basic**

```
Public Function Find(_story_name As String) As long
```

**Description**

Function searches for the story with the specified name and returns its index. If the story with the given name does not exist, function returns zero value. .

### VI.2.3.6 Get

**C++**

```
HRESULT Get(long _idx, IRobotStorey** ret);
```

**C#**

```
public IRobotStorey Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotStorey
```

## Description

Function returns definition of a story with the specified index. Stories are indexed with numbers from 1 to Count ([see page 1053](#)).

## VI.3 IRobotStoreySelection

### Class Hierarchy

### C++

```
interface IRobotStoreySelection : IDispatch;
```

### C#

```
public interface IRobotStoreySelection;
```

### Visual Basic

```
Public Interface IRobotStoreySelection
```

### Description

Interface that enables selecting stories. Stories are identified by name or number. A story no. is simultaneously the index of this story in RobotStoreyMngr. The object can also be used for selecting levels. Adding a specified story to the selection adds the top level of this story. Additionally, you can add the building base level to the selection specifying 0 as a story no.

### Version

Available since version 9.7.

## VI.3.1 IRobotStoreySelection Members

The following tables list the members exposed by IRobotStoreySelection.

### Public Methods

	Name	Description
	Add ( <a href="#">see page 1057</a> )	Function adds to the selection a story with the specified number or name.
	AddAll ( <a href="#">see page 1058</a> )	Function adds all stories to the selection.
	Clear ( <a href="#">see page 1058</a> )	Function clears the selection.
	GetNames ( <a href="#">see page 1058</a> )	Function returns a table with names of stories that belong to the selection.
	GetNumbers ( <a href="#">see page 1059</a> )	Function returns a table with numbers of stories that belong to the selection.
	Remove ( <a href="#">see page 1059</a> )	Function deletes from the selection a story with the specified number or name.

## VI.3.2 IRobotStoreySelection Methods

The methods of the IRobotStoreySelection class are listed here.

### Public Methods

	Name	Description
	Add ( <a href="#">see page 1057</a> )	Function adds to the selection a story with the specified number or name.
	AddAll ( <a href="#">see page 1058</a> )	Function adds all stories to the selection.
	Clear ( <a href="#">see page 1058</a> )	Function clears the selection.
	GetNames ( <a href="#">see page 1058</a> )	Function returns a table with names of stories that belong to the selection.
	GetNumbers ( <a href="#">see page 1059</a> )	Function returns a table with numbers of stories that belong to the selection.
	Remove ( <a href="#">see page 1059</a> )	Function deletes from the selection a story with the specified number or name.

### VI.3.2.1 Add

#### C++

```
HRESULT Add(VARIANT _storey);
```

#### C#

```
public void Add(VARIANT _storey);
```

#### Visual Basic

```
Public Sub Add(_storey As VARIANT)
```

#### Description

Function adds to the selection a story with the specified number or name.

#### Version

Available since version 9.7.

### VI.3.2.2 AddAll

#### C++

```
HRESULT AddAll();
```

#### C#

```
public void AddAll();
```

#### Visual Basic

```
Public Sub AddAll()
```

#### Description

Function adds all stories to the selection.

#### Version

Available since version 9.7.

### VI.3.2.3 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

Function clears the selection.

#### Version

Available since version 9.7.

#### VI.3.2.4 GetNames

C++

```
HRESULT GetNames(SAFEARRAY** ret);
```

C#

```
public Array GetNames();
```

Visual Basic

```
Public Function GetNames() As string()
```

Description

Function returns a table with names of stories that belong to the selection.

Version

Available since version 9.7.

#### VI.3.2.5 GetNumbers

C++

```
HRESULT GetNumbers(SAFEARRAY** ret);
```

C#

```
public Array GetNumbers();
```

Visual Basic

```
Public Function GetNumbers() As long()
```

Description

Function returns a table with numbers of stories that belong to the selection.

Version

Available since version 9.7.

#### VI.3.2.6 Remove

C++

```
HRESULT Remove(VARIANT _storey);
```

C#

```
public void Remove(VARIANT _storey);
```

Visual Basic

```
Public Sub Remove(_storey As VARIANT)
```

Description

Function deletes from the selection a story with the specified number or name.

Version

Available since version 9.7.

### VI.4 IRobotStoreyValues

Class Hierarchy

**C++**

```
interface IRobotStoreyValues : IDispatch;
```

**C#**

```
public interface IRobotStoreyValues;
```

**Visual Basic**

```
Public Interface IRobotStoreyValues
```

**Description**

Bar properties.

**Version**

Available since version 9.7.

**VI.4.1 IRobotStoreyValues Members**

The following tables list the members exposed by IRobotStoreyValues.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Ex0 (see page 1060)	Ex0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the X axis.
❖	Ex1 (see page 1061)	Accidental eccentricity along the X axis.
❖	Ex2 (see page 1061)	Additional eccentricity along the X axis.
❖	Ey0 (see page 1061)	Ey0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the Y axis.
❖	Ey1 (see page 1062)	Accidental eccentricity along the Y axis.
❖	Ey2 (see page 1062)	Additional eccentricity along the Y axis.
❖	F (see page 1062)	Center of gravity of floor slab.
❖	G (see page 1063)	Center of gravity.
❖	Ix (see page 1063)	Moment of inertia (X axis).
❖	Iy (see page 1063)	Moment of inertia (Y axis).
❖	Iz (see page 1063)	Moment of inertia (Z axis).
❖	Mass (see page 1064)	Mass.
❖	R (see page 1064)	Center of rigidity.

**VI.4.2 IRobotStoreyValues Fields**

The fields of the IRobotStoreyValues class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Ex0 (see page 1060)	Ex0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the X axis.
❖	Ex1 (see page 1061)	Accidental eccentricity along the X axis.
❖	Ex2 (see page 1061)	Additional eccentricity along the X axis.
❖	Ey0 (see page 1061)	Ey0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the Y axis.
❖	Ey1 (see page 1062)	Accidental eccentricity along the Y axis.
❖	Ey2 (see page 1062)	Additional eccentricity along the Y axis.
❖	F (see page 1062)	Center of gravity of floor slab.
❖	G (see page 1063)	Center of gravity.

◆	Ix (see page 1063)	Moment of inertia (X axis).
◆	Iy (see page 1063)	Moment of inertia (Y axis).
◆	Iz (see page 1063)	Moment of inertia (Z axis).
◆	Mass (see page 1064)	Mass.
◆	R (see page 1064)	Center of rigidity.

#### VI.4.2.1 Ex0

##### C++

```
HRESULT get_Ex0(double*);
```

##### C#

```
public double Ex0 { get; }
```

##### Visual Basic

```
Public ReadOnly Ex0 As double
```

##### Description

Ex0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the X axis.

##### Version

Available since version 9.7.

#### VI.4.2.2 Ex1

##### C++

```
HRESULT get_Ex1(double*);
```

##### C#

```
public double Ex1 { get; }
```

##### Visual Basic

```
Public ReadOnly Ex1 As double
```

##### Description

Accidental eccentricity along the X axis.

##### Version

Available since version 9.7.

#### VI.4.2.3 Ex2

##### C++

```
HRESULT get_Ex2(double*);
```

##### C#

```
public double Ex2 { get; }
```

##### Visual Basic

```
Public ReadOnly Ex2 As double
```

##### Description

Additional eccentricity along the X axis.

##### Version

Available since version 9.7.

#### VI.4.2.4 Ey0

##### C++

```
HRESULT get_Ey0(double*);
```

##### C#

```
public double Ey0 { get; }
```

##### Visual Basic

```
Public ReadOnly Ey0 As double
```

##### Description

Ey0 eccentricity - a distance between the center of rigidity and the center of gravity projected on the Y axis.

##### Version

Available since version 9.7.

#### VI.4.2.5 Ey1

##### C++

```
HRESULT get_Ey1(double*);
```

##### C#

```
public double Ey1 { get; }
```

##### Visual Basic

```
Public ReadOnly Ey1 As double
```

##### Description

Accidental eccentricity along the Y axis.

##### Version

Available since version 9.7.

#### VI.4.2.6 Ey2

##### C++

```
HRESULT get_Ey2(double*);
```

##### C#

```
public double Ey2 { get; }
```

##### Visual Basic

```
Public ReadOnly Ey2 As double
```

##### Description

Additional eccentricity along the Y axis.

##### Version

Available since version 9.7.

#### VI.4.2.7 F

##### C++

```
HRESULT get_F(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D F { get; }
```

**Visual Basic**

```
Public ReadOnly F As IRobotGeoPoint3D
```

**Description**

Center of gravity of floor slab.

**Version**

Available since version 15.2.

**VI.4.2.8 G****C++**

```
HRESULT get_G(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D G { get; }
```

**Visual Basic**

```
Public ReadOnly G As IRobotGeoPoint3D
```

**Description**

Center of gravity.

**Version**

Available since version 9.7.

**VI.4.2.9 Ix****C++**

```
HRESULT get_Ix(double*);
```

**C#**

```
public double Ix { get; }
```

**Visual Basic**

```
Public ReadOnly Ix As Double
```

**Description**

Moment of inertia (X axis).

**Version**

Available since version 9.7.

**VI.4.2.10 Iy****C++**

```
HRESULT get_Iy(double*);
```

**C#**

```
public double Iy { get; }
```

**Visual Basic**

```
Public ReadOnly Iy As Double
```

**Description**

Moment of inertia (Y axis).

**Version**

Available since version 9.7.

**VI.4.2.11 Iz****C++**

```
HRESULT get_Iz(double*);
```

**C#**

```
public double Iz { get; }
```

**Visual Basic**

```
Public ReadOnly Iz As double
```

**Description**

Moment of inertia (Z axis).

**Version**

Available since version 9.7.

**VI.4.2.12 Mass****C++**

```
HRESULT get_Mass(double*);
```

**C#**

```
public double Mass { get; }
```

**Visual Basic**

```
Public ReadOnly Mass As double
```

**Description**

Mass.

**Version**

Available since version 9.7.

**VI.4.2.13 R****C++**

```
HRESULT get_R(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D R { get; }
```

**Visual Basic**

```
Public ReadOnly R As IRobotGeoPoint3D
```

**Description**

Center of rigidity.

**Version**

Available since version 9.7.

## VI.5 IRobotStoreyReducedForces

### Class Hierarchy

#### C++

```
interface IRobotStoreyReducedForces : IDispatch;
```

#### C#

```
public interface IRobotStoreyReducedForces;
```

### Visual Basic

```
Public Interface IRobotStoreyReducedForces
```

### Version

Available since version 9.7.

## VI.5.1 IRobotStoreyReducedForces Members

The following tables list the members exposed by IRobotStoreyReducedForces.

### Public Fields

	Name	Description
◆	FX ( <a href="#">see page 1066</a> )	
◆	FX_ToColumns ( <a href="#">see page 1066</a> )	
◆	FX_ToWalls ( <a href="#">see page 1066</a> )	
◆	FY ( <a href="#">see page 1066</a> )	
◆	FY_ToColumns ( <a href="#">see page 1067</a> )	
◆	FY_ToWalls ( <a href="#">see page 1067</a> )	
◆	FZ ( <a href="#">see page 1067</a> )	
◆	FZ_ToColumns ( <a href="#">see page 1067</a> )	
◆	FZ_ToWalls ( <a href="#">see page 1068</a> )	
◆	MX ( <a href="#">see page 1068</a> )	
◆	MY ( <a href="#">see page 1068</a> )	
◆	MZ ( <a href="#">see page 1068</a> )	

## VI.5.2 IRobotStoreyReducedForces Fields

The fields of the IRobotStoreyReducedForces class are listed here.

### Public Fields

	Name	Description
◆	FX ( <a href="#">see page 1066</a> )	
◆	FX_ToColumns ( <a href="#">see page 1066</a> )	
◆	FX_ToWalls ( <a href="#">see page 1066</a> )	
◆	FY ( <a href="#">see page 1066</a> )	
◆	FY_ToColumns ( <a href="#">see page 1067</a> )	
◆	FY_ToWalls ( <a href="#">see page 1067</a> )	
◆	FZ ( <a href="#">see page 1067</a> )	
◆	FZ_ToColumns ( <a href="#">see page 1067</a> )	
◆	FZ_ToWalls ( <a href="#">see page 1068</a> )	
◆	MX ( <a href="#">see page 1068</a> )	
◆	MY ( <a href="#">see page 1068</a> )	

	MZ (see page 1068)	
--	--------------------	--

### VI.5.2.1 FX

**C++**

```
HRESULT get_FX(double*);
```

**C#**

```
public double FX { get; }
```

**Visual Basic**

```
Public ReadOnly FX As double
```

**Version**

Available since version 9.7.

### VI.5.2.2 FX\_ToColumns

**C++**

```
HRESULT get_FX_ToColumns(double*);
```

**C#**

```
public double FX_ToColumns { get; }
```

**Visual Basic**

```
Public ReadOnly FX_ToColumns As double
```

**Version**

Available since version 9.7.

### VI.5.2.3 FX\_Towalls

**C++**

```
HRESULT get_FX_Towalls(double*);
```

**C#**

```
public double FX_Towalls { get; }
```

**Visual Basic**

```
Public ReadOnly FX_Towalls As double
```

**Version**

Available since version 9.7.

### VI.5.2.4 FY

**C++**

```
HRESULT get_FY(double*);
```

**C#**

```
public double FY { get; }
```

**Visual Basic**

```
Public ReadOnly FY As double
```

**Version**

Available since version 9.7.

### VI.5.2.5 FY\_ToColumns

**C++**

```
HRESULT get_FY_ToColumns(double*);
```

**C#**

```
public double FY_ToColumns { get; }
```

**Visual Basic**

```
Public ReadOnly FY_ToColumns As double
```

**Version**

Available since version 9.7.

### VI.5.2.6 FY\_ToWalls

**C++**

```
HRESULT get_FY_ToWalls(double*);
```

**C#**

```
public double FY_ToWalls { get; }
```

**Visual Basic**

```
Public ReadOnly FY_ToWalls As double
```

**Version**

Available since version 9.7.

### VI.5.2.7 FZ

**C++**

```
HRESULT get_FZ(double*);
```

**C#**

```
public double FZ { get; }
```

**Visual Basic**

```
Public ReadOnly FZ As double
```

**Version**

Available since version 9.7.

### VI.5.2.8 FZ\_ToColumns

**C++**

```
HRESULT get_FZ_ToColumns(double*);
```

**C#**

```
public double FZ_ToColumns { get; }
```

**Visual Basic**

```
Public ReadOnly FZ_ToColumns As double
```

**Version**

Available since version 9.7.

### VI.5.2.9 FZ\_ToWalls

#### C++

```
HRESULT get_FZ_ToWalls(double*);
```

#### C#

```
public double FZ_ToWalls { get; }
```

#### Visual Basic

```
Public ReadOnly FZ_ToWalls As double
```

#### Version

Available since version 9.7.

### VI.5.2.10 MX

#### C++

```
HRESULT get_MX(double*);
```

#### C#

```
public double MX { get; }
```

#### Visual Basic

```
Public ReadOnly MX As double
```

#### Version

Available since version 9.7.

### VI.5.2.11 MY

#### C++

```
HRESULT get_MY(double*);
```

#### C#

```
public double MY { get; }
```

#### Visual Basic

```
Public ReadOnly MY As double
```

#### Version

Available since version 9.7.

### VI.5.2.12 MZ

#### C++

```
HRESULT get_MZ(double*);
```

#### C#

```
public double MZ { get; }
```

#### Visual Basic

```
Public ReadOnly MZ As double
```

#### Version

Available since version 9.7.

## VI.6 IRobotStoreyDisplacements

### Class Hierarchy

#### C++

```
interface IRobotStoreyDisplacements : IDispatch;
```

#### C#

```
public interface IRobotStoreyDisplacements;
```

### Visual Basic

```
Public Interface IRobotStoreyDisplacements
```

### Version

Available since version 9.7.

## VI.6.1 IRobotStoreyDisplacements Members

The following tables list the members exposed by IRobotStoreyDisplacements.

### Public Fields

	Name	Description
◆	DrUX (see page 1070)	
◆	DrUY (see page 1070)	
◆	MaxUX (see page 1070)	
◆	MaxUY (see page 1070)	
◆	MinUX (see page 1071)	
◆	MinUY (see page 1071)	
◆	NodeMaxUX (see page 1071)	
◆	NodeMaxUY (see page 1071)	
◆	NodeMinUX (see page 1072)	
◆	NodeMinUY (see page 1072)	

## VI.6.2 IRobotStoreyDisplacements Fields

The fields of the IRobotStoreyDisplacements class are listed here.

### Public Fields

	Name	Description
◆	DrUX (see page 1070)	
◆	DrUY (see page 1070)	
◆	MaxUX (see page 1070)	
◆	MaxUY (see page 1070)	
◆	MinUX (see page 1071)	
◆	MinUY (see page 1071)	
◆	NodeMaxUX (see page 1071)	
◆	NodeMaxUY (see page 1071)	
◆	NodeMinUX (see page 1072)	
◆	NodeMinUY (see page 1072)	

### VI.6.2.1 DrUX

#### C++

```
HRESULT get_DrUX(double*);
```

#### C#

```
public double DrUX { get; }
```

#### Visual Basic

```
Public ReadOnly DrUX As double
```

#### Version

Available since version 9.7.

### VI.6.2.2 DrUY

#### C++

```
HRESULT get_DrUY(double*);
```

#### C#

```
public double DrUY { get; }
```

#### Visual Basic

```
Public ReadOnly DrUY As double
```

#### Version

Available since version 9.7.

### VI.6.2.3 MaxUX

#### C++

```
HRESULT get_MaxUX(double*);
```

#### C#

```
public double MaxUX { get; }
```

#### Visual Basic

```
Public ReadOnly MaxUX As double
```

#### Version

Available since version 9.7.

### VI.6.2.4 MaxUY

#### C++

```
HRESULT get_MaxUY(double*);
```

#### C#

```
public double MaxUY { get; }
```

#### Visual Basic

```
Public ReadOnly MaxUY As double
```

#### Version

Available since version 9.7.

### VI.6.2.5 MinUX

#### C++

```
HRESULT get_MinUX(double*);
```

#### C#

```
public double MinUX { get; }
```

#### Visual Basic

```
Public ReadOnly MinUX As double
```

#### Version

Available since version 9.7.

### VI.6.2.6 MinUY

#### C++

```
HRESULT get_MinUY(double*);
```

#### C#

```
public double MinUY { get; }
```

#### Visual Basic

```
Public ReadOnly MinUY As double
```

#### Version

Available since version 9.7.

### VI.6.2.7 NodeMaxUX

#### C++

```
HRESULT get_NodeMaxUX(long*);
```

#### C#

```
public long NodeMaxUX { get; }
```

#### Visual Basic

```
Public ReadOnly NodeMaxUX As long
```

#### Version

Available since version 9.7.

### VI.6.2.8 NodeMaxUY

#### C++

```
HRESULT get_NodeMaxUY(long*);
```

#### C#

```
public long NodeMaxUY { get; }
```

#### Visual Basic

```
Public ReadOnly NodeMaxUY As long
```

#### Version

Available since version 9.7.

### VI.6.2.9 NodeMinUX

#### C++

```
HRESULT get_NodeMinUX(long*);
```

#### C#

```
public long NodeMinUX { get; }
```

#### Visual Basic

```
Public ReadOnly NodeMinUX As long
```

#### Version

Available since version 9.7.

### VI.6.2.10 NodeMinUY

#### C++

```
HRESULT get_NodeMinUY(long*);
```

#### C#

```
public long NodeMinUY { get; }
```

#### Visual Basic

```
Public ReadOnly NodeMinUY As long
```

#### Version

Available since version 9.7.

## VI.7 IRobotStoreyResultServer

#### Class Hierarchy

#### C++

```
interface IRobotStoreyResultServer : IDispatch;
```

#### C#

```
public interface IRobotStoreyResultServer;
```

#### Visual Basic

```
Public Interface IRobotStoreyResultServer
```

#### Description

Server of results for stories.

#### Version

Available since version 9.7.

### VI.7.1 IRobotStoreyResultServer Members

The following tables list the members exposed by IRobotStoreyResultServer.

#### Public Methods

	Name	Description
	Displacements (see page 1073)	Function returns values of displacements for the specified story and load case.

	ReducedForces (see page 1073)	Function returns a set of reduced forces for the specified story and load case.
	Values (see page 1074)	Function returns a set of values for the specified story and load case.

## VI.7.2 IRobotStoreyResultServer Methods

The methods of the IRobotStoreyResultServer class are listed here.

### Public Methods

	Name	Description
	Displacements (see page 1073)	Function returns values of displacements for the specified story and load case.
	ReducedForces (see page 1073)	Function returns a set of reduced forces for the specified story and load case.
	Values (see page 1074)	Function returns a set of values for the specified story and load case.

### VI.7.2.1 Displacements

#### C++

```
HRESULT Displacements(long _storey, long _case, IRobotStoreyDisplacements** ret);
```

#### C#

```
public IRobotStoreyDisplacements Displacements(long _storey, long _case);
```

#### Visual Basic

```
Public Function Displacements(_storey As long, _case As long) As IRobotStoreyDisplacements
```

#### Description

Function returns values of displacements for the specified story and load case.

#### Version

Available since version 9.7.

### VI.7.2.2 ReducedForces

#### C++

```
HRESULT ReducedForces(long _storey, long _case, IRobotStoreyReducedForces** ret);
```

#### C#

```
public IRobotStoreyReducedForces ReducedForces(long _storey, long _case);
```

#### Visual Basic

```
Public Function ReducedForces(_storey As long, _case As long) As IRobotStoreyReducedForces
```

#### Description

Function returns a set of reduced forces for the specified story and load case.

#### Version

Available since version 9.7.

### VI.7.2.3 Values

#### C++

```
HRESULT Values(long _storey, long _case, IRobotStoreyValues** ret);
```

**C#**

```
public IRobotStoreyValues Values(long _storey, long _case);
```

**Visual Basic**

```
Public Function Values(_storey As Long, _case As Long) As IRobotStoreyValues
```

**Description**

Function returns a set of values for the specified story and load case.

**Version**

Available since version 9.7.

## VII Quantity survey

Available since version 7.5.

**Interfaces**

	<b>Name</b>	<b>Description</b>
↳	IRobotMaterialQuantitySurvey (↗ see page 1074)	
↳	IRobotBarSectionQuantitySurvey (↗ see page 1077)	
↳	IRobotThicknessQuantitySurvey (↗ see page 1080)	
↳	IRobotStructureQuantitySurvey (↗ see page 1083)	Quantity survey of a structure.

### VII.1 IRobotMaterialQuantitySurvey

**Class Hierarchy****C++**

```
interface IRobotMaterialQuantitySurvey : IDispatch;
```

**C#**

```
public interface IRobotMaterialQuantitySurvey;
```

**Visual Basic**

```
Public Interface IRobotMaterialQuantitySurvey
```

**Version**

Available since version 7.5.

#### VII.1.1 IRobotMaterialQuantitySurvey Members

The following tables list the members exposed by IRobotMaterialQuantitySurvey.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (↗ see page 1075)	Number of all different materials used in a structure.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	GetName (↗ see page 1075)	Function returns a material name.

	GetType ( <a href="#">see page 1076</a> )	Function returns a type of the given material.
	GetVolume ( <a href="#">see page 1076</a> )	Function returns the total volume of all structure components of the specified type and of the specified material.
	GetWeight ( <a href="#">see page 1076</a> )	Function returns the total weight of all structure components of the specified type and of the specified material.

## VII.1.2 IRobotMaterialQuantitySurvey Fields

The fields of the IRobotMaterialQuantitySurvey class are listed here.

### Public Fields

	Name	Description
	Count ( <a href="#">see page 1075</a> )	Number of all different materials used in a structure.

### VII.1.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of all different materials used in a structure.

#### Version

Available since version 7.5.

## VII.1.3 IRobotMaterialQuantitySurvey Methods

The methods of the IRobotMaterialQuantitySurvey class are listed here.

### Public Methods

	Name	Description
	GetName ( <a href="#">see page 1075</a> )	Function returns a material name.
	GetType ( <a href="#">see page 1076</a> )	Function returns a type of the given material.
	GetVolume ( <a href="#">see page 1076</a> )	Function returns the total volume of all structure components of the specified type and of the specified material.
	GetWeight ( <a href="#">see page 1076</a> )	Function returns the total weight of all structure components of the specified type and of the specified material.

### VII.1.3.1 GetName

#### C++

```
HRESULT GetName(long _idx, BSTR* ret);
```

#### C#

```
public String GetName(long _idx);
```

#### Visual Basic

```
Public Function GetName(_idx As long) As String
```

**Description**

Function returns a material name.

**Version**

Available since version 7.5.

**VII.1.3.2 GetType****C++**

```
HRESULT GetType(VARIANT _idx_or_name, IRobotMaterialType* ret);
```

**C#**

```
public IRobotMaterialType GetType(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetType(_idx_or_name As VARIANT) As IRobotMaterialType
```

**Description**

Function returns a type of the given material.

**Version**

Available since version 7.5.

**VII.1.3.3 GetVolume****C++**

```
HRESULT GetVolume(IRobotObjectType _obj_type, VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetVolume(IRobotObjectType _obj_type, VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetVolume(_obj_type As IRobotObjectType, _idx_or_name As VARIANT) As double
```

**Description**

Function returns the total volume of all structure components of the specified type and of the specified material.

**Version**

Available since version 7.5.

**VII.1.3.4 GetWeight****C++**

```
HRESULT GetWeight(IRobotObjectType _obj_type, VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetWeight(IRobotObjectType _obj_type, VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetWeight(_obj_type As IRobotObjectType, _idx_or_name As VARIANT) As double
```

**Description**

Function returns the total weight of all structure components of the specified type and of the specified material.

## Version

Available since version 7.5.

## VII.2 IRobotBarSectionQuantitySurvey

### Class Hierarchy

#### C++

```
interface IRobotBarSectionQuantitySurvey : IDispatch;
```

#### C#

```
public interface IRobotBarSectionQuantitySurvey;
```

### Visual Basic

```
Public Interface IRobotBarSectionQuantitySurvey
```

## Version

Available since version 7.5.

### VII.2.1 IRobotBarSectionQuantitySurvey Members

The following tables list the members exposed by IRobotBarSectionQuantitySurvey.

#### Public Fields

	Name	Description
◆	Count (see page 1078)	Number of different bar sections used in a structure.

#### Public Methods

	Name	Description
◆	GetLength (see page 1078)	Function returns the total length of all structure members of a selected section.
◆	GetName (see page 1078)	Function returns the name of a section of the given index.
◆	GetPaintArea (see page 1079)	Function returns the total painting area of all structure members of a selected section.
◆	GetUnitWeight (see page 1079)	Function returns a unit weight of a selected section.
◆	GetVolume (see page 1079)	Function returns the total volume of all structure members of a selected section.
◆	GetWeight (see page 1080)	Function returns the total weight of all structure members of a selected section.

### VII.2.2 IRobotBarSectionQuantitySurvey Fields

The fields of the IRobotBarSectionQuantitySurvey class are listed here.

#### Public Fields

	Name	Description
◆	Count (see page 1078)	Number of different bar sections used in a structure.

### VII.2.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of different bar sections used in a structure.

**Version**

Available since version 7.5.

### VII.2.3 IRobotBarSectionQuantitySurvey Methods

The methods of the IRobotBarSectionQuantitySurvey class are listed here.

**Public Methods**

	Name	Description
➊	GetLength (see page 1078)	Function returns the total length of all structure members of a selected section.
➋	GetName (see page 1078)	Function returns the name of a section of the given index.
➌	GetPaintArea (see page 1079)	Function returns the total painting area of all structure members of a selected section.
➍	GetUnitWeight (see page 1079)	Function returns a unit weight of a selected section.
➎	GetVolume (see page 1079)	Function returns the total volume of all structure members of a selected section.
➏	GetWeight (see page 1080)	Function returns the total weight of all structure members of a selected section.

#### VII.2.3.1 GetLength

**C++**

```
HRESULT GetLength(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetLength(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetLength(_idx_or_name As VARIANT) As Double
```

**Description**

Function returns the total length of all structure members of a selected section.

**Version**

Available since version 7.5.

#### VII.2.3.2 GetName

**C++**

```
HRESULT GetName(long _idx, BSTR* ret);
```

**C#**

```
public String GetName(long _idx);
```

**Visual Basic**

```
Public Function GetName(_idx As long) As String
```

**Description**

Function returns the name of a section of the given index.

**Version**

Available since version 7.5.

**VII.2.3.3 GetPaintArea****C++**

```
HRESULT GetPaintArea(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetPaintArea(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetPaintArea(_idx_or_name As VARIANT) As double
```

**Description**

Function returns the total painting area of all structure members of a selected section.

**Version**

Available since version 7.5.

**VII.2.3.4 GetUnitWeight****C++**

```
HRESULT GetUnitWeight(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetUnitWeight(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetUnitWeight(_idx_or_name As VARIANT) As double
```

**Description**

Function returns a unit weight of a selected section.

**Version**

Available since version 7.5.

**VII.2.3.5 GetVolume****C++**

```
HRESULT GetVolume(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetVolume(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetVolume(_idx_or_name As VARIANT) As double
```

**Description**

Function returns the total volume of all structure members of a selected section.

**Version**

Available since version 7.5.

**VII.2.3.6 GetWeight****C++**

```
HRESULT GetWeight(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetWeight(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetWeight(_idx_or_name As VARIANT) As double
```

**Description**

Function returns the total weight of all structure members of a selected section.

**Version**

Available since version 7.5.

**VII.3 IRobotThicknessQuantitySurvey****Class Hierarchy****C++**

```
interface IRobotThicknessQuantitySurvey : IDispatch;
```

**C#**

```
public interface IRobotThicknessQuantitySurvey;
```

**Visual Basic**

```
Public Interface IRobotThicknessQuantitySurvey
```

**Version**

Available since version 7.5.

**VII.3.1 IRobotThicknessQuantitySurvey Members**

The following tables list the members exposed by IRobotThicknessQuantitySurvey.

**Public Fields**

	<b>Name</b>	<b>Description</b>
➊	Count (see page 1081)	Number of different panel thicknesses used in a structure.

**Public Methods**

	<b>Name</b>	<b>Description</b>
➌	GetArea (see page 1081)	Function returns the total area of all structure panels of a selected thickness.
➌	GetName (see page 1082)	Function returns a thickness name of the given index.
➌	GetUnitWeight (see page 1082)	Function returns a unit weight for a selected thickness.

	GetVolume (see page 1082)	Function returns the total volume of all structure panels of a selected thickness.
	GetWeight (see page 1083)	Function returns the total weight of all structure panels of a selected thickness.

## VII.3.2 IRobotThicknessQuantitySurvey Fields

The fields of the IRobotThicknessQuantitySurvey class are listed here.

### Public Fields

	Name	Description
	Count (see page 1081)	Number of different panel thicknesses used in a structure.

### VII.3.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of different panel thicknesses used in a structure.

#### Version

Available since version 7.5.

## VII.3.3 IRobotThicknessQuantitySurvey Methods

The methods of the IRobotThicknessQuantitySurvey class are listed here.

### Public Methods

	Name	Description
	GetArea (see page 1081)	Function returns the total area of all structure panels of a selected thickness.
	GetName (see page 1082)	Function returns a thickness name of the given index.
	GetUnitWeight (see page 1082)	Function returns a unit weight for a selected thickness.
	GetVolume (see page 1082)	Function returns the total volume of all structure panels of a selected thickness.
	GetWeight (see page 1083)	Function returns the total weight of all structure panels of a selected thickness.

### VII.3.3.1 GetArea

#### C++

```
HRESULT GetArea(VARIANT _idx_or_name, double* ret);
```

#### C#

```
public double GetArea(VARIANT _idx_or_name);
```

#### Visual Basic

```
Public Function GetArea(_idx_or_name As VARIANT) As Double
```

**Description**

Function returns the total area of all structure panels of a selected thickness.

**Version**

Available since version 7.5.

**VII.3.3.2 GetName****C++**

```
HRESULT GetName(long _idx, BSTR* ret);
```

**C#**

```
public String GetName(long _idx);
```

**Visual Basic**

```
Public Function GetName(_idx As long) As String
```

**Description**

Function returns a thickness name of the given index.

**Version**

Available since version 7.5.

**VII.3.3.3 GetUnitWeight****C++**

```
HRESULT GetUnitWeight(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetUnitWeight(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetUnitWeight(_idx_or_name As VARIANT) As double
```

**Description**

Function returns a unit weight for a selected thickness.

**Version**

Available since version 7.5.

**VII.3.3.4 GetVolume****C++**

```
HRESULT GetVolume(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetVolume(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetVolume(_idx_or_name As VARIANT) As double
```

**Description**

Function returns the total volume of all structure panels of a selected thickness.

**Version**

Available since version 7.5.

**VII.3.3.5 GetWeight****C++**

```
HRESULT GetWeight(VARIANT _idx_or_name, double* ret);
```

**C#**

```
public double GetWeight(VARIANT _idx_or_name);
```

**Visual Basic**

```
Public Function GetWeight(_idx_or_name As VARIANT) As double
```

**Description**

Function returns the total weight of all structure panels of a selected thickness.

**Version**

Available since version 7.5.

**VII.4 IRobotStructureQuantitySurvey****Class Hierarchy****C++**

```
interface IRobotStructureQuantitySurvey : IDispatch;
```

**C#**

```
public interface IRobotStructureQuantitySurvey;
```

**Visual Basic**

```
Public Interface IRobotStructureQuantitySurvey
```

**Description**

Quantity survey of a structure.

**Version**

Available since version 7.5.

**VII.4.1 IRobotStructureQuantitySurvey Members**

The following tables list the members exposed by IRobotStructureQuantitySurvey.

**Public Fields**

	Name	Description
◆	BarSections (see page 1084)	
◆	Materials (see page 1084)	
◆	PanelThickness (see page 1084)	

**VII.4.2 IRobotStructureQuantitySurvey Fields**

The fields of the IRobotStructureQuantitySurvey class are listed here.

## Public Fields

	Name	Description
◆	BarSections (see page 1084)	
◆	Materials (see page 1084)	
◆	PanelThickness (see page 1084)	

### VII.4.2.1 BarSections

#### C++

```
HRESULT get_BarSections(IRobotBarSectionQuantitySurvey**);
```

#### C#

```
public IRobotBarSectionQuantitySurvey BarSections { get; }
```

#### Visual Basic

```
Public ReadOnly BarSections As IRobotBarSectionQuantitySurvey
```

#### Version

Available since version 7.5.

### VII.4.2.2 Materials

#### C++

```
HRESULT get_Materials(IRobotMaterialQuantitySurvey**);
```

#### C#

```
public IRobotMaterialQuantitySurvey Materials { get; }
```

#### Visual Basic

```
Public ReadOnly Materials As IRobotMaterialQuantitySurvey
```

#### Version

Available since version 7.5.

### VII.4.2.3 PanelThickness

#### C++

```
HRESULT get_PanelThickness(IRobotThicknessQuantitySurvey**);
```

#### C#

```
public IRobotThicknessQuantitySurvey PanelThickness { get; }
```

#### Visual Basic

```
Public ReadOnly PanelThickness As IRobotThicknessQuantitySurvey
```

#### Version

Available since version 7.5.

## VIII IRobotStructure

### Class Hierarchy

#### C++

```
interface IRobotStructure : IDispatch;
```

**C#**

```
public interface IRobotStructure;
```

**Visual Basic**

```
Public Interface IRobotStructure
```

**Description**

RobotStructure represents an entire structure. Particular structure components are represented by appropriate interface components. .

**VIII.1 IRobotStructure Members**

The following tables list the members exposed by IRobotStructure.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Bars ( <a href="#">see page 1086</a> )	Structure bar server .
❖	Cases ( <a href="#">see page 1087</a> )	Load case server .
❖	Edit ( <a href="#">see page 1087</a> )	Set of auxiliary edit functions .
❖	ExtParams ( <a href="#">see page 1087</a> )	External parameter server.
❖	FiniteElems ( <a href="#">see page 1088</a> )	Finite element server.
❖	GroupObjects ( <a href="#">see page 1088</a> )	Server for grouping of structure elements.
❖	Groups ( <a href="#">see page 1088</a> )	Server managing groups of structure components.
❖	IsCalcModelGenerated ( <a href="#">see page 1088</a> )	A flag indicating if the computational model for the structure has been generated and if it is available.
❖	Labels ( <a href="#">see page 1089</a> )	Server of complex attributes, such as node supports, bar sections and others. .
❖	Nodes ( <a href="#">see page 1089</a> )	Structure node server .
❖	Objects ( <a href="#">see page 1089</a> )	Object server .
❖	QuantitySurvey ( <a href="#">see page 1090</a> )	Quantity survey.
❖	Results ( <a href="#">see page 1090</a> )	Server of the currently available results .
❖	ResultsFreeze ( <a href="#">see page 1090</a> )	
❖	Selections ( <a href="#">see page 1090</a> )	Server managing selections (selection factory) .
❖	Storeys ( <a href="#">see page 1091</a> )	Stories defined in a structure.
❖	Type ( <a href="#">see page 1091</a> )	Structure type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	ApplyCache ( <a href="#">see page 1091</a> )	Function inserts data from a specified buffer to a structure and returns information about new structure components generated in this way. .
❖	Clear ( <a href="#">see page 1092</a> )	Function deletes all elements of a structure model.
❖	CreateCache ( <a href="#">see page 1092</a> )	Function creates and returns an empty buffer to which new structure components can be added. To insert a whole component set to a structure, ApplyCache ( <a href="#">see page 1091</a> ) function should be called up. .
❖	ExportXml ( <a href="#">see page 1092</a> )	Function exports selected structure components and calculation results to the indicated xml file.
❖	Merge ( <a href="#">see page 1093</a> )	Function attaches a given data set to the current structure.

**VIII.2 IRobotStructure Fields**

The fields of the IRobotStructure class are listed here.

## Public Fields

	Name	Description
◆	Bars ( <a href="#">see page 1086</a> )	Structure bar server .
◆	Cases ( <a href="#">see page 1087</a> )	Load case server .
◆	Edit ( <a href="#">see page 1087</a> )	Set of auxiliary edit functions .
◆	ExtParams ( <a href="#">see page 1087</a> )	External parameter server.
◆	FiniteElems ( <a href="#">see page 1088</a> )	Finite element server.
◆	GroupObjects ( <a href="#">see page 1088</a> )	Server for grouping of structure elements.
◆	Groups ( <a href="#">see page 1088</a> )	Server managing groups of structure components.
◆	IsCalcModelGenerated ( <a href="#">see page 1088</a> )	A flag indicating if the computational model for the structure has been generated and if it is available.
◆	Labels ( <a href="#">see page 1089</a> )	Server of complex attributes, such as node supports, bar sections and others. .
◆	Nodes ( <a href="#">see page 1089</a> )	Structure node server .
◆	Objects ( <a href="#">see page 1089</a> )	Object server .
◆	QuantitySurvey ( <a href="#">see page 1090</a> )	Quantity survey.
◆	Results ( <a href="#">see page 1090</a> )	Server of the currently available results .
◆	ResultsFreeze ( <a href="#">see page 1090</a> )	
◆	Selections ( <a href="#">see page 1090</a> )	Server managing selections (selection factory) .
◆	Storeys ( <a href="#">see page 1091</a> )	Stories defined in a structure.
◆	Type ( <a href="#">see page 1091</a> )	Structure type.

### VIII.2.1 Bars

#### C++

```
HRESULT get_Bars( IRobotBarServer** );
```

#### C#

```
public IRobotBarServer Bars { get; }
```

#### Visual Basic

```
Public ReadOnly Bars As IRobotBarServer
```

#### Description

Structure bar server .

### VIII.2.2 Cases

#### C++

```
HRESULT get_Cases( IRobotCaseServer** );
```

#### C#

```
public IRobotCaseServer Cases { get; }
```

#### Visual Basic

```
Public ReadOnly Cases As IRobotCaseServer
```

#### Description

Load case server .

### VIII.2.3 Edit

**C++**

```
HRESULT get_Edit(IRobotStructureEditTools**);
```

**C#**

```
public IRobotStructureEditTools Edit { get; }
```

**Visual Basic**

```
Public ReadOnly Edit As IRobotStructureEditTools
```

**Description**

Set of auxiliary edit functions.

**Version**

Available since version 3.5.

### VIII.2.4 ExtParams

**C++**

```
HRESULT get_ExtParams(IRobotParamServer**);
```

**C#**

```
public IRobotParamServer ExtParams { get; }
```

**Visual Basic**

```
Public ReadOnly ExtParams As IRobotParamServer
```

**Description**

External parameter server.

**Version**

Available since version 8.2.

### VIII.2.5 FiniteElems

**C++**

```
HRESULT get_FiniteElems(IRobotFiniteElementServer**);
```

**C#**

```
public IRobotFiniteElementServer FiniteElems { get; }
```

**Visual Basic**

```
Public ReadOnly FiniteElems As IRobotFiniteElementServer
```

**Description**

Finite element server.

**Version**

Available since version 2.5.

### VIII.2.6 GroupObjects

**C++**

```
HRESULT get_GroupObjects(IRobotGroupObjectServer**);
```

**C#**

```
public IRobotGroupObjectServer GroupObjects { get; }
```

**Visual Basic**

```
Public ReadOnly GroupObjects As IRobotGroupObjectServer
```

**Description**

Server for grouping of structure elements.

**Version**

Available since version 7.5.

## VIII.2.7 Groups

**C++**

```
HRESULT get_Groups(IRobotGroupServer**);
```

**C#**

```
public IRobotGroupServer Groups { get; }
```

**Visual Basic**

```
Public ReadOnly Groups As IRobotGroupServer
```

**Description**

Server managing groups of structure components.

**Version**

Available since version 3.

## VIII.2.8 IsCalcModelGenerated

**C++**

```
HRESULT get_IsCalcModelGenerated(VARIANT_BOOL*);
```

**C#**

```
public bool IsCalcModelGenerated { get; }
```

**Visual Basic**

```
Public ReadOnly IsCalcModelGenerated As Boolean
```

**Description**

A flag indicating if the computational model for the structure has been generated and if it is available.

**Version**

Available since version 13.4.

## VIII.2.9 Labels

**C++**

```
HRESULT get_Labels(IRobotLabelServer**);
```

**C#**

```
public IRobotLabelServer Labels { get; }
```

**Visual Basic**

```
Public ReadOnly Labels As IRobotLabelServer
```

**Description**

Server of complex attributes, such as node supports, bar sections and others. .

### VIII.2.10 Nodes

**C++**

```
HRESULT get_Nodes(IRobotNodeServer**);
```

**C#**

```
public IRobotNodeServer Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As IRobotNodeServer
```

**Description**

Structure node server .

### VIII.2.11 Objects

**C++**

```
HRESULT get_Objects(IRobotObjObjectServer**);
```

**C#**

```
public IRobotObjObjectServer Objects { get; }
```

**Visual Basic**

```
Public ReadOnly Objects As IRobotObjObjectServer
```

**Description**

Object server .

### VIII.2.12 QuantitySurvey

**C++**

```
HRESULT get_QuantitySurvey(IRobotStructureQuantitySurvey**);
```

**C#**

```
public IRobotStructureQuantitySurvey QuantitySurvey { get; }
```

**Visual Basic**

```
Public ReadOnly QuantitySurvey As IRobotStructureQuantitySurvey
```

**Description**

Quantity survey.

**Version**

Available since version 7.5.

### VIII.2.13 Results

**C++**

```
HRESULT get_Results(IRobotResultServer**);
```

**C#**

```
public IRobotResultServer Results { get; }
```

**Visual Basic**

```
Public ReadOnly Results As IRobotResultServer
```

**Description**

Server of the currently available results .

**VIII.2.14 ResultsFreeze****C++**

```
HRESULT get_ResultsFreeze(VARIANT_BOOL* );
HRESULT put_ResultsFreeze(VARIANT_BOOL);
```

**C#**

```
public bool ResultsFreeze { get; set; }
```

**Visual Basic**

```
Public ResultsFreeze As Boolean
```

**Version**

Available since version 4.

**VIII.2.15 Selections****C++**

```
HRESULT get_Selections(IRobotSelectionFactory** );
```

**C#**

```
public IRobotSelectionFactory Selections { get; }
```

**Visual Basic**

```
Public ReadOnly Selections As IRobotSelectionFactory
```

**Description**

Server managing selections (selection factory) .

**VIII.2.16 Storeys****C++**

```
HRESULT get_Storeys(IRobotStoreyMngr** );
```

**C#**

```
public IRobotStoreyMngr Storeys { get; }
```

**Visual Basic**

```
Public ReadOnly Storeys As IRobotStoreyMngr
```

**Description**

Stories defined in a structure.

**VIII.2.17 Type****C++**

```
HRESULT get_Type(IRobotProjectType* );
```

**C#**

```
public IRobotProjectType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotProjectType
```

**Description**

Structure type.

**Version**

Available since version 8.

## VIII.3 IRobotStructure Methods

The methods of the IRobotStructure class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	ApplyCache (see page 1091)	Function inserts data from a specified buffer to a structure and returns information about new structure components generated in this way. .
💡	Clear (see page 1092)	Function deletes all elements of a structure model.
💡	CreateCache (see page 1092)	Function creates and returns an empty buffer to which new structure components can be added. To insert a whole component set to a structure, ApplyCache (see page 1091) function should be called up. .
💡	ExportXml (see page 1092)	Function exports selected structure components and calculation results to the indicated xml file.
💡	Merge (see page 1093)	Function attaches a given data set to the current structure.

### VIII.3.1 ApplyCache

**C++**

```
HRESULT ApplyCache(IRobotStructureCache* _struct_cache, IRobotStructureApplyInfo** ret);
```

**C#**

```
public IRobotStructureApplyInfo ApplyCache(IRobotStructureCache _struct_cache);
```

**Visual Basic**

```
Public Function ApplyCache(ByRef _struct_cache As IRobotStructureCache) As IRobotStructureApplyInfo
```

**Description**

Function inserts data from a specified buffer to a structure and returns information about new structure components generated in this way. .

**Version**

Available since version 3.5.

### VIII.3.2 Clear

**C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

Function deletes all elements of a structure model.

**VIII.3.3 CreateCache****C++**

```
HRESULT CreateCache(IRobotStructureCache** ret);
```

**C#**

```
public IRobotStructureCache CreateCache();
```

**Visual Basic**

```
Public Function CreateCache() As IRobotStructureCache
```

**Description**

Function creates and returns an empty buffer to which new structure components can be added. To insert a whole component set to a structure, ApplyCache (see page 1091) function should be called up..

**Version**

Available since version 3.5.

**VIII.3.4 ExportXml****C++**

```
HRESULT ExportXml(BSTR _input_xml, BSTR _output_xml);
```

**C#**

```
public void ExportXml(String _input_xml, String _output_xml);
```

**Visual Basic**

```
Public Sub ExportXml(_input_xml As String, _output_xml As String)
```

**Description**

Function exports selected structure components and calculation results to the indicated xml file.

**Version**

Available since version 4.1.

**VIII.3.5 Merge****C++**

```
HRESULT Merge(IRobotStructureMergeData* _data, IUnknown _params);
```

**C#**

```
public void Merge(IRobotStructureMergeData _data, IUnknown _params);
```

**Visual Basic**

```
Public Sub Merge(ByRef _data As IRobotStructureMergeData, _params As IUnknown)
```

**Description**

Function attaches a given data set to the current structure.

**Version**

Available since version 8.

## IX IRobotStructureCache

### Class Hierarchy

#### C++

```
interface IRobotStructureCache : IDispatch;
```

#### C#

```
public interface IRobotStructureCache;
```

### Visual Basic

```
Public Interface IRobotStructureCache
```

### Description

Interface enabling buffered access to a structure. It enables adding quickly a whole component set to a structure. To add a whole data set to a structure, ApplyCache function of RobotStructure interface should be called up. .

### Version

Available since version 3.5.

## IX.1 IRobotStructureCache Members

The following tables list the members exposed by IRobotStructureCache.

### Public Methods

	Name	Description
≡	AddBar (see page 1094)	Function adds a new bar to the buffer.
≡	AddNode (see page 1094)	Function adds a new node to the buffer.
≡	EnsureNodeExist (see page 1095)	Function checks if a node exists at the position of defined coordinates. If yes, then function returns its number. If no, then function creates a new node and returns its number.
≡	SetBarLabel (see page 1095)	Function enables assigning a specified label to an earlier-generated bar.
≡	SetBarName (see page 1095)	Function sets a name schema for the specified bar.

## IX.2 IRobotStructureCache Methods

The methods of the IRobotStructureCache class are listed here.

### Public Methods

	Name	Description
≡	AddBar (see page 1094)	Function adds a new bar to the buffer.
≡	AddNode (see page 1094)	Function adds a new node to the buffer.
≡	EnsureNodeExist (see page 1095)	Function checks if a node exists at the position of defined coordinates. If yes, then function returns its number. If no, then function creates a new node and returns its number.
≡	SetBarLabel (see page 1095)	Function enables assigning a specified label to an earlier-generated bar.
≡	SetBarName (see page 1095)	Function sets a name schema for the specified bar.

### IX.2.1 AddBar

#### C++

```
HRESULT AddBar(long _number, long _start_node, long _end_node, BSTR _sect_name, BSTR
_mater_name, double _gamma = 0);
```

**C#**

```
public void AddBar(long _number, long _start_node, long _end_node, String _sect_name,
String _mater_name, double _gamma = 0);
```

**Visual Basic**

```
Public Sub AddBar(_number As long, _start_node As long, _end_node As long, _sect_name As
String, _mater_name As String, Optional _gamma As double = 0)
```

**Description**

Function adds a new bar to the buffer.

**Version**

Available since version 3.5.

**IX.2.2 AddNode****C++**

```
HRESULT AddNode(long _number, double _x, double _y, double _z);
```

**C#**

```
public void AddNode(long _number, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub AddNode(_number As long, _x As double, _y As double, _z As double)
```

**Description**

Function adds a new node to the buffer.

**Version**

Available since version 3.5.

**IX.2.3 EnsureNodeExist****C++**

```
HRESULT EnsureNodeExist(double _x, double _y, double _z, long* ret);
```

**C#**

```
public long EnsureNodeExist(double _x, double _y, double _z);
```

**Visual Basic**

```
Public Function EnsureNodeExist(_x As double, _y As double, _z As double) As long
```

**Description**

Function checks if a node exists at the position of defined coordinates. If yes, then function returns its number. If no, then function creates a new node and returns its number.

**IX.2.4 SetBarLabel****C++**

```
HRESULT SetBarLabel(long _bar, IRobotLabelType _lab_type, BSTR _lab_name, long _ext_par =
0);
```

**C#**

```
public void SetBarLabel(long _bar, IRobotLabelType _lab_type, String _lab_name, long _ext_par = 0);
```

**Visual Basic**

```
Public Sub SetBarLabel(_bar As long, _lab_type As IRobotLabelType, _lab_name As String,  
Optional _ext_par As long = 0)
```

**Description**

Function enables assigning a specified label to an earlier-generated bar.

**Version**

Available since version 4.1.

**IX.2.5 SetBarName****C++**

```
HRESULT SetBarName(long _bar_num, BSTR _name_tmpl);
```

**C#**

```
public void SetBarName(long _bar_num, String _name_tmpl);
```

**Visual Basic**

```
Public Sub SetBarName(_bar_num As long, _name_tmpl As String)
```

**Description**

Function sets a name schema for the specified bar.

**Version**

Available since version 8.2.

**X IRobotStructureApplyInfo****Class Hierarchy****C++**

```
interface IRobotStructureApplyInfo : IDispatch;
```

**C#**

```
public interface IRobotStructureApplyInfo;
```

**Visual Basic**

```
Public Interface IRobotStructureApplyInfo
```

**Description**

Interface providing access to information about structure components generated as a result of inserting a new data set to a structure..

**Version**

Available since version 3.5.

**X.1 IRobotStructureApplyInfo Members**

The following tables list the members exposed by IRobotStructureApplyInfo.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Bars (see page 1096)	Table with bar numbers.
◆	Nodes (see page 1097)	Table with node numbers.

## X.2 IRobotStructureApplyInfo Fields

The fields of the IRobotStructureApplyInfo class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Bars (see page 1096)	Table with bar numbers.
◆	Nodes (see page 1097)	Table with node numbers.

### X.2.1 Bars

**C++**

```
HRESULT get_Bars( IRobotNumbersArray** );
```

**C#**

```
public IRobotNumbersArray Bars { get; }
```

**Visual Basic**

```
Public ReadOnly Bars As IRobotNumbersArray
```

**Description**

Table with bar numbers.

**Version**

Available since version 3.5.

### X.2.2 Nodes

**C++**

```
HRESULT get_Nodes( IRobotNumbersArray** );
```

**C#**

```
public IRobotNumbersArray Nodes { get; }
```

**Visual Basic**

```
Public ReadOnly Nodes As IRobotNumbersArray
```

**Description**

Table with node numbers.

**Version**

Available since version 3.5.

## XI IRobotStructureMergeData

**Class Hierarchy****C++**

```
interface IRobotStructureMergeData : IDispatch;
```

**C#**

```
public interface IRobotStructureMergeData;
```

**Visual Basic**

```
Public Interface IRobotStructureMergeData
```

**Description**

Interface defining data that may be merged into a structure by means of the Merge function.

**Version**

Available since version 8.

## XI.1 IRobotStructureMergeData Members

The following tables list the members exposed by IRobotStructureMergeData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Structure ( <a href="#">see page 1098</a> )	Auxiliary structure.

**Public Methods**

	<b>Name</b>	<b>Description</b>
+=	CreateStructure ( <a href="#">see page 1098</a> )	Function creates an empty auxiliary structure of the specified type.
+=	LoadStructure ( <a href="#">see page 1099</a> )	Function loads an auxiliary structure from the specified RTD file.

## XI.2 IRobotStructureMergeData Fields

The fields of the IRobotStructureMergeData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Structure ( <a href="#">see page 1098</a> )	Auxiliary structure.

### XI.2.1 Structure

**C++**

```
HRESULT get_Structure(IRobotStructure**);
```

**C#**

```
public IRobotStructure Structure { get; }
```

**Visual Basic**

```
Public ReadOnly Structure As IRobotStructure
```

**Description**

Auxiliary structure.

**Version**

Available since version 8.

## XI.3 IRobotStructureMergeData Methods

The methods of the IRobotStructureMergeData class are listed here.

## Public Methods

	Name	Description
💡	CreateStructure (see page 1098)	Function creates an empty auxiliary structure of the specified type.
💡	LoadStructure (see page 1099)	Function loads an auxiliary structure from the specified RTD file.

### XI.3.1 CreateStructure

#### C++

```
HRESULT CreateStructure(IRobotProjectType _type, VARIANT_BOOL* ret);
```

#### C#

```
public bool CreateStructure(IRobotProjectType _type);
```

#### Visual Basic

```
Public Function CreateStructure(_type As IRobotProjectType) As Boolean
```

#### Description

Function creates an empty auxiliary structure of the specified type.

#### Version

Available since version 8.

### XI.3.2 LoadStructure

#### C++

```
HRESULT LoadStructure(BSTR _rtt_fpath, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadStructure(String _rtt_fpath);
```

#### Visual Basic

```
Public Function LoadStructure(_rtt_fpath As String) As Boolean
```

#### Description

Function loads an auxiliary structure from the specified RTD file.

#### Version

Available since version 8.

## XII IRobotStructureEvents

### Class Hierarchy

#### C++

```
interface IRobotStructureEvents : IDispatch;
```

#### C#

```
public interface IRobotStructureEvents;
```

#### Visual Basic

```
Public Interface IRobotStructureEvents
```

**Description**

Events of RobotStructure object.

**Version**

Available since version 15.2.

## XII.1 IRobotStructureEvents Members

The following tables list the members exposed by IRobotStructureEvents.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	ResultStatusChanged (↗ see page 1100)	This event is triggered when the status of results is changed. A new status is passed as a parameter.

## XII.2 IRobotStructureEvents Methods

The methods of the IRobotStructureEvents class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	ResultStatusChanged (↗ see page 1100)	This event is triggered when the status of results is changed. A new status is passed as a parameter.

### XII.2.1 ResultStatusChanged

**C++**

```
HRESULT ResultStatusChanged( IRobotResultStatusType _new_status);
```

**C#**

```
public void ResultStatusChanged( IRobotResultStatusType _new_status);
```

**Visual Basic**

```
Public Sub ResultStatusChanged(_new_status As IRobotResultStatusType)
```

**Description**

This event is triggered when the status of results is changed. A new status is passed as a parameter.

**Version**

Available since version 15.2.

---

# Project

Robot Object Model allows one to manage an entire project (task) saved to an RTD-format file. .

**Enumerations**

	<b>Name</b>	<b>Description</b>
💡	IRobotProjectType (↗ see page 1333)	A set of identifiers is defined to refer to structure types recognized by Robot.
💡	IRobotProjectSaveFormat (↗ see page 1349)	Available formats for project saving.

	IRobotActiveModelType (see page 1351)	.
-----------------------------------------------------------------------------------	---------------------------------------	---

## Interfaces

	Name	Description
	IRobotProject (see page 1334)	RobotProject interface allows for managing all the data saved in an RTD-format file. .
	IRobotFileInsertParams (see page 1346)	The process of inserting a structure, defined in an external file of a non-RTD format, is carried out according to parameters determining the insertion mode. The RobotInsertParams interface provides access to the parameters determining the manner of inserting the structure into the current project. .
	IRobotProjectEvents (see page 1350)	Events generated by the RobotProject object.

# Project components

A project may contain many components of different types. They are stored in a tree-like hierarchical structure and managed by an appropriate component manager. .

## Enumerations

	Name	Description
	IRobotProjectComponentType (see page 1261)	A set of identifiers is defined to describe different types of components that may be saved in a project. .

## Interfaces

	Name	Description
	IRobotProjectComponent (see page 1262)	Common interface granting access to any project component. .
	IRobotProjectComponentMngr (see page 1263)	Manager of project components manages tree-like structures containing all components defined in a project. Project components are divided into different levels. The manager makes available a list of all levels that are indexed with numbers from 1 to the number of levels. Additionally, each level has its name. Project components may be taken by levels. It is possible to take from the project all the components of an indicated type from the level of the indicated name. .

## I.1 Spread footing

A set of interfaces allowing access to a component of the 'spread footing' project. .

### Version

Available since version 3.

## Enumerations

	Name	Description
	IRConcrFootingDimType (see page 1113)	A set of values identifying individual spread footing dimensions. .
	IRConcrFootingShapeType (see page 1114)	A set of values defining the spread footing shape. .
	IRConcrFootingType (see page 1114)	A set of values defining the foundation type. .
	IRConcrFootResultType (see page 1122)	
	IRConcrFootingPierType (see page 1123)	

## Interfaces

	Name	Description
↳	IRConcrFooting (see page 1101)	Interface representing the spread footing.
↳	IRConcrFootingGeometry (see page 1110)	Interface representing the spread footing geometry.
↳	IRConcrFootingGround (see page 1115)	Interface representing soils and reference level of the footing.
↳	IRConcrFootingLoads (see page 1116)	Interface representing spread footing loads and backfill loads. .
↳	IRConcrFootingResults (see page 1117)	Interface representing calculation results for the spread footing. .
↳	IRConcrFootingCalculationOptions (see page 1118)	
↳	IRConcrFootingPattern (see page 1120)	

### I.1.1 IRConcrFooting

#### Class Hierarchy

#### C++

```
interface IRConcrFooting : IDispatch;
```

#### C#

```
public interface IRConcrFooting;
```

#### Visual Basic

```
Public Interface IRConcrFooting
```

#### Description

Interface representing the spread footing.

#### Version

Available since version 3.

#### I.1.1.1 IRConcrFooting Members

The following tables list the members exposed by IRConcrFooting.

#### Public Fields

	Name	Description
◆	CalculationOptions (see page 1103)	
◆	Concrete (see page 1103)	
◆	FootRotation (see page 1104)	
◆	Geometry (see page 1104)	Footing geometry.
◆	Ground (see page 1104)	Soils under the spread footing.
◆	IsActive (see page 1105)	
◆	IsSelected (see page 1105)	
◆	Loads (see page 1105)	Spread footing loads and backfill loads .
◆	Name (see page 1105)	
◆	NumberOfElements (see page 1106)	
◆	Pattern (see page 1106)	

◆	Reinforcement ( <a href="#">see page 1106</a> )	
◆	Results ( <a href="#">see page 1106</a> )	Analysis results for spread footing.
◆	Steel ( <a href="#">see page 1107</a> )	
◆	StructureUserNo ( <a href="#">see page 1107</a> )	
◆	StructureUserNoCount ( <a href="#">see page 1107</a> )	
◆	UniqueId ( <a href="#">see page 1107</a> )	

## Public Methods

	Name	Description
≡	Activate ( <a href="#">see page 1108</a> )	
≡	Calculate ( <a href="#">see page 1108</a> )	
≡	CreateCalculationNoteRtf ( <a href="#">see page 1109</a> )	
≡	CreateFromNodes ( <a href="#">see page 1109</a> )	
≡	CreateReinforcement ( <a href="#">see page 1109</a> )	
≡	DisplayDrawing ( <a href="#">see page 1109</a> )	
≡	Save ( <a href="#">see page 1110</a> )	
≡	Verify ( <a href="#">see page 1110</a> )	

### I.1.1.2 IRConcrFooting Fields

The fields of the IRConcrFooting class are listed here.

## Public Fields

	Name	Description
◆	CalculationOptions ( <a href="#">see page 1103</a> )	
◆	Concrete ( <a href="#">see page 1103</a> )	
◆	FootRotation ( <a href="#">see page 1104</a> )	
◆	Geometry ( <a href="#">see page 1104</a> )	Footing geometry.
◆	Ground ( <a href="#">see page 1104</a> )	Soils under the spread footing.
◆	IsActive ( <a href="#">see page 1105</a> )	
◆	IsSelected ( <a href="#">see page 1105</a> )	
◆	Loads ( <a href="#">see page 1105</a> )	Spread footing loads and backfill loads .
◆	Name ( <a href="#">see page 1105</a> )	
◆	NumberOfElements ( <a href="#">see page 1106</a> )	
◆	Pattern ( <a href="#">see page 1106</a> )	
◆	Reinforcement ( <a href="#">see page 1106</a> )	
◆	Results ( <a href="#">see page 1106</a> )	Analysis results for spread footing.
◆	Steel ( <a href="#">see page 1107</a> )	
◆	StructureUserNo ( <a href="#">see page 1107</a> )	
◆	StructureUserNoCount ( <a href="#">see page 1107</a> )	
◆	UniqueId ( <a href="#">see page 1107</a> )	

#### I.1.1.2.1 CalculationOptions

### C++

```
HRESULT get_CalculationOptions(IRConcrFootingCalculationOptions**);
```

**C#**

```
public IRConcrFootingCalculationOptions CalculationOptions { get; }
```

**Visual Basic**

```
Public ReadOnly CalculationOptions As IRConcrFootingCalculationOptions
```

**Version**

Available since version 10.

**I.1.1.2.2 Concrete****C++**

```
HRESULT get_Concrete(IRConcrConcrete**);
```

**C#**

```
public IRConcrConcrete Concrete { get; }
```

**Visual Basic**

```
Public ReadOnly Concrete As IRConcrConcrete
```

**Version**

Available since version 10.

**I.1.1.2.3 FootRotation****C++**

```
HRESULT get_FootRotation(double*);  
HRESULT put_FootRotation(double);
```

**C#**

```
public double FootRotation { get; set; }
```

**Visual Basic**

```
Public FootRotation As Double
```

**Version**

Available since version 11.

**I.1.1.2.4 Geometry****C++**

```
HRESULT get_Geometry(IRConcrFootingGeometry**);
```

**C#**

```
public IRConcrFootingGeometry Geometry { get; }
```

**Visual Basic**

```
Public ReadOnly Geometry As IRConcrFootingGeometry
```

**Description**

Footing geometry.

**Version**

Available since version 3.

### I.1.1.2.5 Ground

#### C++

```
HRESULT get_Ground( IRConcrFootingGround** );
```

#### C#

```
public IRConcrFootingGround Ground { get; }
```

#### Visual Basic

```
Public ReadOnly Ground As IRConcrFootingGround
```

#### Description

Soils under the spread footing.

#### Version

Available since version 3.

### I.1.1.2.6 IsActive

#### C++

```
HRESULT get_IsActive(VARIANT_BOOL* );
```

#### C#

```
public bool IsActive { get; }
```

#### Visual Basic

```
Public ReadOnly IsActive As Boolean
```

#### Version

Available since version 10.

### I.1.1.2.7 IsSelected

#### C++

```
HRESULT get_IsSelected(VARIANT_BOOL* );
HRESULT put_IsSelected(VARIANT_BOOL);
```

#### C#

```
public bool IsSelected { get; set; }
```

#### Visual Basic

```
Public IsSelected As Boolean
```

#### Version

Available since version 12.

### I.1.1.2.8 Loads

#### C++

```
HRESULT get_Loads( IRConcrFootingLoads** );
```

#### C#

```
public IRConcrFootingLoads Loads { get; }
```

#### Visual Basic

```
Public ReadOnly Loads As IRConcrFootingLoads
```

**Description**

Spread footing loads and backfill loads .

**Version**

Available since version 3.

**I.1.1.2.9 Name****C++**

```
HRESULT get_Name(BSTR* );
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

**I.1.1.2.10 NumberOfElements****C++**

```
HRESULT get_NumberOfElements( long* );
HRESULT put_NumberOfElements( long );
```

**C#**

```
public long NumberOfElements { get; set; }
```

**Visual Basic**

```
Public NumberOfElements As Long
```

**Version**

Available since version 11.

**I.1.1.2.11 Pattern****C++**

```
HRESULT get_Pattern(IRConcrFootingPattern** );
```

**C#**

```
public IRConcrFootingPattern Pattern { get; }
```

**Visual Basic**

```
Public ReadOnly Pattern As IRConcrFootingPattern
```

**Version**

Available since version 10.

**I.1.1.2.12 Reinforcement****C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement** );
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

**I.1.1.2.13 Results****C++**

```
HRESULT get_Results(IRConcrFootingResults**);
```

**C#**

```
public IRConcrFootingResults Results { get; }
```

**Visual Basic**

```
Public ReadOnly Results As IRConcrFootingResults
```

**Description**

Analysis results for spread footing.

**Version**

Available since version 3.

**I.1.1.2.14 Steel****C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

**I.1.1.2.15 StructureUserNo****C++**

```
HRESULT get_StructureUserNo(long*);  
HRESULT put_StructureUserNo(long);
```

**C#**

```
public long StructureUserNo { get; set; }
```

**Visual Basic**

```
Public StructureUserNo As long
```

**Version**

Available since version 10.

**I.1.1.2.16 StructureUserNoCount****C++**

```
HRESULT get_StructureUserNoCount(short*);  
HRESULT put_StructureUserNoCount(short);
```

**C#**

```
public short StructureUserNoCount { get; set; }
```

**Visual Basic**

```
Public StructureUserNoCount As short
```

**Version**

Available since version 10.

**I.1.1.2.17 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Version**

Available since version 10.

**I.1.1.3 IRConcrFooting Methods**

The methods of the IRConcrFooting class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Activate (see page 1108)	
≡	Calculate (see page 1108)	
≡	CreateCalculationNoteRtf (see page 1109)	
≡	CreateFromNodes (see page 1109)	
≡	CreateReinforcement (see page 1109)	
≡	DisplayDrawing (see page 1109)	
≡	Save (see page 1110)	
≡	Verify (see page 1110)	

**I.1.1.3.1 Activate****C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

### I.1.1.3.2 Calculate

**C++**

```
HRESULT Calculate(VARIANT_BOOL _createReinforcement, VARIANT_BOOL* ret);
```

**C#**

```
public bool Calculate(bool _createReinforcement);
```

**Visual Basic**

```
Public Function Calculate(_createReinforcement As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.1.1.3.3 CreateCalculationNoteRtf

**C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

### I.1.1.3.4 CreateFromNodes

**C++**

```
HRESULT CreateFromNodes(BSTR _selText, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromNodes(String _selText);
```

**Visual Basic**

```
Public Function CreateFromNodes(_selText As String) As Boolean
```

**Version**

Available since version 10.

### I.1.1.3.5 CreateReinforcement

**C++**

```
HRESULT CreateReinforcement();
```

**C#**

```
public void CreateReinforcement();
```

**Visual Basic**

```
Public Sub CreateReinforcement()
```

**Version**

Available since version 10.

**I.1.1.3.6 DisplayDrawing****C++**

```
HRESULT DisplayDrawing();
```

**C#**

```
public void DisplayDrawing();
```

**Visual Basic**

```
Public Sub DisplayDrawing()
```

**Version**

Available since version 10.

**I.1.1.3.7 Save****C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.1.1.3.8 Verify****C++**

```
HRESULT Verify(VARIANT_BOOL _showErrors, long* ret);
```

**C#**

```
public long Verify(bool _showErrors);
```

**Visual Basic**

```
Public Function Verify(_showErrors As Boolean) As long
```

**Version**

Available since version 10.

**I.1.2 IRConcrFootingGeometry****Class Hierarchy****C++**

```
interface IRConcrFootingGeometry : IDispatch;
```

**C#**

```
public interface IRConcrFootingGeometry;
```

## Visual Basic

```
Public Interface IRConcrFootingGeometry
```

### Description

Interface representing the spread footing geometry.

### Version

Available since version 3.

### I.1.2.1 IRConcrFootingGeometry Members

The following tables list the members exposed by IRConcrFootingGeometry.

#### Public Fields

	Name	Description
❖	ConcreteVolume ( <a href="#">see page 1111</a> )	
❖	PierType ( <a href="#">see page 1111</a> )	
❖	Shape ( <a href="#">see page 1112</a> )	Shape of the spread footing section.
❖	ShutteringArea ( <a href="#">see page 1112</a> )	
❖	Type ( <a href="#">see page 1112</a> )	Foundation type.

#### Public Methods

	Name	Description
♫	GetDim ( <a href="#">see page 1113</a> )	Function returns the value of a relevant spread footing dimension. .
♫	SetDim ( <a href="#">see page 1113</a> )	Function sets the value of a relevant spread footing dimension. .

### I.1.2.2 IRConcrFootingGeometry Fields

The fields of the IRConcrFootingGeometry class are listed here.

#### Public Fields

	Name	Description
❖	ConcreteVolume ( <a href="#">see page 1111</a> )	
❖	PierType ( <a href="#">see page 1111</a> )	
❖	Shape ( <a href="#">see page 1112</a> )	Shape of the spread footing section.
❖	ShutteringArea ( <a href="#">see page 1112</a> )	
❖	Type ( <a href="#">see page 1112</a> )	Foundation type.

### I.1.2.2.1 ConcreteVolume

#### C++

```
HRESULT get_ConcreteVolume(double*);
```

#### C#

```
public double ConcreteVolume { get; }
```

## Visual Basic

```
Public ReadOnly ConcreteVolume As Double
```

### Version

Available since version 10.

### I.1.2.2.2 PierType

#### C++

```
HRESULT get_PierType(IRConcrFootingPierType* );
HRESULT put_PierType(IRConcrFootingPierType);
```

#### C#

```
public IRConcrFootingPierType PierType { get; set; }
```

#### Visual Basic

```
Public PierType As IRConcrFootingPierType
```

#### Version

Available since version 11.

### I.1.2.2.3 Shape

#### C++

```
HRESULT get_Shape(IRConcrFootingShapeType* );
HRESULT put_Shape(IRConcrFootingShapeType);
```

#### C#

```
public IRConcrFootingShapeType Shape { get; set; }
```

#### Visual Basic

```
Public Shape As IRConcrFootingShapeType
```

#### Description

Shape of the spread footing section.

#### Version

Available since version 3.

### I.1.2.2.4 ShutteringArea

#### C++

```
HRESULT get_ShutteringArea(double* );
```

#### C#

```
public double ShutteringArea { get; }
```

#### Visual Basic

```
Public ReadOnly ShutteringArea As double
```

#### Version

Available since version 10.

### I.1.2.2.5 Type

#### C++

```
HRESULT get_Type(IRConcrFootingType* );
HRESULT put_Type(IRConcrFootingType);
```

#### C#

```
public IRConcrFootingType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRConcrFootingType
```

**Description**

Foundation type.

**Version**

Available since version 3.

**I.1.2.3 IRConcrFootingGeometry Methods**

The methods of the IRConcrFootingGeometry class are listed here.

**Public Methods**

	Name	Description
◆	GetDim (see page 1113)	Function returns the value of a relevant spread footing dimension. .
◆	SetDim (see page 1113)	Function sets the value of a relevant spread footing dimension. .

**I.1.2.3.1 GetDim****C++**

```
HRESULT GetDim( IRConcrFootingDimType _dimType, double* ret);
```

**C#**

```
public double GetDim(IRConcrFootingDimType _dimType);
```

**Visual Basic**

```
Public Function GetDim(_dimType As IRConcrFootingDimType) As double
```

**Description**

Function returns the value of a relevant spread footing dimension. .

**Version**

Available since version 3.

**I.1.2.3.2 SetDim****C++**

```
HRESULT SetDim( IRConcrFootingDimType _dimType, double _val);
```

**C#**

```
public void SetDim( IRConcrFootingDimType _dimType, double _val);
```

**Visual Basic**

```
Public Sub SetDim(_dimType As IRConcrFootingDimType, _val As double)
```

**Description**

Function sets the value of a relevant spread footing dimension. .

**Version**

Available since version 3.

## I.1.3 IRConcrFootingDimType

### C++

```
enum IRConcrFootingDimType;
```

### C#

```
public enum IRConcrFootingDimType;
```

### Visual Basic

```
Public Enum IRConcrFootingDimType
```

### Members

Members	Description
I_CFDT_FOOT_DIM_A = 1	Available since version 3.
I_CFDT_FOOT_DIM_B = 2	Available since version 3.
I_CFDT_FOOT_DIM_H1 = 3	Available since version 3.
I_CFDT_FOOT_DIM_H2 = 4	Available since version 3.
I_CFDT_FOOT_DIM_H3 = 5	Available since version 3.
I_CFDT_FOOT_DIM_H4 = 6	Available since version 3.
I_CFDT_FOOT_DIM_EX = 7	Available since version 3.
I_CFDT_FOOT_DIM_EY = 8	Available since version 3.
I_CFDT_FOOT_DIM_AP = 9	Available since version 3.
I_CFDT_FOOT_DIM_BP = 10	Available since version 3.
I_CFDT_FOOT_DIM_L = 11	Available since version 3.
I_CFDT_FOOT_DIM_COL_A = 12	Available since version 3.
I_CFDT_FOOT_DIM_COL_B = 13	Available since version 3.

### Description

A set of values identifying individual spread footing dimensions. .

### Version

Available since version 3.

## I.1.4 IRConcrFootingShapeType

### C++

```
enum IRConcrFootingShapeType;
```

### C#

```
public enum IRConcrFootingShapeType;
```

### Visual Basic

```
Public Enum IRConcrFootingShapeType
```

### Members

Members	Description
I_CFST_FOOT_SHAPE_RECT = 1	Spread footing of a rectangular section. Available since version 3.
I_CFST_FOOT_SHAPE_TRAPEZOIDAL = 2	Spread footing of a trapezoidal section. Available since version 3.
I_CFST_FOOT_SHAPE_TWO_CL = 3	Spread footing under two columns. Available since version 11.

**Description**

A set of values defining the spread footing shape. .

**Version**

Available since version 3.

**I.1.5 IRConcrFootingType****C++**

```
enum IRConcrFootingType;
```

**C#**

```
public enum IRConcrFootingType;
```

**Visual Basic**

```
Public Enum IRConcrFootingType
```

**Members**

Members	Description
I_CFT_FOOT_TYPE_FOOTING = 1	Column footing. Available since version 3.
I_CFT_FOOT_TYPE_WALL = 2	Continuous footing. Available since version 3.

**Description**

A set of values defining the foundation type. .

**Version**

Available since version 3.

**I.1.6 IRConcrFootingGround****Class Hierarchy****C++**

```
interface IRConcrFootingGround : IDispatch;
```

**C#**

```
public interface IRConcrFootingGround;
```

**Visual Basic**

```
Public Interface IRConcrFootingGround
```

**Description**

Interface representing soils and reference level of the footing.

**Version**

Available since version 3.

**I.1.6.1 IRConcrFootingGround Members**

The following tables list the members exposed by IRConcrFootingGround.

## Public Fields

	Name	Description
◆	ColumnPierLevel ( <a href="#">see page 1116</a> )	Column pier level.
◆	SoilLevel ( <a href="#">see page 1116</a> )	Reference level of the footing .

### I.1.6.2 IRConcrFootingGround Fields

The fields of the IRConcrFootingGround class are listed here.

## Public Fields

	Name	Description
◆	ColumnPierLevel ( <a href="#">see page 1116</a> )	Column pier level.
◆	SoilLevel ( <a href="#">see page 1116</a> )	Reference level of the footing .

#### I.1.6.2.1 ColumnPierLevel

##### C++

```
HRESULT get_ColumnPierLevel( double* );
HRESULT put_ColumnPierLevel( double );
```

##### C#

```
public double ColumnPierLevel { get; set; }
```

##### Visual Basic

```
Public ColumnPierLevel As Double
```

##### Description

Column pier level.

##### Version

Available since version 3.

#### I.1.6.2.2 SoilLevel

##### C++

```
HRESULT get_SoilLevel( double* );
HRESULT put_SoilLevel( double );
```

##### C#

```
public double SoilLevel { get; set; }
```

##### Visual Basic

```
Public SoilLevel As Double
```

##### Description

Reference level of the footing .

##### Version

Available since version 3.

## I.1.7 IRConcrFootingLoads

### Class Hierarchy

##### C++

```
interface IRConcrFootingLoads : IDispatch;
```

**C#**

```
public interface IRConcrFootingLoads;
```

**Visual Basic**

```
Public Interface IRConcrFootingLoads
```

**Description**

Interface representing spread footing loads and backfill loads. .

**Version**

Available since version 3.

**I.1.8 IRConcrFootingResults****Class Hierarchy****C++**

```
interface IRConcrFootingResults : IDispatch;
```

**C#**

```
public interface IRConcrFootingResults;
```

**Visual Basic**

```
Public Interface IRConcrFootingResults
```

**Description**

Interface representing calculation results for the spread footing. .

**Version**

Available since version 3.

**I.1.8.1 IRConcrFootingResults Members**

The following tables list the members exposed by IRConcrFootingResults.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Ax (❑ see page 1117)	Required reinforcement area (X direction).
❖	Ay (❑ see page 1118)	Required reinforcement area (Y direction).
❖	Val (❑ see page 1118)	

**I.1.8.2 IRConcrFootingResults Fields**

The fields of the IRConcrFootingResults class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Ax (❑ see page 1117)	Required reinforcement area (X direction).
❖	Ay (❑ see page 1118)	Required reinforcement area (Y direction).
❖	Val (❑ see page 1118)	

**I.1.8.2.1 Ax****C++**

```
HRESULT get_Ax( double* );
HRESULT put_Ax( double );
```

**C#**

```
public double Ax { get; set; }
```

**Visual Basic**

```
Public Ax As Double
```

**Description**

Required reinforcement area (X direction).

**Version**

Available since version 3.

### I.1.8.2.2 Ay

**C++**

```
HRESULT get_Ay( double* );
HRESULT put_Ay( double );
```

**C#**

```
public double Ay { get; set; }
```

**Visual Basic**

```
Public Ay As Double
```

**Description**

Required reinforcement area (Y direction).

**Version**

Available since version 3.

### I.1.8.2.3 Val

**C++**

```
HRESULT get_Val(double* );
HRESULT put_Val(double );
```

**C#**

```
public double Val { get; set; }
```

**Visual Basic**

```
Public Val As Double
```

**Version**

Available since version 10.

## I.1.9 IRConcrFootingCalculationOptions

**Class Hierarchy****C++**

```
interface IRConcrFootingCalculationOptions : IDispatch;
```

**C#**

```
public interface IRConcrFootingCalculationOptions;
```

## Visual Basic

```
Public Interface IRConcrFootingCalculationOptions
```

### Version

Available since version 10.

#### I.1.9.1 IRConcrFootingCalculationOptions Members

The following tables list the members exposed by IRConcrFootingCalculationOptions.

### Public Fields

	Name	Description
◆	Cover (see page 1119)	
◆	NominalCover (see page 1119)	
◆	NominalPierCover (see page 1119)	
◆	PierCover (see page 1120)	

#### I.1.9.2 IRConcrFootingCalculationOptions Fields

The fields of the IRConcrFootingCalculationOptions class are listed here.

### Public Fields

	Name	Description
◆	Cover (see page 1119)	
◆	NominalCover (see page 1119)	
◆	NominalPierCover (see page 1119)	
◆	PierCover (see page 1120)	

#### I.1.9.2.1 Cover

##### C++

```
HRESULT get_Cover(double*);  
HRESULT put_Cover(double);
```

##### C#

```
public double Cover { get; set; }
```

## Visual Basic

```
Public Cover As Double
```

### Version

Available since version 10.

#### I.1.9.2.2 NominalCover

##### C++

```
HRESULT get_NominalCover(double*);
```

##### C#

```
public double NominalCover { get; }
```

## Visual Basic

```
Public ReadOnly NominalCover As Double
```

**Version**

Available since version 11.

**I.1.9.2.3 NominalPierCover****C++**

```
HRESULT get_NominalPierCover( double* );
```

**C#**

```
public double NominalPierCover { get; }
```

**Visual Basic**

```
Public ReadOnly NominalPierCover As double
```

**Version**

Available since version 11.

**I.1.9.2.4 PierCover****C++**

```
HRESULT get_PierCover(double* );
HRESULT put_PierCover(double);
```

**C#**

```
public double PierCover { get; set; }
```

**Visual Basic**

```
Public PierCover As double
```

**Version**

Available since version 11.

**I.1.10 IRConcrFootingPattern****Class Hierarchy****C++**

```
interface IRConcrFootingPattern : IDispatch;
```

**C#**

```
public interface IRConcrFootingPattern;
```

**Visual Basic**

```
Public Interface IRConcrFootingPattern
```

**Version**

Available since version 10.

**I.1.10.1 IRConcrFootingPattern Members**

The following tables list the members exposed by IRConcrFootingPattern.

**Public Fields**

	Name	Description
◆	Diameter1 (↗ see page 1121)	
◆	Diameter2 (↗ see page 1121)	

◆	Spacing1Max (see page 1121)	
◆	Spacing1Min (see page 1122)	
◆	Spacing2Max (see page 1122)	
◆	Spacing2Min (see page 1122)	

### I.1.10.2 IRConcrFootingPattern Fields

The fields of the IRConcrFootingPattern class are listed here.

#### Public Fields

	Name	Description
◆	Diameter1 (see page 1121)	
◆	Diameter2 (see page 1121)	
◆	Spacing1Max (see page 1121)	
◆	Spacing1Min (see page 1122)	
◆	Spacing2Max (see page 1122)	
◆	Spacing2Min (see page 1122)	

#### I.1.10.2.1 Diameter1

##### C++

```
HRESULT get_Diameter1(double* );
HRESULT put_Diameter1(double);
```

##### C#

```
public double Diameter1 { get; set; }
```

##### Visual Basic

```
Public Diameter1 As Double
```

##### Version

Available since version 10.

#### I.1.10.2.2 Diameter2

##### C++

```
HRESULT get_Diameter2(double* );
HRESULT put_Diameter2(double);
```

##### C#

```
public double Diameter2 { get; set; }
```

##### Visual Basic

```
Public Diameter2 As Double
```

##### Version

Available since version 10.

#### I.1.10.2.3 Spacing1Max

##### C++

```
HRESULT get_Spacing1Max(double* );
HRESULT put_Spacing1Max(double);
```

##### C#

```
public double Spacing1Max { get; set; }
```

**Visual Basic**

```
Public Spacing1Max As double
```

**Version**

Available since version 10.

**I.1.10.2.4 Spacing1Min****C++**

```
HRESULT get_Spacing1Min(double*);  
HRESULT put_Spacing1Min(double);
```

**C#**

```
public double Spacing1Min { get; set; }
```

**Visual Basic**

```
Public Spacing1Min As double
```

**Version**

Available since version 10.

**I.1.10.2.5 Spacing2Max****C++**

```
HRESULT get_Spacing2Max(double*);  
HRESULT put_Spacing2Max(double);
```

**C#**

```
public double Spacing2Max { get; set; }
```

**Visual Basic**

```
Public Spacing2Max As double
```

**Version**

Available since version 10.

**I.1.10.2.6 Spacing2Min****C++**

```
HRESULT get_Spacing2Min(double*);  
HRESULT put_Spacing2Min(double);
```

**C#**

```
public double Spacing2Min { get; set; }
```

**Visual Basic**

```
Public Spacing2Min As double
```

**Version**

Available since version 10.

**I.1.11 IRConcrFootResultType****C++**

```
enum IRConcrFootResultType;
```

**C#**

```
public enum IRConcrFootResultType;
```

**Visual Basic**

```
Public Enum IRConcrFootResultType
```

**Members**

Members	Description
I_FRT_FOOT_RES_AX = 1	Available since version 10.
I_FRT_FOOT_RES_AY = 2	Available since version 10.

**Version**

Available since version 10.

**I.1.12 IRConcrFootingPierType****C++**

```
enum IRConcrFootingPierType;
```

**C#**

```
public enum IRConcrFootingPierType;
```

**Visual Basic**

```
Public Enum IRConcrFootingPierType
```

**Members**

Members	Description
I_CFT_FOOT_PIER_TYPE_PLAIN = 1	Available since version 11.
I_CFT_FOOT_PIER_TYPE_DOWELED = 2	Available since version 11.
I_CFT_FOOT_PIER_TYPE_BOLTED = 3	Available since version 11.
I_CFT_FOOT_PIER_TYPE_HINGE_1 = 4	Available since version 11.
I_CFT_FOOT_PIER_TYPE_HINGE_2 = 5	Available since version 11.
I_CFT_FOOT_PIER_TYPE_SOCKETED = 6	Available since version 11.

**Version**

Available since version 11.

**I.2 RC Beam**

Available since version 10.

**Enumerations**

	Name	Description
	IRConcrBeamLinearLoadType (see page 1159)	
	IRConcrBeamLoadNatureType (see page 1159)	
	IRConcrBeamPointLoadType (see page 1160)	
	IRConcrBeamSectionType (see page 1160)	
	IRConcrBeamSectionDimType (see page 1161)	

	IRConcrBeamCrackingType (see page 1162)	
	IRConcrBeamSupportType (see page 1162)	
	IRConcrBeamSupportMaterialType (see page 1162)	

## Interfaces

	Name	Description
	IRConcrBeam (see page 1124)	
	IRConcrBeamCalcOptions (see page 1133)	
	IRConcrBeamGeometry (see page 1136)	
	IRConcrBeamImportOptions (see page 1140)	
	IRConcrBeamLinearLoad (see page 1142)	
	IRConcrBeamPatternOptions (see page 1147)	
	IRConcrBeamPointLoad (see page 1148)	
	IRConcrBeamSegment (see page 1150)	
	IRConcrBeamSpan (see page 1151)	
	IRConcrBeamSpanNumbers (see page 1154)	
	IRConcrBeamStoryOptions (see page 1156)	
	IRConcrBeamSupport (see page 1157)	

## I.2.1 IRConcrBeam

### Class Hierarchy

#### C++

```
interface IRConcrBeam : IDispatch;
```

#### C#

```
public interface IRConcrBeam;
```

### Visual Basic

```
Public Interface IRConcrBeam
```

### Version

Available since version 10.

## I.2.1.1 IRConcrBeam Members

The following tables list the members exposed by IRConcrBeam.

## Public Fields

	Name	Description
◆	CalculationOptions ( <a href="#">see page 1126</a> )	
◆	Concrete ( <a href="#">see page 1126</a> )	
◆	Geometry ( <a href="#">see page 1127</a> )	
◆	ImportOptions ( <a href="#">see page 1127</a> )	
◆	IsActive ( <a href="#">see page 1127</a> )	
◆	IsSelected ( <a href="#">see page 1127</a> )	
◆	LinearLoad ( <a href="#">see page 1128</a> )	
◆	LinearLoadsCount ( <a href="#">see page 1128</a> )	
◆	Name ( <a href="#">see page 1128</a> )	
◆	NumberOfElements ( <a href="#">see page 1128</a> )	
◆	PatternOptions ( <a href="#">see page 1129</a> )	
◆	PointLoad ( <a href="#">see page 1129</a> )	
◆	PointLoadsCount ( <a href="#">see page 1129</a> )	
◆	Reinforcement ( <a href="#">see page 1129</a> )	
◆	Steel ( <a href="#">see page 1130</a> )	
◆	StoryOptions ( <a href="#">see page 1130</a> )	
◆	StructureUserNo ( <a href="#">see page 1130</a> )	
◆	StructureUserNoCount ( <a href="#">see page 1130</a> )	
◆	UniqueId ( <a href="#">see page 1131</a> )	

## Public Methods

	Name	Description
≡▼	Activate ( <a href="#">see page 1131</a> )	
≡▼	Calculate ( <a href="#">see page 1132</a> )	
≡▼	CreateCalculationNoteRtf ( <a href="#">see page 1132</a> )	
≡▼	CreateFromBars ( <a href="#">see page 1132</a> )	
≡▼	GenerateSpliceBars ( <a href="#">see page 1132</a> )	
≡▼	Save ( <a href="#">see page 1133</a> )	
≡▼	Verify ( <a href="#">see page 1133</a> )	
≡▼	VerifySpan ( <a href="#">see page 1133</a> )	

### I.2.1.2 IRConcrBeam Fields

The fields of the IRConcrBeam class are listed here.

## Public Fields

	Name	Description
◆	CalculationOptions ( <a href="#">see page 1126</a> )	
◆	Concrete ( <a href="#">see page 1126</a> )	
◆	Geometry ( <a href="#">see page 1127</a> )	
◆	ImportOptions ( <a href="#">see page 1127</a> )	
◆	IsActive ( <a href="#">see page 1127</a> )	

❖	IsSelected ( [ see page 1127) )	
❖	LinearLoad ( [ see page 1128) )	
❖	LinearLoadsCount ( [ see page 1128) )	
❖	Name ( [ see page 1128) )	
❖	NumberOfElements ( [ see page 1128) )	
❖	PatternOptions ( [ see page 1129) )	
❖	PointLoad ( [ see page 1129) )	
❖	PointLoadsCount ( [ see page 1129) )	
❖	Reinforcement ( [ see page 1129) )	
❖	Steel ( [ see page 1130) )	
❖	StoryOptions ( [ see page 1130) )	
❖	StructureUserNo ( [ see page 1130) )	
❖	StructureUserNoCount ( [ see page 1130) )	
❖	UniqueId ( [ see page 1131) )	

### I.2.1.2.1 CalculationOptions

#### C++

```
HRESULT get_CalculationOptions( IRConcrBeamCalcOptions** );
```

#### C#

```
public IRConcrBeamCalcOptions CalculationOptions { get; }
```

#### Visual Basic

```
Public ReadOnly CalculationOptions As IRConcrBeamCalcOptions
```

#### Version

Available since version 10.

### I.2.1.2.2 Concrete

#### C++

```
HRESULT get_Concrete( IRConcrConcrete** );
```

#### C#

```
public IRConcrConcrete Concrete { get; }
```

#### Visual Basic

```
Public ReadOnly Concrete As IRConcrConcrete
```

#### Version

Available since version 10.

### I.2.1.2.3 Geometry

#### C++

```
HRESULT get_Geometry( IRConcrBeamGeometry** );
```

#### C#

```
public IRConcrBeamGeometry Geometry { get; }
```

**Visual Basic**

```
Public ReadOnly Geometry As IRConcrBeamGeometry
```

**Version**

Available since version 10.

**I.2.1.2.4 ImportOptions****C++**

```
HRESULT get_ImportOptions(IRConcrBeamImportOptions**);
```

**C#**

```
public IRConcrBeamImportOptions ImportOptions { get; }
```

**Visual Basic**

```
Public ReadOnly ImportOptions As IRConcrBeamImportOptions
```

**Version**

Available since version 10.

**I.2.1.2.5 IsActive****C++**

```
HRESULT get_IsActive(VARIANT_BOOL*);
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Version**

Available since version 10.

**I.2.1.2.6 IsSelected****C++**

```
HRESULT get_IsSelected(VARIANT_BOOL*);  
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

**I.2.1.2.7 LinearLoad****C++**

```
HRESULT get_LinearLoad(IRConcrBeamLinearLoad**);
```

**C#**

```
public IRConcrBeamLinearLoad LinearLoad { get; }
```

**Visual Basic**

```
Public ReadOnly LinearLoad As IRConcrBeamLinearLoad
```

**Version**

Available since version 10.

**I.2.1.2.8 LinearLoadsCount****C++**

```
HRESULT get_LinearLoadsCount(short*);  
HRESULT put_LinearLoadsCount(short);
```

**C#**

```
public short LinearLoadsCount { get; set; }
```

**Visual Basic**

```
Public LinearLoadsCount As short
```

**Version**

Available since version 10.

**I.2.1.2.9 Name****C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

**I.2.1.2.10 NumberOfElements****C++**

```
HRESULT get_NumberOfElements(long*);  
HRESULT put_NumberOfElements(long);
```

**C#**

```
public long NumberOfElements { get; set; }
```

**Visual Basic**

```
Public NumberOfElements As long
```

**Version**

Available since version 12.

**I.2.1.2.11 PatternOptions****C++**

```
HRESULT get_PatternOptions(IRConcrBeamPatternOptions**);
```

**C#**

```
public IRConcrBeamPatternOptions PatternOptions { get; }
```

**Visual Basic**

```
Public ReadOnly PatternOptions As IRConcrBeamPatternOptions
```

**Version**

Available since version 10.

**I.2.1.2.12 PointLoad****C++**

```
HRESULT get_PointLoad(IRConcrBeamPointLoad**);
```

**C#**

```
public IRConcrBeamPointLoad PointLoad { get; }
```

**Visual Basic**

```
Public ReadOnly PointLoad As IRConcrBeamPointLoad
```

**Version**

Available since version 10.

**I.2.1.2.13 PointLoadsCount****C++**

```
HRESULT get_PointLoadsCount(short*);  
HRESULT put_PointLoadsCount(short);
```

**C#**

```
public short PointLoadsCount { get; set; }
```

**Visual Basic**

```
Public PointLoadsCount As short
```

**Version**

Available since version 10.

**I.2.1.2.14 Reinforcement****C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement**);
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

**I.2.1.2.15 Steel****C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

**I.2.1.2.16 StoryOptions****C++**

```
HRESULT get_StoryOptions(IRConcrBeamStoryOptions**);
```

**C#**

```
public IRConcrBeamStoryOptions StoryOptions { get; }
```

**Visual Basic**

```
Public ReadOnly StoryOptions As IRConcrBeamStoryOptions
```

**Version**

Available since version 10.

**I.2.1.2.17 StructureUserNo****C++**

```
HRESULT get_StructureUserNo(long*);  
HRESULT put_StructureUserNo(long);
```

**C#**

```
public long StructureUserNo { get; set; }
```

**Visual Basic**

```
Public StructureUserNo As long
```

**Version**

Available since version 10.

**I.2.1.2.18 StructureUserNoCount****C++**

```
HRESULT get_StructureUserNoCount(short*);  
HRESULT put_StructureUserNoCount(short);
```

**C#**

```
public short StructureUserNoCount { get; set; }
```

**Visual Basic**

```
Public StructureUserNoCount As short
```

**Version**

Available since version 10.

**I.2.1.2.19 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Version**

Available since version 10.

**I.2.1.3 IRConcrBeam Methods**

The methods of the IRConcrBeam class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Activate (see page 1131)	
💡	Calculate (see page 1132)	
💡	CreateCalculationNoteRtf (see page 1132)	
💡	CreateFromBars (see page 1132)	
💡	GenerateSpliceBars (see page 1132)	
💡	Save (see page 1133)	
💡	Verify (see page 1133)	
💡	VerifySpan (see page 1133)	

**I.2.1.3.1 Activate****C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

**I.2.1.3.2 Calculate****C++**

```
HRESULT Calculate(VARIANT_BOOL _createReinforcement, VARIANT_BOOL* ret);
```

**C#**

```
public bool Calculate(bool _createReinforcement);
```

**Visual Basic**

```
Public Function Calculate(_createReinforcement As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.2.1.3.3 CreateCalculationNoteRtf

**C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

### I.2.1.3.4 CreateFromBars

**C++**

```
HRESULT CreateFromBars(VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromBars();
```

**Visual Basic**

```
Public Function CreateFromBars() As Boolean
```

**Version**

Available since version 10.

### I.2.1.3.5 GenerateSpliceBars

**C++**

```
HRESULT GenerateSpliceBars(VARIANT_BOOL* ret);
```

**C#**

```
public bool GenerateSpliceBars();
```

**Visual Basic**

```
Public Function GenerateSpliceBars() As Boolean
```

**Version**

Available since version 10.

### I.2.1.3.6 Save

**C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.2.1.3.7 Verify****C++**

```
HRESULT Verify(VARIANT_BOOL _showErrors, VARIANT_BOOL* ret);
```

**C#**

```
public bool Verify(bool _showErrors);
```

**Visual Basic**

```
Public Function Verify(_showErrors As Boolean) As Boolean
```

**Version**

Available since version 10.

**I.2.1.3.8 VerifySpan****C++**

```
HRESULT VerifySpan(VARIANT_BOOL _showErrors, short _spanNo, long* ret);
```

**C#**

```
public long VerifySpan(bool _showErrors, short _spanNo);
```

**Visual Basic**

```
Public Function VerifySpan(_showErrors As Boolean, _spanNo As short) As long
```

**Version**

Available since version 10.

**I.2.2 IRConcrBeamCalcOptions****Class Hierarchy****C++**

```
interface IRConcrBeamCalcOptions : IDispatch;
```

**C#**

```
public interface IRConcrBeamCalcOptions;
```

**Visual Basic**

```
Public Interface IRConcrBeamCalcOptions
```

**Version**

Available since version 10.

**I.2.2.1 IRConcrBeamCalcOptions Members**

The following tables list the members exposed by IRConcrBeamCalcOptions.

## Public Fields

	Name	Description
◆	CalcSpanLengthInAxis (see page 1134)	
◆	CoverBottom (see page 1135)	
◆	CoverBottomFixed (see page 1135)	
◆	CoverSide (see page 1135)	
◆	CoverSideFixed (see page 1135)	
◆	CoverTop (see page 1136)	
◆	CoverTopFixed (see page 1136)	

### I.2.2.2 IRConcrBeamCalcOptions Fields

The fields of the IRConcrBeamCalcOptions class are listed here.

## Public Fields

	Name	Description
◆	CalcSpanLengthInAxis (see page 1134)	
◆	CoverBottom (see page 1135)	
◆	CoverBottomFixed (see page 1135)	
◆	CoverSide (see page 1135)	
◆	CoverSideFixed (see page 1135)	
◆	CoverTop (see page 1136)	
◆	CoverTopFixed (see page 1136)	

#### I.2.2.2.1 CalcSpanLengthInAxis

##### C++

```
HRESULT get_CalcSpanLengthInAxis(VARIANT_BOOL* );
HRESULT put_CalcSpanLengthInAxis(VARIANT_BOOL);
```

##### C#

```
public bool CalcSpanLengthInAxis { get; set; }
```

##### Visual Basic

```
Public CalcSpanLengthInAxis As Boolean
```

##### Version

Available since version 10.

#### I.2.2.2.2 CoverBottom

##### C++

```
HRESULT get_CoverBottom(double* );
HRESULT put_CoverBottom(double);
```

##### C#

```
public double CoverBottom { get; set; }
```

##### Visual Basic

```
Public CoverBottom As Double
```

**Version**

Available since version 10.

**I.2.2.2.3 CoverBottomFixed****C++**

```
HRESULT get_CoverBottomFixed(VARIANT_BOOL* );
HRESULT put_CoverBottomFixed(VARIANT_BOOL);
```

**C#**

```
public bool CoverBottomFixed { get; set; }
```

**Visual Basic**

```
Public CoverBottomFixed As Boolean
```

**Version**

Available since version 10.

**I.2.2.2.4 CoverSide****C++**

```
HRESULT get_CoverSide(double* );
HRESULT put_CoverSide(double);
```

**C#**

```
public double CoverSide { get; set; }
```

**Visual Basic**

```
Public CoverSide As Double
```

**Version**

Available since version 10.

**I.2.2.2.5 CoverSideFixed****C++**

```
HRESULT get_CoverSideFixed(VARIANT_BOOL* );
HRESULT put_CoverSideFixed(VARIANT_BOOL);
```

**C#**

```
public bool CoverSideFixed { get; set; }
```

**Visual Basic**

```
Public CoverSideFixed As Boolean
```

**Version**

Available since version 10.

**I.2.2.2.6 CoverTop****C++**

```
HRESULT get_CoverTop(double* );
HRESULT put_CoverTop(double);
```

**C#**

```
public double CoverTop { get; set; }
```

**Visual Basic**

```
Public CoverTop As double
```

**Version**

Available since version 10.

**I.2.2.2.7 CoverTopFixed****C++**

```
HRESULT get_CoverTopFixed(VARIANT_BOOL* );
HRESULT put_CoverTopFixed(VARIANT_BOOL);
```

**C#**

```
public bool CoverTopFixed { get; set; }
```

**Visual Basic**

```
Public CoverTopFixed As Boolean
```

**Version**

Available since version 10.

**I.2.3 IRConcrBeamGeometry****Class Hierarchy****C++**

```
interface IRConcrBeamGeometry : IDispatch;
```

**C#**

```
public interface IRConcrBeamGeometry;
```

**Visual Basic**

```
Public Interface IRConcrBeamGeometry
```

**Version**

Available since version 10.

**I.2.3.1 IRConcrBeamGeometry Members**

The following tables list the members exposed by IRConcrBeamGeometry.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AutoNameSpans (see page 1137)	
❖	AutoNameSupports (see page 1138)	
❖	ConcreteVolume (see page 1138)	
❖	LeftCantilever (see page 1138)	
❖	RightCantilever (see page 1138)	
❖	ShutteringArea (see page 1139)	
❖	Span (see page 1139)	
❖	SpanNumber (see page 1139)	

## Public Methods

	Name	Description
!	UpdateInternalGeometryData ( <a href="#">see page 1140</a> )	

### I.2.3.2 IRConcrBeamGeometry Fields

The fields of the IRConcrBeamGeometry class are listed here.

#### Public Fields

	Name	Description
!	AutoNameSpans ( <a href="#">see page 1137</a> )	
!	AutoNameSupports ( <a href="#">see page 1138</a> )	
!	ConcreteVolume ( <a href="#">see page 1138</a> )	
!	LeftCantilever ( <a href="#">see page 1138</a> )	
!	RightCantilever ( <a href="#">see page 1138</a> )	
!	ShutteringArea ( <a href="#">see page 1139</a> )	
!	Span ( <a href="#">see page 1139</a> )	
!	SpanNumber ( <a href="#">see page 1139</a> )	

#### I.2.3.2.1 AutoNameSpans

##### C++

```
HRESULT get_AutoNameSpans(VARIANT_BOOL* );
HRESULT put_AutoNameSpans(VARIANT_BOOL);
```

##### C#

```
public bool AutoNameSpans { get; set; }
```

##### Visual Basic

```
Public AutoNameSpans As Boolean
```

##### Version

Available since version 10.

#### I.2.3.2.2 AutoNameSupports

##### C++

```
HRESULT get_AutoNameSupports(VARIANT_BOOL* );
HRESULT put_AutoNameSupports(VARIANT_BOOL);
```

##### C#

```
public bool AutoNameSupports { get; set; }
```

##### Visual Basic

```
Public AutoNameSupports As Boolean
```

##### Version

Available since version 10.

#### I.2.3.2.3 ConcreteVolume

##### C++

```
HRESULT get_ConcreteVolume(double* );
```

**C#**

```
public double ConcreteVolume { get; }
```

**Visual Basic**

```
Public ReadOnly ConcreteVolume As Double
```

**Version**

Available since version 10.

**I.2.3.2.4 LeftCantilever****C++**

```
HRESULT get_LeftCantilever(VARIANT_BOOL* );
HRESULT put_LeftCantilever(VARIANT_BOOL);
```

**C#**

```
public bool LeftCantilever { get; set; }
```

**Visual Basic**

```
Public LeftCantilever As Boolean
```

**Version**

Available since version 10.

**I.2.3.2.5 RightCantilever****C++**

```
HRESULT get_RightCantilever(VARIANT_BOOL* );
HRESULT put_RightCantilever(VARIANT_BOOL);
```

**C#**

```
public bool RightCantilever { get; set; }
```

**Visual Basic**

```
Public RightCantilever As Boolean
```

**Version**

Available since version 10.

**I.2.3.2.6 ShutteringArea****C++**

```
HRESULT get_ShutteringArea(double* );
```

**C#**

```
public double ShutteringArea { get; }
```

**Visual Basic**

```
Public ReadOnly ShutteringArea As Double
```

**Version**

Available since version 10.

**I.2.3.2.7 Span****C++**

```
HRESULT get_Span(IRConcrBeamSpan** );
```

**C#**

```
public IRConcrBeamSpan Span { get; }
```

**Visual Basic**

```
Public ReadOnly Span As IRConcrBeamSpan
```

**Version**

Available since version 10.

**I.2.3.2.8 SpanNumber****C++**

```
HRESULT get_SpanNumber(short* );
HRESULT put_SpanNumber(short);
```

**C#**

```
public short SpanNumber { get; set; }
```

**Visual Basic**

```
Public SpanNumber As short
```

**Version**

Available since version 10.

**I.2.3.3 IRConcrBeamGeometry Methods**

The methods of the IRConcrBeamGeometry class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	UpdateInternalGeometryData ( <a href="#">[?]</a> see page 1140)	

**I.2.3.3.1 UpdateInternalGeometryData****C++**

```
HRESULT UpdateInternalGeometryData();
```

**C#**

```
public void UpdateInternalGeometryData();
```

**Visual Basic**

```
Public Sub UpdateInternalGeometryData()
```

**Version**

Available since version 10.

**I.2.4 IRConcrBeamImportOptions****Class Hierarchy****C++**

```
interface IRConcrBeamImportOptions : IDispatch;
```

**C#**

```
public interface IRConcrBeamImportOptions;
```

## Visual Basic

```
Public Interface IRConcrBeamImportOptions
```

### Version

Available since version 10.

#### I.2.4.1 IRConcrBeamImportOptions Members

The following tables list the members exposed by IRConcrBeamImportOptions.

### Public Fields

	Name	Description
◆	GroupByGeometry ( [ see page 1141 ] )	
◆	GroupByLevel ( [ see page 1141 ] )	
◆	ImportCombinations ( [ see page 1141 ] )	
◆	LiveLongCoeff ( [ see page 1142 ] )	
◆	RunCalcAuto ( [ see page 1142 ] )	
◆	ShowDialog ( [ see page 1142 ] )	

#### I.2.4.2 IRConcrBeamImportOptions Fields

The fields of the IRConcrBeamImportOptions class are listed here.

### Public Fields

	Name	Description
◆	GroupByGeometry ( [ see page 1141 ] )	
◆	GroupByLevel ( [ see page 1141 ] )	
◆	ImportCombinations ( [ see page 1141 ] )	
◆	LiveLongCoeff ( [ see page 1142 ] )	
◆	RunCalcAuto ( [ see page 1142 ] )	
◆	ShowDialog ( [ see page 1142 ] )	

#### I.2.4.2.1 GroupByGeometry

##### C++

```
HRESULT get_GroupByGeometry(VARIANT_BOOL* );
HRESULT put_GroupByGeometry(VARIANT_BOOL);
```

##### C#

```
public bool GroupByGeometry { get; set; }
```

## Visual Basic

```
Public GroupByGeometry As Boolean
```

### Version

Available since version 10.

#### I.2.4.2.2 GroupByLevel

##### C++

```
HRESULT get_GroupByLevel(VARIANT_BOOL* );
HRESULT put_GroupByLevel(VARIANT_BOOL);
```

**C#**

```
public bool GroupByLevel { get; set; }
```

**Visual Basic**

```
Public GroupByLevel As Boolean
```

**Version**

Available since version 10.

**I.2.4.2.3 ImportCombinations****C++**

```
HRESULT get_ImportCombinations(VARIANT_BOOL* );
HRESULT put_ImportCombinations(VARIANT_BOOL);
```

**C#**

```
public bool ImportCombinations { get; set; }
```

**Visual Basic**

```
Public ImportCombinations As Boolean
```

**Version**

Available since version 10.

**I.2.4.2.4 LiveLongCoeff****C++**

```
HRESULT get_LiveLongCoeff(double* );
HRESULT put_LiveLongCoeff(double);
```

**C#**

```
public double LiveLongCoeff { get; set; }
```

**Visual Basic**

```
Public LiveLongCoeff As Double
```

**Version**

Available since version 10.

**I.2.4.2.5 RunCalcAuto****C++**

```
HRESULT get_RunCalcAuto(VARIANT_BOOL* );
HRESULT put_RunCalcAuto(VARIANT_BOOL);
```

**C#**

```
public bool RunCalcAuto { get; set; }
```

**Visual Basic**

```
Public RunCalcAuto As Boolean
```

**Version**

Available since version 10.

### I.2.4.2.6 ShowDialog

#### C++

```
HRESULT get_ShowDialog(VARIANT_BOOL* );
HRESULT put_ShowDialog(VARIANT_BOOL);
```

#### C#

```
public bool ShowDialog { get; set; }
```

#### Visual Basic

```
Public ShowDialog As Boolean
```

#### Version

Available since version 10.

## I.2.5 IRConcrBeamLinearLoad

#### Class Hierarchy

#### C++

```
interface IRConcrBeamLinearLoad : IDispatch;
```

#### C#

```
public interface IRConcrBeamLinearLoad;
```

#### Visual Basic

```
Public Interface IRConcrBeamLinearLoad
```

#### Version

Available since version 10.

### I.2.5.1 IRConcrBeamLinearLoad Members

The following tables list the members exposed by IRConcrBeamLinearLoad.

#### Public Fields

	Name	Description
◆	Case (see page 1143)	
◆	Nature (see page 1144)	
◆	RelativeCoordinates (see page 1144)	
◆	Spans (see page 1144)	
◆	Type (see page 1145)	
◆	Value1 (see page 1145)	
◆	Value2 (see page 1145)	
◆	Value3 (see page 1145)	
◆	X1 (see page 1146)	
◆	X2 (see page 1146)	
◆	X3 (see page 1146)	
◆	X4 (see page 1146)	

### I.2.5.2 IRConcrBeamLinearLoad Fields

The fields of the IRConcrBeamLinearLoad class are listed here.

## Public Fields

	Name	Description
◆	Case ( [ see page 1143)	
◆	Nature ( [ see page 1144)	
◆	RelativeCoordinates ( [ see page 1144)	
◆	Spans ( [ see page 1144)	
◆	Type ( [ see page 1145)	
◆	Value1 ( [ see page 1145)	
◆	Value2 ( [ see page 1145)	
◆	Value3 ( [ see page 1145)	
◆	X1 ( [ see page 1146)	
◆	X2 ( [ see page 1146)	
◆	X3 ( [ see page 1146)	
◆	X4 ( [ see page 1146)	

### I.2.5.2.1 Case

#### C++

```
HRESULT get_Case(short* );
HRESULT put_Case(short);
```

#### C#

```
public short Case { get; set; }
```

#### Visual Basic

```
Public Case As short
```

#### Version

Available since version 10.

### I.2.5.2.2 Nature

#### C++

```
HRESULT get_Nature(IRConcrBeamLoadNatureType* );
HRESULT put_Nature(IRConcrBeamLoadNatureType);
```

#### C#

```
public IRConcrBeamLoadNatureType Nature { get; set; }
```

#### Visual Basic

```
Public Nature As IRConcrBeamLoadNatureType
```

#### Version

Available since version 10.

### I.2.5.2.3 RelativeCoordinates

#### C++

```
HRESULT get_RelativeCoordinates(VARIANT_BOOL* );
HRESULT put_RelativeCoordinates(VARIANT_BOOL);
```

#### C#

```
public bool RelativeCoordinates { get; set; }
```

**Visual Basic**

```
Public RelativeCoordinates As Boolean
```

**Version**

Available since version 10.

**I.2.5.2.4 Spans****C++**

```
HRESULT get_Spans(IRConcrBeamSpanNumbers**);
```

**C#**

```
public IRConcrBeamSpanNumbers Spans { get; }
```

**Visual Basic**

```
Public ReadOnly Spans As IRConcrBeamSpanNumbers
```

**Version**

Available since version 10.

**I.2.5.2.5 Type****C++**

```
HRESULT get_Type(IRConcrBeamLinearLoadType*);  
HRESULT put_Type(IRConcrBeamLinearLoadType);
```

**C#**

```
public IRConcrBeamLinearLoadType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRConcrBeamLinearLoadType
```

**Version**

Available since version 10.

**I.2.5.2.6 Value1****C++**

```
HRESULT get_Value1(double*);  
HRESULT put_Value1(double);
```

**C#**

```
public double Value1 { get; set; }
```

**Visual Basic**

```
Public Value1 As Double
```

**Version**

Available since version 10.

**I.2.5.2.7 Value2****C++**

```
HRESULT get_Value2(double*);  
HRESULT put_Value2(double);
```

**C#**

```
public double Value2 { get; set; }
```

**Visual Basic**

```
Public Value2 As Double
```

**Version**

Available since version 10.

**I.2.5.2.8 Value3****C++**

```
HRESULT get_Value3(double*);  
HRESULT put_Value3(double);
```

**C#**

```
public double Value3 { get; set; }
```

**Visual Basic**

```
Public Value3 As Double
```

**Version**

Available since version 10.

**I.2.5.2.9 X1****C++**

```
HRESULT get_X1(double*);  
HRESULT put_X1(double);
```

**C#**

```
public double X1 { get; set; }
```

**Visual Basic**

```
Public X1 As Double
```

**Version**

Available since version 10.

**I.2.5.2.10 X2****C++**

```
HRESULT get_X2(double*);  
HRESULT put_X2(double);
```

**C#**

```
public double X2 { get; set; }
```

**Visual Basic**

```
Public X2 As Double
```

**Version**

Available since version 10.

**I.2.5.2.11 X3****C++**

```
HRESULT get_X3(double*);
```

```
HRESULT put_X3(double);
```

**C#**

```
public double X3 { get; set; }
```

**Visual Basic**

```
Public X3 As Double
```

**Version**

Available since version 10.

**I.2.5.2.12 X4****C++**

```
HRESULT get_X4(double*);  
HRESULT put_X4(double);
```

**C#**

```
public double X4 { get; set; }
```

**Visual Basic**

```
Public X4 As Double
```

**Version**

Available since version 10.

**I.2.6 IRConcrBeamPatternOptions****Class Hierarchy****C++**

```
interface IRConcrBeamPatternOptions : IDispatch;
```

**C#**

```
public interface IRConcrBeamPatternOptions;
```

**Visual Basic**

```
Public Interface IRConcrBeamPatternOptions
```

**Version**

Available since version 10.

**I.2.6.1 IRConcrBeamPatternOptions Members**

The following tables list the members exposed by IRConcrBeamPatternOptions.

**Public Fields**

	Name	Description
◆	SpanBySpan (see page 1147)	

**I.2.6.2 IRConcrBeamPatternOptions Fields**

The fields of the IRConcrBeamPatternOptions class are listed here.

**Public Fields**

	Name	Description
◆	SpanBySpan (see page 1147)	

### I.2.6.2.1 SpanBySpan

#### C++

```
HRESULT get_SpanBySpan(VARIANT_BOOL* );
HRESULT put_SpanBySpan(VARIANT_BOOL);
```

#### C#

```
public bool SpanBySpan { get; set; }
```

#### Visual Basic

```
Public SpanBySpan As Boolean
```

#### Version

Available since version 10.

## I.2.7 IRConcrBeamPointLoad

#### Class Hierarchy

#### C++

```
interface IRConcrBeamPointLoad : IDispatch;
```

#### C#

```
public interface IRConcrBeamPointLoad;
```

#### Visual Basic

```
Public Interface IRConcrBeamPointLoad
```

#### Version

Available since version 10.

### I.2.7.1 IRConcrBeamPointLoad Members

The following tables list the members exposed by IRConcrBeamPointLoad.

#### Public Fields

	Name	Description
❖	Case (see page 1148)	
❖	Nature (see page 1149)	
❖	RelativeCoordinates (see page 1149)	
❖	Spans (see page 1149)	
❖	Type (see page 1149)	
❖	Value (see page 1150)	
❖	X1 (see page 1150)	

### I.2.7.2 IRConcrBeamPointLoad Fields

The fields of the IRConcrBeamPointLoad class are listed here.

#### Public Fields

	Name	Description
❖	Case (see page 1148)	
❖	Nature (see page 1149)	

❖	RelativeCoordinates ( <a href="#">see page 1149</a> )	
❖	Spans ( <a href="#">see page 1149</a> )	
❖	Type ( <a href="#">see page 1149</a> )	
❖	Value ( <a href="#">see page 1150</a> )	
❖	X1 ( <a href="#">see page 1150</a> )	

### I.2.7.2.1 Case

#### C++

```
HRESULT get_Case(short* );
HRESULT put_Case(short);
```

#### C#

```
public short Case { get; set; }
```

#### Visual Basic

```
Public Case As short
```

#### Version

Available since version 10.

### I.2.7.2.2 Nature

#### C++

```
HRESULT get_Nature(IRConcrBeamLoadNatureType* );
HRESULT put_Nature(IRConcrBeamLoadNatureType);
```

#### C#

```
public IRConcrBeamLoadNatureType Nature { get; set; }
```

#### Visual Basic

```
Public Nature As IRConcrBeamLoadNatureType
```

#### Version

Available since version 10.

### I.2.7.2.3 RelativeCoordinates

#### C++

```
HRESULT get_RelativeCoordinates(VARIANT_BOOL* );
HRESULT put_RelativeCoordinates(VARIANT_BOOL);
```

#### C#

```
public bool RelativeCoordinates { get; set; }
```

#### Visual Basic

```
Public RelativeCoordinates As Boolean
```

#### Version

Available since version 10.

### I.2.7.2.4 Spans

#### C++

```
HRESULT get_Spans(IRConcrBeamSpanNumbers** );
```

**C#**

```
public IRConcrBeamSpanNumbers Spans { get; }
```

**Visual Basic**

```
Public ReadOnly Spans As IRConcrBeamSpanNumbers
```

**Version**

Available since version 10.

### I.2.7.2.5 Type

**C++**

```
HRESULT get_Type(IRConcrBeamPointLoadType* );
HRESULT put_Type(IRConcrBeamPointLoadType);
```

**C#**

```
public IRConcrBeamPointLoadType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRConcrBeamPointLoadType
```

**Version**

Available since version 10.

### I.2.7.2.6 Value

**C++**

```
HRESULT get_Value(double* );
HRESULT put_Value(double);
```

**C#**

```
public double Value { get; set; }
```

**Visual Basic**

```
Public Value As double
```

**Version**

Available since version 10.

### I.2.7.2.7 X1

**C++**

```
HRESULT get_X1(double* );
HRESULT put_X1(double);
```

**C#**

```
public double X1 { get; set; }
```

**Visual Basic**

```
Public X1 As double
```

**Version**

Available since version 10.

## I.2.8 IRConcrBeamSegment

### Class Hierarchy

**C++**

```
interface IRConcrBeamSegment : IDispatch;
```

**C#**

```
public interface IRConcrBeamSegment;
```

**Visual Basic**

```
Public Interface IRConcrBeamSegment
```

**Version**

Available since version 10.

**I.2.8.1 IRConcrBeamSegment Members**

The following tables list the members exposed by IRConcrBeamSegment.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Dim ( <a href="#">see page 1151</a> )	
◆	SectionType ( <a href="#">see page 1151</a> )	

**I.2.8.2 IRConcrBeamSegment Fields**

The fields of the IRConcrBeamSegment class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Dim ( <a href="#">see page 1151</a> )	
◆	SectionType ( <a href="#">see page 1151</a> )	

**I.2.8.2.1 Dim****C++**

```
HRESULT get_Dim(double* );
HRESULT put_Dim(double);
```

**C#**

```
public double Dim { get; set; }
```

**Visual Basic**

```
Public Dim As double
```

**Version**

Available since version 10.

**I.2.8.2.2 SectionType****C++**

```
HRESULT get_SectionType(IRConcrBeamSectionType* );
HRESULT put_SectionType(IRConcrBeamSectionType);
```

**C#**

```
public IRConcrBeamSectionType SectionType { get; set; }
```

**Visual Basic**

```
Public SectionType As IRConcrBeamSectionType
```

## Version

Available since version 10.

## I.2.9 IRConcrBeamSpan

### Class Hierarchy

#### C++

```
interface IRConcrBeamSpan : IDispatch;
```

#### C#

```
public interface IRConcrBeamSpan;
```

### Visual Basic

```
Public Interface IRConcrBeamSpan
```

## Version

Available since version 10.

### I.2.9.1 IRConcrBeamSpan Members

The following tables list the members exposed by IRConcrBeamSpan.

#### Public Fields

	Name	Description
◆	CalculationLength ( <a href="#">see page 1152</a> )	
◆	HasOpenings ( <a href="#">see page 1153</a> )	
◆	LeftSupport ( <a href="#">see page 1153</a> )	
◆	Length ( <a href="#">see page 1153</a> )	
◆	Name ( <a href="#">see page 1153</a> )	
◆	RightSupport ( <a href="#">see page 1154</a> )	
◆	Segment ( <a href="#">see page 1154</a> )	
◆	SegmentNumber ( <a href="#">see page 1154</a> )	

### I.2.9.2 IRConcrBeamSpan Fields

The fields of the IRConcrBeamSpan class are listed here.

#### Public Fields

	Name	Description
◆	CalculationLength ( <a href="#">see page 1152</a> )	
◆	HasOpenings ( <a href="#">see page 1153</a> )	
◆	LeftSupport ( <a href="#">see page 1153</a> )	
◆	Length ( <a href="#">see page 1153</a> )	
◆	Name ( <a href="#">see page 1153</a> )	
◆	RightSupport ( <a href="#">see page 1154</a> )	
◆	Segment ( <a href="#">see page 1154</a> )	
◆	SegmentNumber ( <a href="#">see page 1154</a> )	

### I.2.9.2.1 CalculationLength

**C++**

```
HRESULT get_CalculationLength(double*);
```

**C#**

```
public double CalculationLength { get; }
```

**Visual Basic**

```
Public ReadOnly CalculationLength As double
```

**Version**

Available since version 10.

### I.2.9.2.2 HasOpenings

**C++**

```
HRESULT get_HasOpenings(VARIANT_BOOL*);
```

**C#**

```
public bool HasOpenings { get; }
```

**Visual Basic**

```
Public ReadOnly HasOpenings As Boolean
```

**Version**

Available since version 10.

### I.2.9.2.3 LeftSupport

**C++**

```
HRESULT get_LeftSupport(IRConcrBeamSupport**);
```

**C#**

```
public IRConcrBeamSupport LeftSupport { get; }
```

**Visual Basic**

```
Public ReadOnly LeftSupport As IRConcrBeamSupport
```

**Version**

Available since version 10.

### I.2.9.2.4 Length

**C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Version**

Available since version 10.

### I.2.9.2.5 Name

#### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Version

Available since version 10.

### I.2.9.2.6 RightSupport

#### C++

```
HRESULT get_RightSupport(IRConcrBeamSupport** );
```

#### C#

```
public IRConcrBeamSupport RightSupport { get; }
```

#### Visual Basic

```
Public ReadOnly RightSupport As IRConcrBeamSupport
```

#### Version

Available since version 10.

### I.2.9.2.7 Segment

#### C++

```
HRESULT get_Segment(IRConcrBeamSegment** );
```

#### C#

```
public IRConcrBeamSegment Segment { get; }
```

#### Visual Basic

```
Public ReadOnly Segment As IRConcrBeamSegment
```

#### Version

Available since version 10.

### I.2.9.2.8 SegmentNumber

#### C++

```
HRESULT get_SegmentNumber(short* );
HRESULT put_SegmentNumber(short);
```

#### C#

```
public short SegmentNumber { get; set; }
```

#### Visual Basic

```
Public SegmentNumber As short
```

#### Version

Available since version 10.

## I.2.10 IRConcrBeamSpanNumbers

### Class Hierarchy

#### C++

```
interface IRConcrBeamSpanNumbers : IDispatch;
```

#### C#

```
public interface IRConcrBeamSpanNumbers;
```

### Visual Basic

```
Public Interface IRConcrBeamSpanNumbers
```

### Version

Available since version 10.

## I.2.10.1 IRConcrBeamSpanNumbers Members

The following tables list the members exposed by IRConcrBeamSpanNumbers.

### Public Fields

	Name	Description
◆	Count (see page 1155)	
◆	Item (see page 1155)	

### Public Methods

	Name	Description
+=	Add (see page 1156)	
-=	RemoveAll (see page 1156)	

## I.2.10.2 IRConcrBeamSpanNumbers Fields

The fields of the IRConcrBeamSpanNumbers class are listed here.

### Public Fields

	Name	Description
◆	Count (see page 1155)	
◆	Item (see page 1155)	

## I.2.10.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

### Visual Basic

```
Public ReadOnly Count As long
```

### Version

Available since version 10.

### I.2.10.2.2 Item

#### C++

```
HRESULT get_Item(short*);
```

#### C#

```
public short Item { get; }
```

#### Visual Basic

```
Public ReadOnly Item As short
```

#### Version

Available since version 10.

### I.2.10.3 IRConcrBeamSpanNumbers Methods

The methods of the IRConcrBeamSpanNumbers class are listed here.

#### Public Methods

	Name	Description
💡	Add (see page 1156)	
💡	RemoveAll (see page 1156)	

### I.2.10.3.1 Add

#### C++

```
HRESULT Add(short newVal);
```

#### C#

```
public void Add(short newVal);
```

#### Visual Basic

```
Public Sub Add(_newVal As short)
```

#### Version

Available since version 10.

### I.2.10.3.2 RemoveAll

#### C++

```
HRESULT RemoveAll();
```

#### C#

```
public void RemoveAll();
```

#### Visual Basic

```
Public Sub RemoveAll()
```

#### Version

Available since version 10.

### I.2.11 IRConcrBeamStoryOptions

#### Class Hierarchy

**C++**

```
interface IRConcrBeamStoryOptions : IDispatch;
```

**C#**

```
public interface IRConcrBeamStoryOptions;
```

**Visual Basic**

```
Public Interface IRConcrBeamStoryOptions
```

**Version**

Available since version 10.

**I.2.11.1 IRConcrBeamStoryOptions Members**

The following tables list the members exposed by IRConcrBeamStoryOptions.

**Public Fields**

	Name	Description
◆	Cracking (see page 1157)	

**I.2.11.2 IRConcrBeamStoryOptions Fields**

The fields of the IRConcrBeamStoryOptions class are listed here.

**Public Fields**

	Name	Description
◆	Cracking (see page 1157)	

**I.2.11.2.1 Cracking****C++**

```
HRESULT get_Cracking(IRConcrBeamCrackingType* );
HRESULT put_Cracking(IRConcrBeamCrackingType);
```

**C#**

```
public IRConcrBeamCrackingType Cracking { get; set; }
```

**Visual Basic**

```
Public Cracking As IRConcrBeamCrackingType
```

**Version**

Available since version 10.

**I.2.12 IRConcrBeamSupport****Class Hierarchy****C++**

```
interface IRConcrBeamSupport : IDispatch;
```

**C#**

```
public interface IRConcrBeamSupport;
```

**Visual Basic**

```
Public Interface IRConcrBeamSupport
```

## Version

Available since version 10.

### I.2.12.1 IRConcrBeamSupport Members

The following tables list the members exposed by IRConcrBeamSupport.

#### Public Fields

	Name	Description
◆	MaterialType ( <a href="#">see page 1158</a> )	
◆	Name ( <a href="#">see page 1158</a> )	
◆	Type ( <a href="#">see page 1158</a> )	
◆	Width ( <a href="#">see page 1159</a> )	

### I.2.12.2 IRConcrBeamSupport Fields

The fields of the IRConcrBeamSupport class are listed here.

#### Public Fields

	Name	Description
◆	MaterialType ( <a href="#">see page 1158</a> )	
◆	Name ( <a href="#">see page 1158</a> )	
◆	Type ( <a href="#">see page 1158</a> )	
◆	Width ( <a href="#">see page 1159</a> )	

### I.2.12.2.1 MaterialType

#### C++

```
HRESULT get_MaterialType(IRConcrBeamSupportMaterialType\*);
HRESULT put_MaterialType(IRConcrBeamSupportMaterialType);
```

#### C#

```
public IRConcrBeamSupportMaterialType MaterialType { get; set; }
```

#### Visual Basic

```
Public MaterialType As IRConcrBeamSupportMaterialType
```

#### Version

Available since version 10.

### I.2.12.2.2 Name

#### C++

```
HRESULT get_Name(BSTR\*);
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Version

Available since version 10.

### I.2.12.2.3 Type

#### C++

```
HRESULT get_Type(IRConcrBeamSupportType* );
HRESULT put_Type(IRConcrBeamSupportType);
```

#### C#

```
public IRConcrBeamSupportType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRConcrBeamSupportType
```

#### Version

Available since version 10.

### I.2.12.2.4 Width

#### C++

```
HRESULT get_Width(double* );
HRESULT put_Width(double);
```

#### C#

```
public double Width { get; set; }
```

#### Visual Basic

```
Public Width As double
```

#### Version

Available since version 10.

## I.2.13 IRConcrBeamLinearLoadType

#### C++

```
enum IRConcrBeamLinearLoadType;
```

#### C#

```
public enum IRConcrBeamLinearLoadType;
```

#### Visual Basic

```
Public Enum IRConcrBeamLinearLoadType
```

#### Members

Members	Description
I_LLT_BEAM_LOAD_SELF_WEIGHT = 1	Available since version 10.
I_LLT_BEAM_LOAD_UNIFORM = 2	Available since version 10.
I_LLT_BEAM_LOAD_CONTINUOUS = 3	Available since version 10.
I_LLT_BEAM_LOAD_TRAPEZOIDAL1 = 4	Available since version 10.
I_LLT_BEAM_LOAD_TRAPEZOIDAL2 = 5	Available since version 10.
I_LLT_BEAM_LOAD_TRAPEZOIDAL3 = 6	Available since version 10.

#### Version

Available since version 10.

## I.2.14 IRConcrBeamLoadNatureType

### C++

```
enum IRConcrBeamLoadNatureType;
```

### C#

```
public enum IRConcrBeamLoadNatureType;
```

### Visual Basic

```
Public Enum IRConcrBeamLoadNatureType
```

### Members

Members	Description
I_BLN_BEAM_NATURE_DEAD = 1	Available since version 10.
I_BLN_BEAM_NATURE_LIVE = 2	Available since version 10.
I_BLN_BEAM_NATURE_WIND = 3	Available since version 10.
I_BLN_BEAM_NATURE_SNOW = 4	Available since version 10.
I_BLN_BEAM_NATURE_SEISMIC = 5	Available since version 10.

### Version

Available since version 10.

## I.2.15 IRConcrBeamPointLoadType

### C++

```
enum IRConcrBeamPointLoadType;
```

### C#

```
public enum IRConcrBeamPointLoadType;
```

### Visual Basic

```
Public Enum IRConcrBeamPointLoadType
```

### Members

Members	Description
I_PLT_BEAM_LOAD_FORCE = 1	Available since version 10.
I_PLT_BEAM_LOAD_MOMENT = 2	Available since version 10.
I_PLT_BEAM_LOAD_AXIAL_FORCE = 3	Available since version 10.
I_PLT_BEAM_LOAD_TORSION_MOMENT = 4	Available since version 10.
I_PLT_BEAM_LOAD_UNIFORM_TORSION_MOMENT = 5	Available since version 10.

### Version

Available since version 10.

## I.2.16 IRConcrBeamSectionType

### C++

```
enum IRConcrBeamSectionType;
```

### C#

```
public enum IRConcrBeamSectionType;
```

## Visual Basic

```
Public Enum IRConcrBeamSectionType
```

### Members

Members	Description
I_CBS_BEAM_SECTION_NG_NP = 1	Available since version 10.
I_CBS_BEAM_SECTION_NG_LP = 2	Available since version 10.
I_CBS_BEAM_SECTION_NG_RP = 3	Available since version 10.
I_CBS_BEAM_SECTION_NG_BP = 4	Available since version 10.
I_CBS_BEAM_SECTION_LG_NP = 5	Available since version 10.
I_CBS_BEAM_SECTION_LG_LP = 6	Available since version 10.
I_CBS_BEAM_SECTION_LG_RP = 7	Available since version 10.
I_CBS_BEAM_SECTION_LG_BP = 8	Available since version 10.
I_CBS_BEAM_SECTION_RG_NP = 9	Available since version 10.
I_CBS_BEAM_SECTION_RG_LP = 10	Available since version 10.
I_CBS_BEAM_SECTION_RG_RP = 11	Available since version 10.
I_CBS_BEAM_SECTION_RG_BP = 12	Available since version 10.

### Version

Available since version 10.

## I.2.17 IRConcrBeamSectionDimType

### C++

```
enum IRConcrBeamSectionDimType;
```

### C#

```
public enum IRConcrBeamSectionDimType;
```

## Visual Basic

```
Public Enum IRConcrBeamSectionDimType
```

### Members

Members	Description
I_BSD_BEAM_SECTION_DIM_L = 1	Available since version 10.
I_BSD_BEAM_SECTION_DIM_H1 = 2	Available since version 10.
I_BSD_BEAM_SECTION_DIM_H2 = 3	Available since version 10.
I_BSD_BEAM_SECTION_DIM_OFFY1 = 4	Available since version 10.
I_BSD_BEAM_SECTION_DIM_OFFY2 = 5	Available since version 10.
I_BSD_BEAM_SECTION_DIM_OFFZ1 = 6	Available since version 10.
I_BSD_BEAM_SECTION_DIM_OFFZ2 = 7	Available since version 10.
I_BSD_BEAM_SECTION_DIM_B = 8	Available since version 10.
I_BSD_BEAM_SECTION_DIM_EL1 = 9	Available since version 10.
I_BSD_BEAM_SECTION_DIM_EL2 = 10	Available since version 10.
I_BSD_BEAM_SECTION_DIM_PL = 11	Available since version 10.
I_BSD_BEAM_SECTION_DIM_RL = 12	Available since version 10.
I_BSD_BEAM_SECTION_DIM_HG1 = 13	Available since version 10.
I_BSD_BEAM_SECTION_DIM_BG1 = 14	Available since version 10.
I_BSD_BEAM_SECTION_DIM_HG2 = 15	Available since version 10.
I_BSD_BEAM_SECTION_DIM_BG2 = 16	Available since version 10.
I_BSD_BEAM_SECTION_DIM_ER1 = 17	Available since version 10.

I_BSD_BEAM_SECTION_DIM_ER2 = 18	Available since version 10.
I_BSD_BEAM_SECTION_DIM_PR = 19	Available since version 10.
I_BSD_BEAM_SECTION_DIM_RR = 20	Available since version 10.
I_BSD_BEAM_SECTION_DIM_HG3 = 21	Available since version 10.
I_BSD_BEAM_SECTION_DIM_BG3 = 22	Available since version 10.
I_BSD_BEAM_SECTION_DIM_HG4 = 23	Available since version 10.
I_BSD_BEAM_SECTION_DIM_BG4 = 24	Available since version 10.
I_BSD_BEAM_SECTION_DIM_PL_CALC = 25	Available since version 10.
I_BSD_BEAM_SECTION_DIM_PR_CALC = 26	Available since version 10.
I_BSD_BEAM_SECTION_DIM_CL = 27	Available since version 10.
I_BSD_BEAM_SECTION_DIM_CR = 28	Available since version 10.

**Version**

Available since version 10.

**I.2.18 IRConcrBeamCrackingType****C++**

```
enum IRConcrBeamCrackingType;
```

**C#**

```
public enum IRConcrBeamCrackingType;
```

**Visual Basic**

```
Public Enum IRConcrBeamCrackingType
```

**Members**

Members	Description
I_CBC_BEAM_CRACKING_UNDEFINED = 1	Available since version 10.
I_CBC_BEAM_CRACKING_PERMISSIBLE = 2	Available since version 10.
I_CBC_BEAM_CRACKING_LIMITED = 3	Available since version 10.
I_CBC_BEAM_CRACKING_NOT_PERMISSIBLE = 4	Available since version 10.

**Version**

Available since version 10.

**I.2.19 IRConcrBeamSupportType****C++**

```
enum IRConcrBeamSupportType;
```

**C#**

```
public enum IRConcrBeamSupportType;
```

**Visual Basic**

```
Public Enum IRConcrBeamSupportType
```

**Members**

Members	Description
IBST_BEAM_SUPPORT_PINNED = 1	Available since version 10.
IBST_BEAM_SUPPORT_FIXED = 2	Available since version 10.
IBST_BEAM_SUPPORT_SWAY = 3	Available since version 10.

**Version**

Available since version 10.

**I.2.20 IRConcrBeamSupportMaterialType****C++**

```
enum IRConcrBeamSupportMaterialType;
```

**C#**

```
public enum IRConcrBeamSupportMaterialType;
```

**Visual Basic**

```
Public Enum IRConcrBeamSupportMaterialType
```

**Members**

Members	Description
I_BSM_BEAM_SUPPORT_CONCRETE = 1	Available since version 10.
I_BSM_BEAM_SUPPORT_MASONRY = 2	Available since version 10.

**Version**

Available since version 10.

**I.3 Continuous Footing**

Available since version 10.

**Enumerations**

	Name	Description
	IRConcrContinuousFootingSectionType (see page 1177)	
	IRConcrContinuousFootingSectionDimType (see page 1178)	
	IRConcrContinuousFootingSupportMaterialType (see page 1178)	

**Interfaces**

	Name	Description
	IRConcrContinuousFooting (see page 1163)	
	IRConcrContinuousFootingGeometry (see page 1170)	
	IRConcrContinuousFootingSegment (see page 1173)	
	IRConcrContinuousFootingSpan (see page 1175)	

**I.3.1 IRConcrContinuousFooting****Class Hierarchy****C++**

```
interface IRConcrContinuousFooting : IDispatch;
```

**C#**

```
public interface IRConcrContinuousFooting;
```

**Visual Basic**

```
Public Interface IRConcrContinuousFooting
```

**Version**

Available since version 10.

**I.3.1.1 IRConcrContinuousFooting Members**

The following tables list the members exposed by IRConcrContinuousFooting.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CalculationOptions ( <a href="#">see page 1165</a> )	
◆	Concrete ( <a href="#">see page 1165</a> )	
◆	Geometry ( <a href="#">see page 1165</a> )	
◆	ImportOptions ( <a href="#">see page 1166</a> )	
◆	IsActive ( <a href="#">see page 1166</a> )	
◆	IsSelected ( <a href="#">see page 1166</a> )	
◆	Name ( <a href="#">see page 1166</a> )	
◆	NumberOfElements ( <a href="#">see page 1167</a> )	
◆	PatternOptions ( <a href="#">see page 1167</a> )	
◆	Reinforcement ( <a href="#">see page 1167</a> )	
◆	Steel ( <a href="#">see page 1167</a> )	
◆	StructureUserNo ( <a href="#">see page 1168</a> )	
◆	StructureUserNoCount ( <a href="#">see page 1168</a> )	
◆	UniqueId ( <a href="#">see page 1168</a> )	

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Activate ( <a href="#">see page 1169</a> )	
◆	Calculate ( <a href="#">see page 1169</a> )	
◆	CreateCalculationNoteRtf ( <a href="#">see page 1169</a> )	
◆	Save ( <a href="#">see page 1169</a> )	
◆	Verify ( <a href="#">see page 1170</a> )	

**I.3.1.2 IRConcrContinuousFooting Fields**

The fields of the IRConcrContinuousFooting class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CalculationOptions ( <a href="#">see page 1165</a> )	
◆	Concrete ( <a href="#">see page 1165</a> )	
◆	Geometry ( <a href="#">see page 1165</a> )	
◆	ImportOptions ( <a href="#">see page 1166</a> )	

❖	IsActive ( [ see page 1166 ] )	
❖	IsSelected ( [ see page 1166 ] )	
❖	Name ( [ see page 1166 ] )	
❖	NumberOfElements ( [ see page 1167 ] )	
❖	PatternOptions ( [ see page 1167 ] )	
❖	Reinforcement ( [ see page 1167 ] )	
❖	Steel ( [ see page 1167 ] )	
❖	StructureUserNo ( [ see page 1168 ] )	
❖	StructureUserNoCount ( [ see page 1168 ] )	
❖	UniqueId ( [ see page 1168 ] )	

### I.3.1.2.1 CalculationOptions

#### C++

```
HRESULT get_CalculationOptions( IRConcrBeamCalcOptions** );
```

#### C#

```
public IRConcrBeamCalcOptions CalculationOptions { get; }
```

#### Visual Basic

```
Public ReadOnly CalculationOptions As IRConcrBeamCalcOptions
```

#### Version

Available since version 10.

### I.3.1.2.2 Concrete

#### C++

```
HRESULT get_Concrete( IRConcrConcrete** );
```

#### C#

```
public IRConcrConcrete Concrete { get; }
```

#### Visual Basic

```
Public ReadOnly Concrete As IRConcrConcrete
```

#### Version

Available since version 10.

### I.3.1.2.3 Geometry

#### C++

```
HRESULT get_Geometry( IRConcrContinuousFootingGeometry** );
```

#### C#

```
public IRConcrContinuousFootingGeometry Geometry { get; }
```

#### Visual Basic

```
Public ReadOnly Geometry As IRConcrContinuousFootingGeometry
```

#### Version

Available since version 10.

#### I.3.1.2.4 ImportOptions

**C++**

```
HRESULT get_ImportOptions(IRConcrBeamImportOptions**);
```

**C#**

```
public IRConcrBeamImportOptions ImportOptions { get; }
```

**Visual Basic**

```
Public ReadOnly ImportOptions As IRConcrBeamImportOptions
```

**Version**

Available since version 10.

#### I.3.1.2.5 IsActive

**C++**

```
HRESULT get_IsActive(VARIANT_BOOL*);
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Version**

Available since version 10.

#### I.3.1.2.6 IsSelected

**C++**

```
HRESULT get_IsSelected(VARIANT_BOOL*);  
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

#### I.3.1.2.7 Name

**C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

### I.3.1.2.8 NumberOfElements

**C++**

```
HRESULT get_NumberOfElements(long*);  
HRESULT put_NumberOfElements(long);
```

**C#**

```
public long NumberOfElements { get; set; }
```

**Visual Basic**

```
Public NumberOfElements As long
```

**Version**

Available since version 12.

### I.3.1.2.9 PatternOptions

**C++**

```
HRESULT get_PatternOptions(IRConcrBeamPatternOptions**);
```

**C#**

```
public IRConcrBeamPatternOptions PatternOptions { get; }
```

**Visual Basic**

```
Public ReadOnly PatternOptions As IRConcrBeamPatternOptions
```

**Version**

Available since version 10.

### I.3.1.2.10 Reinforcement

**C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement**);
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

### I.3.1.2.11 Steel

**C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

### I.3.1.2.12 StructureUserNo

#### C++

```
HRESULT get_StructureUserNo(long* );
HRESULT put_StructureUserNo(long);
```

#### C#

```
public long StructureUserNo { get; set; }
```

#### Visual Basic

```
Public StructureUserNo As long
```

#### Version

Available since version 10.

### I.3.1.2.13 StructureUserNoCount

#### C++

```
HRESULT get_StructureUserNoCount(short* );
HRESULT put_StructureUserNoCount(short);
```

#### C#

```
public short StructureUserNoCount { get; set; }
```

#### Visual Basic

```
Public StructureUserNoCount As short
```

#### Version

Available since version 10.

### I.3.1.2.14 UniqueId

#### C++

```
HRESULT get_UniqueId(long* );
```

#### C#

```
public long UniqueId { get; }
```

#### Visual Basic

```
Public ReadOnly UniqueId As long
```

#### Version

Available since version 10.

### I.3.1.3 IRConcrContinuousFooting Methods

The methods of the IRConcrContinuousFooting class are listed here.

#### Public Methods

	Name	Description
⊕	Activate ( [ see page 1169 )	
⊕	Calculate ( [ see page 1169 )	
⊕	CreateCalculationNoteRtf ( [ see page 1169 )	
⊕	Save ( [ see page 1169 )	
⊕	Verify ( [ see page 1170 )	

### I.3.1.3.1 Activate

**C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

### I.3.1.3.2 Calculate

**C++**

```
HRESULT Calculate(VARIANT_BOOL _createReinforcement, VARIANT_BOOL* ret);
```

**C#**

```
public bool Calculate(bool _createReinforcement);
```

**Visual Basic**

```
Public Function Calculate(_createReinforcement As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.3.1.3.3 CreateCalculationNoteRtf

**C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

### I.3.1.3.4 Save

**C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.3.1.3.5 Verify****C++**

```
HRESULT Verify(VARIANT_BOOL _showErrors, long* ret);
```

**C#**

```
public long Verify(bool _showErrors);
```

**Visual Basic**

```
Public Function Verify(_showErrors As Boolean) As long
```

**Version**

Available since version 10.

**I.3.2 IRConcrContinuousFootingGeometry****Class Hierarchy****C++**

```
interface IRConcrContinuousFootingGeometry : IDispatch;
```

**C#**

```
public interface IRConcrContinuousFootingGeometry;
```

**Visual Basic**

```
Public Interface IRConcrContinuousFootingGeometry
```

**Version**

Available since version 10.

**I.3.2.1 IRConcrContinuousFootingGeometry Members**

The following tables list the members exposed by IRConcrContinuousFootingGeometry.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AutoNameSpans (see page 1171)	
❖	AutoNameSupports (see page 1171)	
❖	ConcreteVolume (see page 1172)	
❖	LeftCantilever (see page 1172)	
❖	RightCantilever (see page 1172)	
❖	ShutteringArea (see page 1172)	
❖	Span (see page 1173)	
❖	SpanNumber (see page 1173)	

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	UpdateInternalGeometryData (see page 1173)	

### I.3.2.2 IRConcrContinuousFootingGeometry Fields

The fields of the IRConcrContinuousFootingGeometry class are listed here.

#### Public Fields

	Name	Description
◆	AutoNameSpans (see page 1171)	
◆	AutoNameSupports (see page 1171)	
◆	ConcreteVolume (see page 1172)	
◆	LeftCantilever (see page 1172)	
◆	RightCantilever (see page 1172)	
◆	ShutteringArea (see page 1172)	
◆	Span (see page 1173)	
◆	SpanNumber (see page 1173)	

#### I.3.2.2.1 AutoNameSpans

##### C++

```
HRESULT get_AutoNameSpans(VARIANT_BOOL* );
HRESULT put_AutoNameSpans(VARIANT_BOOL);
```

##### C#

```
public bool AutoNameSpans { get; set; }
```

##### Visual Basic

```
Public AutoNameSpans As Boolean
```

##### Version

Available since version 10.

#### I.3.2.2.2 AutoNameSupports

##### C++

```
HRESULT get_AutoNameSupports(VARIANT_BOOL* );
HRESULT put_AutoNameSupports(VARIANT_BOOL);
```

##### C#

```
public bool AutoNameSupports { get; set; }
```

##### Visual Basic

```
Public AutoNameSupports As Boolean
```

##### Version

Available since version 10.

#### I.3.2.2.3 ConcreteVolume

##### C++

```
HRESULT get_ConcreteVolume(double* );
```

##### C#

```
public double ConcreteVolume { get; }
```

##### Visual Basic

```
Public ReadOnly ConcreteVolume As Double
```

**Version**

Available since version 10.

**I.3.2.2.4 LeftCantilever****C++**

```
HRESULT get_LeftCantilever(VARIANT_BOOL* );
HRESULT put_LeftCantilever(VARIANT_BOOL);
```

**C#**

```
public bool LeftCantilever { get; set; }
```

**Visual Basic**

```
Public LeftCantilever As Boolean
```

**Version**

Available since version 10.

**I.3.2.2.5 RightCantilever****C++**

```
HRESULT get_RightCantilever(VARIANT_BOOL* );
HRESULT put_RightCantilever(VARIANT_BOOL);
```

**C#**

```
public bool RightCantilever { get; set; }
```

**Visual Basic**

```
Public RightCantilever As Boolean
```

**Version**

Available since version 10.

**I.3.2.2.6 ShutteringArea****C++**

```
HRESULT get_ShutteringArea(double* );
```

**C#**

```
public double ShutteringArea { get; }
```

**Visual Basic**

```
Public ReadOnly ShutteringArea As Double
```

**Version**

Available since version 10.

**I.3.2.2.7 Span****C++**

```
HRESULT get_Span(IRConcrContinuousFootingSpan** );
```

**C#**

```
public IRConcrContinuousFootingSpan Span { get; }
```

**Visual Basic**

```
Public ReadOnly Span As IRConcrContinuousFootingSpan
```

**Version**

Available since version 10.

**I.3.2.2.8 SpanNumber****C++**

```
HRESULT get_SpanNumber(short* );
HRESULT put_SpanNumber(short);
```

**C#**

```
public short SpanNumber { get; set; }
```

**Visual Basic**

```
Public SpanNumber As short
```

**Version**

Available since version 10.

**I.3.2.3 IRConcrContinuousFootingGeometry Methods**

The methods of the IRConcrContinuousFootingGeometry class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	UpdateInternalGeometryData (  see page 1173)	

**I.3.2.3.1 UpdateInternalGeometryData****C++**

```
HRESULT UpdateInternalGeometryData();
```

**C#**

```
public void UpdateInternalGeometryData();
```

**Visual Basic**

```
Public Sub UpdateInternalGeometryData()
```

**Version**

Available since version 10.

**I.3.3 IRConcrContinuousFootingSegment****Class Hierarchy****C++**

```
interface IRConcrContinuousFootingSegment : IDispatch;
```

**C#**

```
public interface IRConcrContinuousFootingSegment;
```

**Visual Basic**

```
Public Interface IRConcrContinuousFootingSegment
```

**Version**

Available since version 10.

### I.3.3.1 IRConcrContinuousFootingSegment Members

The following tables list the members exposed by IRConcrContinuousFootingSegment.

#### Public Fields

	Name	Description
◆	Dim ( <a href="#">see page 1174</a> )	
◆	SectionType ( <a href="#">see page 1174</a> )	

### I.3.3.2 IRConcrContinuousFootingSegment Fields

The fields of the IRConcrContinuousFootingSegment class are listed here.

#### Public Fields

	Name	Description
◆	Dim ( <a href="#">see page 1174</a> )	
◆	SectionType ( <a href="#">see page 1174</a> )	

#### I.3.3.2.1 Dim

##### C++

```
HRESULT get_Dim(double* );
HRESULT put_Dim(double);
```

##### C#

```
public double Dim { get; set; }
```

##### Visual Basic

```
Public Dim As Double
```

##### Version

Available since version 10.

#### I.3.3.2.2 SectionType

##### C++

```
HRESULT get_SectionType(IRConcrContinuousFootingSectionType* );
HRESULT put_SectionType(IRConcrContinuousFootingSectionType);
```

##### C#

```
public IRConcrContinuousFootingSectionType SectionType { get; set; }
```

##### Visual Basic

```
Public SectionType As IRConcrContinuousFootingSectionType
```

##### Version

Available since version 10.

### I.3.4 IRConcrContinuousFootingSpan

#### Class Hierarchy

##### C++

```
interface IRConcrContinuousFootingSpan : IDispatch;
```

##### C#

```
public interface IRConcrContinuousFootingSpan;
```

## Visual Basic

```
Public Interface IRConcrContinuousFootingSpan
```

### Version

Available since version 10.

#### I.3.4.1 IRConcrContinuousFootingSpan Members

The following tables list the members exposed by IRConcrContinuousFootingSpan.

### Public Fields

	Name	Description
◆	CalculationLength ( <a href="#">see page 1176</a> )	
◆	LeftSupport ( <a href="#">see page 1176</a> )	
◆	Length ( <a href="#">see page 1176</a> )	
◆	Name ( <a href="#">see page 1176</a> )	
◆	RightSupport ( <a href="#">see page 1177</a> )	
◆	Segment ( <a href="#">see page 1177</a> )	
◆	SegmentNumber ( <a href="#">see page 1177</a> )	

#### I.3.4.2 IRConcrContinuousFootingSpan Fields

The fields of the IRConcrContinuousFootingSpan class are listed here.

### Public Fields

	Name	Description
◆	CalculationLength ( <a href="#">see page 1176</a> )	
◆	LeftSupport ( <a href="#">see page 1176</a> )	
◆	Length ( <a href="#">see page 1176</a> )	
◆	Name ( <a href="#">see page 1176</a> )	
◆	RightSupport ( <a href="#">see page 1177</a> )	
◆	Segment ( <a href="#">see page 1177</a> )	
◆	SegmentNumber ( <a href="#">see page 1177</a> )	

#### I.3.4.2.1 CalculationLength

##### C++

```
HRESULT get_CalculationLength(double*);
```

##### C#

```
public double CalculationLength { get; }
```

## Visual Basic

```
Public ReadOnly CalculationLength As Double
```

### Version

Available since version 10.

#### I.3.4.2.2 LeftSupport

##### C++

```
HRESULT get_LeftSupport(IRConcrContinuousFootingSupport**);
```

**C#**

```
public IRConcrContinuousFootingSupport  
LeftSupport { get; }
```

**Visual Basic**

```
Public ReadOnly LeftSupport As IRConcrContinuousFootingSupport
```

**Version**

Available since version 10.

**I.3.4.2.3 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Version**

Available since version 10.

**I.3.4.2.4 Name****C++**

```
HRESULT get_Name(BSTR*);  
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Version**

Available since version 10.

**I.3.4.2.5 RightSupport****C++**

```
HRESULT get_RightSupport(IRConcrContinuousFootingSupport  
**);
```

**C#**

```
public IRConcrContinuousFootingSupport  
RightSupport { get; }
```

**Visual Basic**

```
Public ReadOnly RightSupport As IRConcrContinuousFootingSupport
```

**Version**

Available since version 10.

### I.3.4.2.6 Segment

#### C++

```
HRESULT get_Segment( IRConcrContinuousFootingSegment** );
```

#### C#

```
public IRConcrContinuousFootingSegment Segment { get; }
```

#### Visual Basic

```
Public ReadOnly Segment As IRConcrContinuousFootingSegment
```

#### Version

Available since version 10.

### I.3.4.2.7 SegmentNumber

#### C++

```
HRESULT get_SegmentNumber(short* );
HRESULT put_SegmentNumber(short);
```

#### C#

```
public short SegmentNumber { get; set; }
```

#### Visual Basic

```
Public SegmentNumber As short
```

#### Version

Available since version 10.

## I.3.5 IRConcrContinuousFootingSectionType

#### C++

```
enum IRConcrContinuousFootingSectionType;
```

#### C#

```
public enum IRConcrContinuousFootingSectionType;
```

#### Visual Basic

```
Public Enum IRConcrContinuousFootingSectionType
```

#### Members

Members	Description
I_CBS_CFOOT_SECTION_NP = 1	Available since version 10.
I_CBS_CFOOT_SECTION_LP = 2	Available since version 10.
I_CBS_CFOOT_SECTION_RP = 3	Available since version 10.
I_CBS_CFOOT_SECTION_BP = 4	Available since version 10.

#### Version

Available since version 10.

## I.3.6 IRConcrContinuousFootingSectionDimType

#### C++

```
enum IRConcrContinuousFootingSectionDimType;
```

**C#**

```
public enum IRConcrContinuousFootingSectionDimType;
```

**Visual Basic**

```
Public Enum IRConcrContinuousFootingSectionDimType
```

**Members**

Members	Description
I_BSD_CFOOT_SECTION_DIM_L = 1	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_H1 = 2	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_H2 = 3	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_B = 4	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_EL1 = 5	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_ER1 = 6	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_PL = 7	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_RL = 8	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_PR = 9	Available since version 10.
I_BSD_CFOOT_SECTION_DIM_RR = 10	Available since version 10.

**Version**

Available since version 10.

**I.3.7 IRConcrContinuousFootingSupportMaterialType****C++**

```
enum IRConcrContinuousFootingSupportMaterialType;
```

**C#**

```
public enum IRConcrContinuousFootingSupportMaterialType;
```

**Visual Basic**

```
Public Enum IRConcrContinuousFootingSupportMaterialType
```

**Members**

Members	Description
I_BSM_CFOOT_SUPPORT_CONCRETE = 1	Available since version 10.
I_BSM_CFOOT_SUPPORT_MASONRY = 2	Available since version 10.

**Version**

Available since version 10.

**I.4 RC Column**

Available since version 10.

**Enumerations**

	Name	Description
	IRConcrColumnDimensionType (see page 1195)	
	IRConcrColumnSectionType (see page 1195)	

	IRConcrColumnLoadCaseType (see page 1207)	
--	-------------------------------------------	--

## Interfaces

	Name	Description
	IRConcrColumn (see page 1179)	
	IRConcrColumnCalcOptions (see page 1188)	
	IRConcrColumnGeometry (see page 1189)	
	IRConcrColumnImportOptions (see page 1191)	
	IRConcrColumnPatternOptions (see page 1194)	
	IRConcrColumnBucklingModel (see page 1196)	
	IRConcrColumnLoads (see page 1200)	
	IRConcrColumnLoad (see page 1202)	

## I.4.1 IRConcrColumn

### Class Hierarchy

#### C++

```
interface IRConcrColumn : IDispatch;
```

#### C#

```
public interface IRConcrColumn;
```

### Visual Basic

```
Public Interface IRConcrColumn
```

### Version

Available since version 10.

### I.4.1.1 IRConcrColumn Members

The following tables list the members exposed by IRConcrColumn.

#### Public Fields

	Name	Description
	BucklingModel (see page 1181)	
	CalculationOptions (see page 1182)	
	Concrete (see page 1182)	
	Geometry (see page 1182)	
	HasUpperColumn (see page 1182)	
	ImportOptions (see page 1183)	
	IsActive (see page 1183)	
	IsSelected (see page 1183)	
	Loads (see page 1183)	
	Name (see page 1184)	

◆	NumberOfElements ( [ see page 1184) )	
◆	PatternOptions ( [ see page 1184) )	
◆	Reinforcement ( [ see page 1184) )	
◆	Steel ( [ see page 1185) )	
◆	StructureUserNo ( [ see page 1185) )	
◆	StructureUserNoCount ( [ see page 1185) )	
◆	UniqueId ( [ see page 1185) )	
◆	UpperColumn ( [ see page 1186) )	
◆	UpperColumnDY ( [ see page 1186) )	
◆	UpperColumnDZ ( [ see page 1186) )	

## Public Methods

	Name	Description
≡	Activate ( [ see page 1187) )	
≡	Calculate ( [ see page 1187) )	
≡	CreateCalculationNoteRtf ( [ see page 1187) )	
≡	CreateFromBars ( [ see page 1187) )	
≡	Save ( [ see page 1188) )	
≡	Verify ( [ see page 1188) )	

### I.4.1.2 IRConcrColumn Fields

The fields of the IRConcrColumn class are listed here.

## Public Fields

	Name	Description
◆	BucklingModel ( [ see page 1181) )	
◆	CalculationOptions ( [ see page 1182) )	
◆	Concrete ( [ see page 1182) )	
◆	Geometry ( [ see page 1182) )	
◆	HasUpperColumn ( [ see page 1182) )	
◆	ImportOptions ( [ see page 1183) )	
◆	IsActive ( [ see page 1183) )	
◆	IsSelected ( [ see page 1183) )	
◆	Loads ( [ see page 1183) )	
◆	Name ( [ see page 1184) )	
◆	NumberOfElements ( [ see page 1184) )	
◆	PatternOptions ( [ see page 1184) )	
◆	Reinforcement ( [ see page 1184) )	
◆	Steel ( [ see page 1185) )	
◆	StructureUserNo ( [ see page 1185) )	
◆	StructureUserNoCount ( [ see page 1185) )	
◆	UniqueId ( [ see page 1185) )	
◆	UpperColumn ( [ see page 1186) )	
◆	UpperColumnDY ( [ see page 1186) )	



UpperColumnDZ (see page 1186)

#### I.4.1.2.1 BucklingModel

**C++**

```
HRESULT get_BucklingModel( IRConcrColumnBucklingModel** );
```

**C#**

```
public IRConcrColumnBucklingModel BucklingModel { get; }
```

**Visual Basic**

```
Public ReadOnly BucklingModel As IRConcrColumnBucklingModel
```

**Version**

Available since version 11.

#### I.4.1.2.2 CalculationOptions

**C++**

```
HRESULT get_CalculationOptions( IRConcrColumnCalcOptions** );
```

**C#**

```
public IRConcrColumnCalcOptions CalculationOptions { get; }
```

**Visual Basic**

```
Public ReadOnly CalculationOptions As IRConcrColumnCalcOptions
```

**Version**

Available since version 10.

#### I.4.1.2.3 Concrete

**C++**

```
HRESULT get_Concrete( IRConcrConcrete** );
```

**C#**

```
public IRConcrConcrete Concrete { get; }
```

**Visual Basic**

```
Public ReadOnly Concrete As IRConcrConcrete
```

**Version**

Available since version 10.

#### I.4.1.2.4 Geometry

**C++**

```
HRESULT get_Geometry( IRConcrColumnGeometry** );
```

**C#**

```
public IRConcrColumnGeometry Geometry { get; }
```

**Visual Basic**

```
Public ReadOnly Geometry As IRConcrColumnGeometry
```

**Version**

Available since version 10.

#### I.4.1.2.5 HasUpperColumn

**C++**

```
HRESULT get_HasUpperColumn(VARIANT_BOOL*);
```

**C#**

```
public bool HasUpperColumn { get; }
```

**Visual Basic**

```
Public ReadOnly HasUpperColumn As Boolean
```

**Version**

Available since version 12.

#### I.4.1.2.6 ImportOptions

**C++**

```
HRESULT get_ImportOptions(IRConcrColumnImportOptions**);
```

**C#**

```
public IRConcrColumnImportOptions ImportOptions { get; }
```

**Visual Basic**

```
Public ReadOnly ImportOptions As IRConcrColumnImportOptions
```

**Version**

Available since version 10.

#### I.4.1.2.7 IsActive

**C++**

```
HRESULT get_IsActive(VARIANT_BOOL*);
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Version**

Available since version 10.

#### I.4.1.2.8 IsSelected

**C++**

```
HRESULT get_IsSelected(VARIANT_BOOL*);  
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

#### I.4.1.2.9 Loads

**C++**

```
HRESULT get_Loads( IRConcrColumnLoads** );
```

**C#**

```
public IRConcrColumnLoads Loads { get; }
```

**Visual Basic**

```
Public ReadOnly Loads As IRConcrColumnLoads
```

**Version**

Available since version 11.

#### I.4.1.2.10 Name

**C++**

```
HRESULT get_Name(BSTR* );
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

#### I.4.1.2.11 NumberOfElements

**C++**

```
HRESULT get_NumberOfElements( long* );
HRESULT put_NumberOfElements( long );
```

**C#**

```
public long NumberOfElements { get; set; }
```

**Visual Basic**

```
Public NumberOfElements As Long
```

**Version**

Available since version 10.

#### I.4.1.2.12 PatternOptions

**C++**

```
HRESULT get_PatternOptions( IRConcrColumnPatternOptions** );
```

**C#**

```
public IRConcrColumnPatternOptions PatternOptions { get; }
```

**Visual Basic**

```
Public ReadOnly PatternOptions As IRConcrColumnPatternOptions
```

**Version**

Available since version 10.

#### I.4.1.2.13 Reinforcement

**C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement**);
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

#### I.4.1.2.14 Steel

**C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

#### I.4.1.2.15 StructureUserNo

**C++**

```
HRESULT get_StructureUserNo(long*);  
HRESULT put_StructureUserNo(long);
```

**C#**

```
public long StructureUserNo { get; set; }
```

**Visual Basic**

```
Public StructureUserNo As long
```

**Version**

Available since version 10.

#### I.4.1.2.16 StructureUserNoCount

**C++**

```
HRESULT get_StructureUserNoCount(short*);  
HRESULT put_StructureUserNoCount(short);
```

**C#**

```
public short StructureUserNoCount { get; set; }
```

**Visual Basic**

```
Public StructureUserNoCount As short
```

**Version**

Available since version 10.

#### I.4.1.2.17 UniqueId

**C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Version**

Available since version 10.

#### I.4.1.2.18 UpperColumn

**C++**

```
HRESULT get_UpperColumn(IRConcrColumn**);
HRESULT put_UpperColumn(IRConcrColumn*);
```

**C#**

```
public IRConcrColumn UpperColumn { get; set; }
```

**Visual Basic**

```
Public UpperColumn As IRConcrColumn
```

**Version**

Available since version 12.

#### I.4.1.2.19 UpperColumnDY

**C++**

```
HRESULT get_UpperColumnDY(double* );
HRESULT put_UpperColumnDY(double);
```

**C#**

```
public double UpperColumnDY { get; set; }
```

**Visual Basic**

```
Public UpperColumnDY As double
```

**Version**

Available since version 12.

#### I.4.1.2.20 UpperColumnDZ

**C++**

```
HRESULT get_UpperColumnDZ(double* );
HRESULT put_UpperColumnDZ(double);
```

**C#**

```
public double UpperColumnDZ { get; set; }
```

**Visual Basic**

```
Public UpperColumnDZ As double
```

## Version

Available since version 12.

### I.4.1.3 IRConcrColumn Methods

The methods of the IRConcrColumn class are listed here.

#### Public Methods

	Name	Description
💡	Activate (🔗 see page 1187)	
💡	Calculate (🔗 see page 1187)	
💡	CreateCalculationNoteRtf (🔗 see page 1187)	
💡	CreateFromBars (🔗 see page 1187)	
💡	Save (🔗 see page 1188)	
💡	Verify (🔗 see page 1188)	

#### I.4.1.3.1 Activate

##### C++

```
HRESULT Activate();
```

##### C#

```
public void Activate();
```

##### Visual Basic

```
Public Sub Activate()
```

#### Version

Available since version 10.

#### I.4.1.3.2 Calculate

##### C++

```
HRESULT Calculate(VARIANT_BOOL _createReinforcement, VARIANT_BOOL* ret);
```

##### C#

```
public bool Calculate(bool _createReinforcement);
```

##### Visual Basic

```
Public Function Calculate(_createReinforcement As Boolean) As Boolean
```

#### Version

Available since version 10.

#### I.4.1.3.3 CreateCalculationNoteRtf

##### C++

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

##### C#

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

**I.4.1.3.4 CreateFromBars****C++**

```
HRESULT CreateFromBars(BSTR _selText, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromBars(String _selText);
```

**Visual Basic**

```
Public Function CreateFromBars(_selText As String) As Boolean
```

**Version**

Available since version 10.

**I.4.1.3.5 Save****C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.4.1.3.6 Verify****C++**

```
HRESULT Verify(VARIANT_BOOL _showErrors, long* ret);
```

**C#**

```
public long Verify(bool _showErrors);
```

**Visual Basic**

```
Public Function Verify(_showErrors As Boolean) As long
```

**Version**

Available since version 10.

**I.4.2 IRConcrColumnCalcOptions****Class Hierarchy****C++**

```
interface IRConcrColumnCalcOptions : IDispatch;
```

**C#**

```
public interface IRConcrColumnCalcOptions;
```

**Visual Basic**

```
Public Interface IRConcrColumnCalcOptions
```

**Version**

Available since version 10.

**I.4.2.1 IRConcrColumnCalcOptions Members**

The following tables list the members exposed by IRConcrColumnCalcOptions.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Cover (↗ see page 1189)	

**I.4.2.2 IRConcrColumnCalcOptions Fields**

The fields of the IRConcrColumnCalcOptions class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Cover (↗ see page 1189)	

**I.4.2.2.1 Cover****C++**

```
HRESULT get_Cover(double* );
HRESULT put_Cover(double);
```

**C#**

```
public double Cover { get; set; }
```

**Visual Basic**

```
Public Cover As Double
```

**Version**

Available since version 10.

**I.4.3 IRConcrColumnGeometry****Class Hierarchy****C++**

```
interface IRConcrColumnGeometry : IDispatch;
```

**C#**

```
public interface IRConcrColumnGeometry;
```

**Visual Basic**

```
Public Interface IRConcrColumnGeometry
```

**Version**

Available since version 10.

### I.4.3.1 IRConcrColumnGeometry Members

The following tables list the members exposed by IRConcrColumnGeometry.

#### Public Fields

	Name	Description
◆	ConcreteVolume (see page 1190)	
◆	Dim (see page 1190)	
◆	DimIsFixed (see page 1190)	
◆	Section (see page 1191)	
◆	SectionName (see page 1191)	
◆	ShutteringArea (see page 1191)	

### I.4.3.2 IRConcrColumnGeometry Fields

The fields of the IRConcrColumnGeometry class are listed here.

#### Public Fields

	Name	Description
◆	ConcreteVolume (see page 1190)	
◆	Dim (see page 1190)	
◆	DimIsFixed (see page 1190)	
◆	Section (see page 1191)	
◆	SectionName (see page 1191)	
◆	ShutteringArea (see page 1191)	

#### I.4.3.2.1 ConcreteVolume

##### C++

```
HRESULT get_ConcreteVolume(double*);
```

##### C#

```
public double ConcreteVolume { get; }
```

##### Visual Basic

```
Public ReadOnly ConcreteVolume As double
```

##### Version

Available since version 10.

#### I.4.3.2.2 Dim

##### C++

```
HRESULT get_Dim(double*);  
HRESULT put_Dim(double);
```

##### C#

```
public double Dim { get; set; }
```

##### Visual Basic

```
Public Dim As double
```

##### Version

Available since version 10.

#### I.4.3.2.3 DimIsFixed

**C++**

```
HRESULT get_DimIsFixed(VARIANT_BOOL* );
HRESULT put_DimIsFixed(VARIANT_BOOL);
```

**C#**

```
public bool DimIsFixed { get; set; }
```

**Visual Basic**

```
Public DimIsFixed As Boolean
```

**Version**

Available since version 11.

#### I.4.3.2.4 Section

**C++**

```
HRESULT get_Section(IRCConcrColumnSectionType* );
HRESULT put_Section(IRCConcrColumnSectionType);
```

**C#**

```
public IRCConcrColumnSectionType Section { get; set; }
```

**Visual Basic**

```
Public Section As IRCConcrColumnSectionType
```

**Version**

Available since version 10.

#### I.4.3.2.5 SectionName

**C++**

```
HRESULT get_SectionName(BSTR* );
HRESULT put_SectionName(BSTR);
```

**C#**

```
public String SectionName { get; set; }
```

**Visual Basic**

```
Public SectionName As String
```

**Version**

Available since version 10.

#### I.4.3.2.6 ShutteringArea

**C++**

```
HRESULT get_ShutteringArea(double* );
```

**C#**

```
public double ShutteringArea { get; }
```

**Visual Basic**

```
Public ReadOnly ShutteringArea As double
```

## Version

Available since version 10.

## I.4.4 IRConcrColumnImportOptions

### Class Hierarchy

#### C++

```
interface IRConcrColumnImportOptions : IDispatch;
```

#### C#

```
public interface IRConcrColumnImportOptions;
```

### Visual Basic

```
Public Interface IRConcrColumnImportOptions
```

## Version

Available since version 10.

### I.4.4.1 IRConcrColumnImportOptions Members

The following tables list the members exposed by IRConcrColumnImportOptions.

#### Public Fields

	Name	Description
❖	GroupByGeometry ( <a href="#">see page 1192</a> )	
❖	GroupByLevel ( <a href="#">see page 1193</a> )	
❖	ImportCombinations ( <a href="#">see page 1193</a> )	
❖	RunCalcAuto ( <a href="#">see page 1193</a> )	
❖	ShowDialog ( <a href="#">see page 1193</a> )	

### I.4.4.2 IRConcrColumnImportOptions Fields

The fields of the IRConcrColumnImportOptions class are listed here.

#### Public Fields

	Name	Description
❖	GroupByGeometry ( <a href="#">see page 1192</a> )	
❖	GroupByLevel ( <a href="#">see page 1193</a> )	
❖	ImportCombinations ( <a href="#">see page 1193</a> )	
❖	RunCalcAuto ( <a href="#">see page 1193</a> )	
❖	ShowDialog ( <a href="#">see page 1193</a> )	

### I.4.4.2.1 GroupByGeometry

#### C++

```
HRESULT get_GroupByGeometry(VARIANT_BOOL* );
HRESULT put_GroupByGeometry(VARIANT_BOOL);
```

#### C#

```
public bool GroupByGeometry { get; set; }
```

**Visual Basic**

```
Public GroupByGeometry As Boolean
```

**Version**

Available since version 10.

**I.4.4.2.2 GroupByLevel****C++**

```
HRESULT get_GroupByLevel(VARIANT_BOOL* );
HRESULT put_GroupByLevel(VARIANT_BOOL);
```

**C#**

```
public bool GroupByLevel { get; set; }
```

**Visual Basic**

```
Public GroupByLevel As Boolean
```

**Version**

Available since version 10.

**I.4.4.2.3 ImportCombinations****C++**

```
HRESULT get_ImportCombinations(VARIANT_BOOL* );
HRESULT put_ImportCombinations(VARIANT_BOOL);
```

**C#**

```
public bool ImportCombinations { get; set; }
```

**Visual Basic**

```
Public ImportCombinations As Boolean
```

**Version**

Available since version 10.

**I.4.4.2.4 RunCalcAuto****C++**

```
HRESULT get_RunCalcAuto(VARIANT_BOOL* );
HRESULT put_RunCalcAuto(VARIANT_BOOL);
```

**C#**

```
public bool RunCalcAuto { get; set; }
```

**Visual Basic**

```
Public RunCalcAuto As Boolean
```

**Version**

Available since version 10.

**I.4.4.2.5 ShowDialog****C++**

```
HRESULT get_ShowDialog(VARIANT_BOOL* );
HRESULT put_ShowDialog(VARIANT_BOOL);
```

**C#**

```
public bool ShowDialog { get; set; }
```

**Visual Basic**

```
Public ShowDialog As Boolean
```

**Version**

Available since version 10.

**I.4.5 IRConcrColumnPatternOptions****Class Hierarchy****C++**

```
interface IRConcrColumnPatternOptions : IDispatch;
```

**C#**

```
public interface IRConcrColumnPatternOptions;
```

**Visual Basic**

```
Public Interface IRConcrColumnPatternOptions
```

**Version**

Available since version 10.

**I.4.5.1 IRConcrColumnPatternOptions Members**

The following tables list the members exposed by IRConcrColumnPatternOptions.

**Public Fields**

	Name	Description
◆	TiesToBeam (see page 1194)	

**I.4.5.2 IRConcrColumnPatternOptions Fields**

The fields of the IRConcrColumnPatternOptions class are listed here.

**Public Fields**

	Name	Description
◆	TiesToBeam (see page 1194)	

**I.4.5.2.1 TiesToBeam****C++**

```
HRESULT get_TiesToBeam(VARIANT_BOOL* );
HRESULT put_TiesToBeam(VARIANT_BOOL);
```

**C#**

```
public bool TiesToBeam { get; set; }
```

**Visual Basic**

```
Public TiesToBeam As Boolean
```

**Version**

Available since version 10.

## I.4.6 IRConcrColumnDimensionType

### C++

```
enum IRConcrColumnDimensionType;
```

### C#

```
public enum IRConcrColumnDimensionType;
```

### Visual Basic

```
Public Enum IRConcrColumnDimensionType
```

### Members

Members	Description
I_CCD_COLUMN_DIM_HSP = 1	Available since version 10.
I_CCD_COLUMN_DIM_HSD = 2	Available since version 10.
I_CCD_COLUMN_DIM_EPD = 3	Available since version 10.
I_CCD_COLUMN_DIM_A = 4	Available since version 10.
I_CCD_COLUMN_DIM_B = 5	Available since version 10.
I_CCD_COLUMN_DIM_C = 6	Available since version 10.
I_CCD_COLUMN_DIM_H1 = 7	Available since version 10.
I_CCD_COLUMN_DIM_L1 = 8	Available since version 10.
I_CCD_COLUMN_DIM_H2 = 9	Available since version 10.
I_CCD_COLUMN_DIM_L2 = 10	Available since version 10.
I_CCD_COLUMN_DIM_N = 11	Available since version 10.
I_CCD_COLUMN_DIM_DE = 12	Available since version 10.
I_CCD_COLUMN_DIM_HSP_X = 1	Available since version 10.
I_CCD_COLUMN_DIM_HSP_Y = 13	Available since version 10.

### Version

Available since version 10.

## I.4.7 IRConcrColumnSectionType

### C++

```
enum IRConcrColumnSectionType;
```

### C#

```
public enum IRConcrColumnSectionType;
```

### Visual Basic

```
Public Enum IRConcrColumnSectionType
```

### Members

Members	Description
I_CCT_COLUMN_SEC_R = 1	Available since version 10.
I_CCT_COLUMN_SEC_L = 2	Available since version 10.
I_CCT_COLUMN_SEC_TC = 3	Available since version 10.
I_CCT_COLUMN_SEC_Z = 4	Available since version 10.
I_CCT_COLUMN_SEC_P = 5	Available since version 10.
I_CCT_COLUMN_SEC_C36 = 6	Available since version 10.
I_CCT_COLUMN_SEC_C18 = 7	Available since version 10.

I_CCT_COLUMN_SEC_C90 = 8	Available since version 10.
--------------------------	-----------------------------

## Version

Available since version 10.

## I.4.8 IRConcrColumnBucklingModel

### Class Hierarchy

#### C++

```
interface IRConcrColumnBucklingModel : IDispatch;
```

#### C#

```
public interface IRConcrColumnBucklingModel;
```

### Visual Basic

```
Public Interface IRConcrColumnBucklingModel
```

## Version

Available since version 11.

### I.4.8.1 IRConcrColumnBucklingModel Members

The following tables list the members exposed by IRConcrColumnBucklingModel.

#### Public Fields

	Name	Description
◆	IsDirectionYOff (see page 1197)	
◆	IsDirectionZOff (see page 1197)	
◆	IsSwayY (see page 1197)	
◆	IsSwayZ (see page 1198)	
◆	IsTotalStructureHeight (see page 1198)	
◆	ky (see page 1198)	
◆	kz (see page 1198)	
◆	Ly (see page 1199)	
◆	Lz (see page 1199)	
◆	NumberOfStories (see page 1199)	
◆	TotalStructureHeight (see page 1199)	

### I.4.8.2 IRConcrColumnBucklingModel Fields

The fields of the IRConcrColumnBucklingModel class are listed here.

#### Public Fields

	Name	Description
◆	IsDirectionYOff (see page 1197)	
◆	IsDirectionZOff (see page 1197)	
◆	IsSwayY (see page 1197)	
◆	IsSwayZ (see page 1198)	
◆	IsTotalStructureHeight (see page 1198)	
◆	ky (see page 1198)	

◆	kz (see page 1198)	
◆	Ly (see page 1199)	
◆	Lz (see page 1199)	
◆	NumberOfStories (see page 1199)	
◆	TotalStructureHeight (see page 1199)	

#### I.4.8.2.1 IsDirectionYOff

##### C++

```
HRESULT get_IsDirectionYOff(VARIANT_BOOL* );
HRESULT put_IsDirectionYOff(VARIANT_BOOL);
```

##### C#

```
public bool IsDirectionYOff { get; set; }
```

##### Visual Basic

```
Public IsDirectionYOff As Boolean
```

##### Version

Available since version 11.

#### I.4.8.2.2 IsDirectionZOff

##### C++

```
HRESULT get_IsDirectionZOff(VARIANT_BOOL* );
HRESULT put_IsDirectionZOff(VARIANT_BOOL);
```

##### C#

```
public bool IsDirectionZOff { get; set; }
```

##### Visual Basic

```
Public IsDirectionZOff As Boolean
```

##### Version

Available since version 11.

#### I.4.8.2.3 IsSwayY

##### C++

```
HRESULT get_IsSwayY(VARIANT_BOOL* );
HRESULT put_IsSwayY(VARIANT_BOOL);
```

##### C#

```
public bool IsSwayY { get; set; }
```

##### Visual Basic

```
Public IsSwayY As Boolean
```

##### Version

Available since version 11.

#### I.4.8.2.4 IsSwayZ

##### C++

```
HRESULT get_IsSwayZ(VARIANT_BOOL* );
```

```
HRESULT put_IsSwayZ(VARIANT_BOOL);
```

**C#**

```
public bool IsSwayZ { get; set; }
```

**Visual Basic**

```
Public IsSwayZ As Boolean
```

**Version**

Available since version 11.

**I.4.8.2.5 IsTotalStructureHeight****C++**

```
HRESULT get_IsTotalStructureHeight(VARIANT_BOOL*);  
HRESULT put_IsTotalStructureHeight(VARIANT_BOOL);
```

**C#**

```
public bool IsTotalStructureHeight { get; set; }
```

**Visual Basic**

```
Public IsTotalStructureHeight As Boolean
```

**Version**

Available since version 11.

**I.4.8.2.6 ky****C++**

```
HRESULT get_ky(double*);  
HRESULT put_ky(double);
```

**C#**

```
public double ky { get; set; }
```

**Visual Basic**

```
Public ky As double
```

**Version**

Available since version 11.

**I.4.8.2.7 kz****C++**

```
HRESULT get_kz(double*);  
HRESULT put_kz(double);
```

**C#**

```
public double kz { get; set; }
```

**Visual Basic**

```
Public kz As double
```

**Version**

Available since version 11.

### I.4.8.2.8 Ly

#### C++

```
HRESULT get_Ly(double*);  
HRESULT put_Ly(double);
```

#### C#

```
public double Ly { get; set; }
```

#### Visual Basic

```
Public Ly As Double
```

#### Version

Available since version 11.

### I.4.8.2.9 Lz

#### C++

```
HRESULT get_Lz(double*);  
HRESULT put_Lz(double);
```

#### C#

```
public double Lz { get; set; }
```

#### Visual Basic

```
Public Lz As Double
```

#### Version

Available since version 11.

### I.4.8.2.10 NumberOfStories

#### C++

```
HRESULT get_NumberOfStories( long*);  
HRESULT put_NumberOfStories( long);
```

#### C#

```
public long NumberOfStories { get; set; }
```

#### Visual Basic

```
Public NumberOfStories As Long
```

#### Version

Available since version 11.

### I.4.8.2.11 TotalStructureHeight

#### C++

```
HRESULT get_TotalStructureHeight( double*);  
HRESULT put_TotalStructureHeight( double);
```

#### C#

```
public double TotalStructureHeight { get; set; }
```

#### Visual Basic

```
Public TotalStructureHeight As Double
```

**Version**

Available since version 11.

**I.4.9 IRConcrColumnLoads****Class Hierarchy****C++**

```
interface IRConcrColumnLoads : IDispatch;
```

**C#**

```
public interface IRConcrColumnLoads;
```

**Visual Basic**

```
Public Interface IRConcrColumnLoads
```

**Version**

Available since version 11.

**I.4.9.1 IRConcrColumnLoads Members**

The following tables list the members exposed by IRConcrColumnLoads.

**Public Fields**

	Name	Description
◆	Count (↗ see page 1200)	
◆	Item (↗ see page 1201)	

**Public Methods**

	Name	Description
☞	Add (↗ see page 1201)	
☞	RemoveAll (↗ see page 1201)	

**I.4.9.2 IRConcrColumnLoads Fields**

The fields of the IRConcrColumnLoads class are listed here.

**Public Fields**

	Name	Description
◆	Count (↗ see page 1200)	
◆	Item (↗ see page 1201)	

**I.4.9.2.1 Count****C++**

```
HRESULT get_Count(long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Version**

Available since version 11.

### I.4.9.2.2 Item

#### C++

```
HRESULT get_Item( IRConcrColumnLoad** );
```

#### C#

```
public IRConcrColumnLoad Item { get; }
```

#### Visual Basic

```
Public ReadOnly Item As IRConcrColumnLoad
```

#### Version

Available since version 11.

### I.4.9.3 IRConcrColumnLoads Methods

The methods of the IRConcrColumnLoads class are listed here.

#### Public Methods

	Name	Description
💡	Add (see page 1201)	
💡	RemoveAll (see page 1201)	

### I.4.9.3.1 Add

#### C++

```
HRESULT Add( IRConcrColumnLoad** ret );
```

#### C#

```
public IRConcrColumnLoad Add();
```

#### Visual Basic

```
Public Function Add() As IRConcrColumnLoad
```

#### Version

Available since version 11.

### I.4.9.3.2 RemoveAll

#### C++

```
HRESULT RemoveAll();
```

#### C#

```
public void RemoveAll();
```

#### Visual Basic

```
Public Sub RemoveAll()
```

#### Version

Available since version 11.

## I.4.10 IRConcrColumnLoad

### Class Hierarchy

**C++**

```
interface IRConcrColumnLoad : IDispatch;
```

**C#**

```
public interface IRConcrColumnLoad;
```

**Visual Basic**

```
Public Interface IRConcrColumnLoad
```

**Version**

Available since version 11.

**I.4.10.1 IRConcrColumnLoad Members**

The following tables list the members exposed by IRConcrColumnLoad.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CaseName (❑ see page 1203)	
❖	CaseType (❑ see page 1203)	
❖	Fy (❑ see page 1203)	
❖	Fz (❑ see page 1204)	
❖	Gamma (❑ see page 1204)	
❖	MnsY (❑ see page 1204)	
❖	MnsZ (❑ see page 1204)	
❖	MyA (❑ see page 1205)	
❖	MyB (❑ see page 1205)	
❖	MyC (❑ see page 1205)	
❖	MzA (❑ see page 1205)	
❖	MzB (❑ see page 1206)	
❖	MzC (❑ see page 1206)	
❖	N (❑ see page 1206)	
❖	NdN (❑ see page 1206)	

**I.4.10.2 IRConcrColumnLoad Fields**

The fields of the IRConcrColumnLoad class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CaseName (❑ see page 1203)	
❖	CaseType (❑ see page 1203)	
❖	Fy (❑ see page 1203)	
❖	Fz (❑ see page 1204)	
❖	Gamma (❑ see page 1204)	
❖	MnsY (❑ see page 1204)	
❖	MnsZ (❑ see page 1204)	
❖	MyA (❑ see page 1205)	
❖	MyB (❑ see page 1205)	
❖	MyC (❑ see page 1205)	
❖	MzA (❑ see page 1205)	
❖	MzB (❑ see page 1206)	

❖	MzC (see page 1206)	
❖	N (see page 1206)	
❖	NdN (see page 1206)	

#### I.4.10.2.1 CaseName

##### C++

```
HRESULT get_CaseName(BSTR* );
HRESULT put_CaseName(BSTR);
```

##### C#

```
public String CaseName { get; set; }
```

##### Visual Basic

```
Public CaseName As String
```

##### Version

Available since version 11.

#### I.4.10.2.2 CaseType

##### C++

```
HRESULT get_CaseType(IRConcrColumnLoadCaseType* );
HRESULT put_CaseType(IRConcrColumnLoadCaseType);
```

##### C#

```
public IRConcrColumnLoadCaseType CaseType { get; set; }
```

##### Visual Basic

```
Public CaseType As IRConcrColumnLoadCaseType
```

##### Version

Available since version 11.

#### I.4.10.2.3 Fy

##### C++

```
HRESULT get_Fy(double* );
HRESULT put_Fy(double);
```

##### C#

```
public double Fy { get; set; }
```

##### Visual Basic

```
Public Fy As double
```

##### Version

Available since version 11.

#### I.4.10.2.4 Fz

##### C++

```
HRESULT get_Fz(double* );
HRESULT put_Fz(double);
```

##### C#

```
public double Fz { get; set; }
```

**Visual Basic**

```
Public Fz As double
```

**Version**

Available since version 11.

**I.4.10.2.5 Gamma****C++**

```
HRESULT get_Gamma(double*);  
HRESULT put_Gamma(double);
```

**C#**

```
public double Gamma { get; set; }
```

**Visual Basic**

```
Public Gamma As double
```

**Version**

Available since version 11.

**I.4.10.2.6 MnsY****C++**

```
HRESULT get_MnsY(double*);  
HRESULT put_MnsY(double);
```

**C#**

```
public double MnsY { get; set; }
```

**Visual Basic**

```
Public MnsY As double
```

**Version**

Available since version 11.

**I.4.10.2.7 MnsZ****C++**

```
HRESULT get_MnsZ(double*);  
HRESULT put_MnsZ(double);
```

**C#**

```
public double MnsZ { get; set; }
```

**Visual Basic**

```
Public MnsZ As double
```

**Version**

Available since version 11.

**I.4.10.2.8 MyA****C++**

```
HRESULT get_MyA(double*);  
HRESULT put_MyA(double);
```

**C#**

```
public double MyA { get; set; }
```

**Visual Basic**

```
Public MyA As Double
```

**Version**

Available since version 11.

**I.4.10.2.9 MyB****C++**

```
HRESULT get_MyB(double*);  
HRESULT put_MyB(double);
```

**C#**

```
public double MyB { get; set; }
```

**Visual Basic**

```
Public MyB As Double
```

**Version**

Available since version 11.

**I.4.10.2.10 MyC****C++**

```
HRESULT get_MyC(double*);  
HRESULT put_MyC(double);
```

**C#**

```
public double MyC { get; set; }
```

**Visual Basic**

```
Public MyC As Double
```

**Version**

Available since version 11.

**I.4.10.2.11 MzA****C++**

```
HRESULT get_MzA(double*);  
HRESULT put_MzA(double);
```

**C#**

```
public double MzA { get; set; }
```

**Visual Basic**

```
Public MzA As Double
```

**Version**

Available since version 11.

**I.4.10.2.12 MzB****C++**

```
HRESULT get_MzB(double*);
```

```
HRESULT put_MzB(double);
```

**C#**

```
public double MzB { get; set; }
```

**Visual Basic**

```
Public MzB As double
```

**Version**

Available since version 11.

**I.4.10.2.13 MzC****C++**

```
HRESULT get_MzC(double*);  
HRESULT put_MzC(double);
```

**C#**

```
public double MzC { get; set; }
```

**Visual Basic**

```
Public MzC As double
```

**Version**

Available since version 11.

**I.4.10.2.14 N****C++**

```
HRESULT get_N(double*);  
HRESULT put_N(double);
```

**C#**

```
public double N { get; set; }
```

**Visual Basic**

```
Public N As double
```

**Version**

Available since version 11.

**I.4.10.2.15 NdN****C++**

```
HRESULT get_NdN(double*);  
HRESULT put_NdN(double);
```

**C#**

```
public double NdN { get; set; }
```

**Visual Basic**

```
Public NdN As double
```

**Version**

Available since version 11.

## I.4.11 IRConcrColumnLoadCaseType

### C++

```
enum IRConcrColumnLoadCaseType;
```

### C#

```
public enum IRConcrColumnLoadCaseType;
```

### Visual Basic

```
Public Enum IRConcrColumnLoadCaseType
```

### Members

Members	Description
I_RCL_LOAD_CASE_PERMANENT = 1	Available since version 11.
I_RCL_LOAD_CASE_EXPLOITATION = 2	Available since version 11.
I_RCL_LOAD_CASE_ACCIDENTAL = 3	Available since version 11.
I_RCL_LOAD_CASE_WIND = 4	Available since version 11.
I_RCL_LOAD_CASE_DESIGN = 5	Available since version 11.
I_RCL_LOAD_CASE_DESIGN_ACC = 6	Available since version 11.
I_RCL_LOAD_CASE_TEMPERATURE = 7	Available since version 11.
I_RCL_LOAD_CASE_SNOW = 8	Available since version 11.
I_RCL_LOAD_CASE_TEST = 9	Available since version 11.
I_RCL_LOAD_CASE_SEISMIC = 10	Available since version 11.

### Version

Available since version 11.

## I.5 RC Slab

Available since version 10.

### Enumerations

	Name	Description
	IRConcrSlabRnfType (see page 1219)	
	IRConcrSlabRnfSegmentType (see page 1220)	
	IRConcrSlabSupportType (see page 1220)	

### Interfaces

	Name	Description
	IRConcrSlab (see page 1208)	
	IRConcrSlabCalculationOptions (see page 1213)	
	IRConcrSlabGeometry (see page 1214)	
	IRConcrSlabPatternOptions (see page 1215)	
	IRConcrSlabWizard (see page 1216)	

## I.5.1 IRConcrSlab

### Class Hierarchy

C++

```
interface IRConcrSlab : IDispatch;
```

C#

```
public interface IRConcrSlab;
```

### Visual Basic

```
Public Interface IRConcrSlab
```

### Version

Available since version 10.

### I.5.1.1 IRConcrSlab Members

The following tables list the members exposed by IRConcrSlab.

#### Public Fields

	Name	Description
◆	CalculationOptions (↗ see page 1209)	
◆	Concrete (↗ see page 1209)	
◆	Geometry (↗ see page 1209)	
◆	IsSelected (↗ see page 1210)	
◆	Name (↗ see page 1210)	
◆	PatternOptions (↗ see page 1210)	
◆	Reinforcement (↗ see page 1210)	
◆	Steel (↗ see page 1211)	
◆	UniqueId (↗ see page 1211)	

#### Public Methods

	Name	Description
≡▼	Activate (↗ see page 1211)	
≡▼	CreateCalculationNoteRtf (↗ see page 1212)	
≡▼	CreateFromObjects (↗ see page 1212)	
≡▼	Save (↗ see page 1212)	
≡▼	StartWizard (↗ see page 1213)	

### I.5.1.2 IRConcrSlab Fields

The fields of the IRConcrSlab class are listed here.

#### Public Fields

	Name	Description
◆	CalculationOptions (↗ see page 1209)	
◆	Concrete (↗ see page 1209)	
◆	Geometry (↗ see page 1209)	
◆	IsSelected (↗ see page 1210)	

◆	Name ( <a href="#">see page 1210</a> )	
◆	PatternOptions ( <a href="#">see page 1210</a> )	
◆	Reinforcement ( <a href="#">see page 1210</a> )	
◆	Steel ( <a href="#">see page 1211</a> )	
◆	Uniqueld ( <a href="#">see page 1211</a> )	

### I.5.1.2.1 CalculationOptions

#### C++

```
HRESULT get_CalculationOptions( IRConcrSlabCalculationOptions** );
```

#### C#

```
public IRConcrSlabCalculationOptions CalculationOptions { get; }
```

#### Visual Basic

```
Public ReadOnly CalculationOptions As IRConcrSlabCalculationOptions
```

#### Version

Available since version 10.

### I.5.1.2.2 Concrete

#### C++

```
HRESULT get_Concrete( IRConcrConcrete** );
```

#### C#

```
public IRConcrConcrete Concrete { get; }
```

#### Visual Basic

```
Public ReadOnly Concrete As IRConcrConcrete
```

#### Version

Available since version 10.

### I.5.1.2.3 Geometry

#### C++

```
HRESULT get_Geometry( IRConcrSlabGeometry** );
```

#### C#

```
public IRConcrSlabGeometry Geometry { get; }
```

#### Visual Basic

```
Public ReadOnly Geometry As IRConcrSlabGeometry
```

#### Version

Available since version 10.

### I.5.1.2.4 IsSelected

#### C++

```
HRESULT get_IsSelected(VARIANT_BOOL* );
HRESULT put_IsSelected(VARIANT_BOOL);
```

#### C#

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

**I.5.1.2.5 Name****C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

**I.5.1.2.6 PatternOptions****C++**

```
HRESULT get_PatternOptions(IRConcrSlabPatternOptions**);
```

**C#**

```
public IRConcrSlabPatternOptions PatternOptions { get; }
```

**Visual Basic**

```
Public ReadOnly PatternOptions As IRConcrSlabPatternOptions
```

**Version**

Available since version 10.

**I.5.1.2.7 Reinforcement****C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement**);
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

**I.5.1.2.8 Steel****C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

**I.5.1.2.9 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Version**

Available since version 10.

**I.5.1.3 IRConcrSlab Methods**

The methods of the IRConcrSlab class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Activate ( <a href="#">see page 1211</a> )	
💡	CreateCalculationNoteRtf ( <a href="#">see page 1212</a> )	
💡	CreateFromObjects ( <a href="#">see page 1212</a> )	
💡	Save ( <a href="#">see page 1212</a> )	
💡	StartWizard ( <a href="#">see page 1213</a> )	

**I.5.1.3.1 Activate****C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

**I.5.1.3.2 CreateCalculationNoteRtf****C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

**I.5.1.3.3 CreateFromObjects****C++**

```
HRESULT CreateFromObjects(BSTR _selText, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromObjects(String _selText);
```

**Visual Basic**

```
Public Function CreateFromObjects(_selText As String) As Boolean
```

**Version**

Available since version 10.

**I.5.1.3.4 Save****C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.5.1.3.5 StartWizard****C++**

```
HRESULT StartWizard(IRConcrSlabWizard** ret);
```

**C#**

```
public IRConcrSlabWizard StartWizard();
```

**Visual Basic**

```
Public Function StartWizard() As IRConcrSlabWizard
```

**Version**

Available since version 10.

**I.5.2 IRConcrSlabCalculationOptions****Class Hierarchy****C++**

```
interface IRConcrSlabCalculationOptions : IDispatch;
```

**C#**

```
public interface IRConcrSlabCalculationOptions;
```

**Visual Basic**

```
Public Interface IRConcrSlabCalculationOptions
```

**Version**

Available since version 10.

**I.5.2.1 IRConcrSlabCalculationOptions Members**

The following tables list the members exposed by IRConcrSlabCalculationOptions.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Cover (see page 1213)	
◆	CoverLateral (see page 1214)	

**I.5.2.2 IRConcrSlabCalculationOptions Fields**

The fields of the IRConcrSlabCalculationOptions class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Cover (see page 1213)	
◆	CoverLateral (see page 1214)	

**I.5.2.2.1 Cover****C++**

```
HRESULT get_Cover(double* );
HRESULT put_Cover(double);
```

**C#**

```
public double Cover { get; set; }
```

**Visual Basic**

```
Public Cover As double
```

**Version**

Available since version 10.

**I.5.2.2.2 CoverLateral****C++**

```
HRESULT get_CoverLateral(double* );
HRESULT put_CoverLateral(double);
```

**C#**

```
public double CoverLateral { get; set; }
```

**Visual Basic**

```
Public CoverLateral As double
```

**Version**

Available since version 10.

## I.5.3 IRConcrSlabGeometry

### Class Hierarchy

#### C++

```
interface IRConcrSlabGeometry : IDispatch;
```

#### C#

```
public interface IRConcrSlabGeometry;
```

### Visual Basic

```
Public Interface IRConcrSlabGeometry
```

### Version

Available since version 10.

### I.5.3.1 IRConcrSlabGeometry Members

The following tables list the members exposed by IRConcrSlabGeometry.

#### Public Fields

	Name	Description
◆	ConcreteVolume (see page 1215)	
◆	ShutteringArea (see page 1215)	

### I.5.3.2 IRConcrSlabGeometry Fields

The fields of the IRConcrSlabGeometry class are listed here.

#### Public Fields

	Name	Description
◆	ConcreteVolume (see page 1215)	
◆	ShutteringArea (see page 1215)	

### I.5.3.2.1 ConcreteVolume

#### C++

```
HRESULT get_ConcreteVolume(double*);
```

#### C#

```
public double ConcreteVolume { get; }
```

### Visual Basic

```
Public ReadOnly ConcreteVolume As Double
```

### Version

Available since version 10.

### I.5.3.2.2 ShutteringArea

#### C++

```
HRESULT get_ShutteringArea(double*);
```

#### C#

```
public double ShutteringArea { get; }
```

## Visual Basic

```
Public ReadOnly ShutteringArea As double
```

### Version

Available since version 10.

## I.5.4 IRConcrSlabPatternOptions

### Class Hierarchy

#### C++

```
interface IRConcrSlabPatternOptions : IDispatch;
```

#### C#

```
public interface IRConcrSlabPatternOptions;
```

## Visual Basic

```
Public Interface IRConcrSlabPatternOptions
```

### Version

Available since version 10.

## I.5.4.1 IRConcrSlabPatternOptions Members

The following tables list the members exposed by IRConcrSlabPatternOptions.

### Public Fields

	Name	Description
◆	ReinforcementSegment (see page 1216)	
◆	ReinforcementType (see page 1216)	

## I.5.4.2 IRConcrSlabPatternOptions Fields

The fields of the IRConcrSlabPatternOptions class are listed here.

### Public Fields

	Name	Description
◆	ReinforcementSegment (see page 1216)	
◆	ReinforcementType (see page 1216)	

## I.5.4.2.1 ReinforcementSegment

#### C++

```
HRESULT get_ReinforcementSegment( IRConcrSlabRnfSegmentType* );
HRESULT put_ReinforcementSegment( IRConcrSlabRnfSegmentType );
```

#### C#

```
public IRConcrSlabRnfSegmentType ReinforcementSegment { get; set; }
```

## Visual Basic

```
Public ReinforcementSegment As IRConcrSlabRnfSegmentType
```

**Version**

Available since version 10.

**I.5.4.2.2 ReinforcementType****C++**

```
HRESULT get_ReinforcementType( IRConcrSlabRnfType* );
HRESULT put_ReinforcementType( IRConcrSlabRnfType );
```

**C#**

```
public IRConcrSlabRnfType ReinforcementType { get; set; }
```

**Visual Basic**

```
Public ReinforcementType As IRConcrSlabRnfType
```

**Version**

Available since version 10.

**I.5.5 IRConcrSlabWizard****Class Hierarchy****C++**

```
interface IRConcrSlabWizard : IDispatch;
```

**C#**

```
public interface IRConcrSlabWizard;
```

**Visual Basic**

```
Public Interface IRConcrSlabWizard
```

**Version**

Available since version 10.

**I.5.5.1 IRConcrSlabWizard Members**

The following tables list the members exposed by IRConcrSlabWizard.

**Public Fields**

	Name	Description
◆	Thickness ( [ see page 1217 )	

**Public Methods**

	Name	Description
◆	AddColumnSupport ( [ see page 1218 )	
◆	AddOpening ( [ see page 1218 )	
◆	AddOpeningNode ( [ see page 1218 )	
◆	AddSlabNode ( [ see page 1219 )	
◆	AddWallSupport ( [ see page 1219 )	
◆	Finish ( [ see page 1219 )	

**I.5.5.2 IRConcrSlabWizard Fields**

The fields of the IRConcrSlabWizard class are listed here.

## Public Fields

	Name	Description
◆	Thickness ( <a href="#">see page 1217</a> )	

### I.5.5.2.1 Thickness

#### C++

```
HRESULT get_Thickness(double* );
HRESULT put_Thickness(double);
```

#### C#

```
public double Thickness { get; set; }
```

#### Visual Basic

```
Public Thickness As double
```

#### Version

Available since version 10.

### I.5.5.3 IRConcrSlabWizard Methods

The methods of the IRConcrSlabWizard class are listed here.

#### Public Methods

	Name	Description
◆	AddColumnSupport ( <a href="#">see page 1218</a> )	
◆	AddOpening ( <a href="#">see page 1218</a> )	
◆	AddOpeningNode ( <a href="#">see page 1218</a> )	
◆	AddSlabNode ( <a href="#">see page 1219</a> )	
◆	AddWallSupport ( <a href="#">see page 1219</a> )	
◆	Finish ( <a href="#">see page 1219</a> )	

### I.5.5.3.1 AddColumnSupport

#### C++

```
HRESULT AddColumnSupport(double _x, double _y, double _b, double _h, double _alpha);
```

#### C#

```
public void AddColumnSupport(double _x, double _y, double _b, double _h, double _alpha);
```

#### Visual Basic

```
Public Sub AddColumnSupport(_x As double, _y As double, _b As double, _h As double, _alpha As double)
```

#### Version

Available since version 10.

### I.5.5.3.2 AddOpening

#### C++

```
HRESULT AddOpening();
```

**C#**

```
public void AddOpening();
```

**Visual Basic**

```
Public Sub AddOpening()
```

**Version**

Available since version 10.

**I.5.5.3.3 AddOpeningNode****C++**

```
HRESULT AddOpeningNode(double _x, double _y);
```

**C#**

```
public void AddOpeningNode(double _x, double _y);
```

**Visual Basic**

```
Public Sub AddOpeningNode(_x As Double, _y As Double)
```

**Version**

Available since version 10.

**I.5.5.3.4 AddSlabNode****C++**

```
HRESULT AddSlabNode(double _x, double _y);
```

**C#**

```
public void AddSlabNode(double _x, double _y);
```

**Visual Basic**

```
Public Sub AddSlabNode(_x As Double, _y As Double)
```

**Version**

Available since version 10.

**I.5.5.3.5 AddWallSupport****C++**

```
HRESULT AddWallSupport(double _x1, double _y1, double _x2, double _y2, double _thickness,  
IRConcrSlabSupportType _type);
```

**C#**

```
public void AddWallSupport(double _x1, double _y1, double _x2, double _y2, double  
_thickness, IRConcrSlabSupportType _type);
```

**Visual Basic**

```
Public Sub AddWallSupport(_x1 As Double, _y1 As Double, _x2 As Double, _y2 As Double,  
_thickness As Double, _type As IRConcrSlabSupportType)
```

**Version**

Available since version 10.

### I.5.5.3.6 Finish

**C++**

```
HRESULT Finish(VARIANT_BOOL* ret);
```

**C#**

```
public bool Finish();
```

**Visual Basic**

```
Public Function Finish() As Boolean
```

**Version**

Available since version 10.

### I.5.6 IRConcrSlabRnfType

**C++**

```
enum IRConcrSlabRnfType;
```

**C#**

```
public enum IRConcrSlabRnfType;
```

**Visual Basic**

```
Public Enum IRConcrSlabRnfType
```

**Members**

Members	Description
I_CSR_SLAB_RNF_BAR_ONLY = 1	Available since version 10.
I_CSR_SLAB_RNF_NET_ONLY = 2	Available since version 10.
I_CSR_SLAB_RNF_BAR_AND_NET = 3	Available since version 10.

**Version**

Available since version 10.

### I.5.7 IRConcrSlabRnfSegmentType

**C++**

```
enum IRConcrSlabRnfSegmentType;
```

**C#**

```
public enum IRConcrSlabRnfSegmentType;
```

**Visual Basic**

```
Public Enum IRConcrSlabRnfSegmentType
```

**Members**

Members	Description
I_SRS_SLAB_SEG_SINGLE = 1	Available since version 10.
I_SRS_SLAB_SEG_ENTIRE = 2	Available since version 10.

**Version**

Available since version 10.

## I.5.8 IRConcrSlabSupportType

### C++

```
enum IRConcrSlabSupportType;
```

### C#

```
public enum IRConcrSlabSupportType;
```

### Visual Basic

```
Public Enum IRConcrSlabSupportType
```

### Members

Members	Description
I_CSS_SLAB_SUPPORT_PINNED = 1	Available since version 10.
I_CSS_SLAB_SUPPORT_FIXED = 2	Available since version 10.

### Version

Available since version 10.

## I.6 RC Deep Beam

Available since version 10.

### Interfaces

	Name	Description
↪	IRConcrDeepBeam (see page 1221)	
↪	IRConcrDeepBeamGeometry (see page 1225)	

## I.6.1 IRConcrDeepBeam

### Class Hierarchy

### C++

```
interface IRConcrDeepBeam : IDispatch;
```

### C#

```
public interface IRConcrDeepBeam;
```

### Visual Basic

```
Public Interface IRConcrDeepBeam
```

### Version

Available since version 10.

### I.6.1.1 IRConcrDeepBeam Members

The following tables list the members exposed by IRConcrDeepBeam.

### Public Fields

	Name	Description
❖	Concrete (see page 1222)	
❖	Geometry (see page 1222)	

❖	IsActive ( [ see page 1222) )	
❖	IsSelected ( [ see page 1223) )	
❖	Name ( [ see page 1223) )	
❖	Reinforcement ( [ see page 1223) )	
❖	Steel ( [ see page 1223) )	
❖	Uniqueld ( [ see page 1224) )	

## Public Methods

	Name	Description
☞	Activate ( [ see page 1224) )	
☞	CreateCalculationNoteRtf ( [ see page 1224) )	
☞	CreateFromObjects ( [ see page 1225) )	
☞	Save ( [ see page 1225) )	

### I.6.1.2 IRConcrDeepBeam Fields

The fields of the IRConcrDeepBeam class are listed here.

## Public Fields

	Name	Description
❖	Concrete ( [ see page 1222) )	
❖	Geometry ( [ see page 1222) )	
❖	IsActive ( [ see page 1222) )	
❖	IsSelected ( [ see page 1223) )	
❖	Name ( [ see page 1223) )	
❖	Reinforcement ( [ see page 1223) )	
❖	Steel ( [ see page 1223) )	
❖	Uniqueld ( [ see page 1224) )	

#### I.6.1.2.1 Concrete

##### C++

```
HRESULT get_Concrete(IRConcrConcrete**);
```

##### C#

```
public IRConcrConcrete Concrete { get; }
```

##### Visual Basic

```
Public ReadOnly Concrete As IRConcrConcrete
```

##### Version

Available since version 10.

#### I.6.1.2.2 Geometry

##### C++

```
HRESULT get_Geometry(IRConcrDeepBeamGeometry**);
```

##### C#

```
public IRConcrDeepBeamGeometry Geometry { get; }
```

##### Visual Basic

```
Public ReadOnly Geometry As IRConcrDeepBeamGeometry
```

**Version**

Available since version 10.

**I.6.1.2.3 IsActive****C++**

```
HRESULT get_IsActive(VARIANT_BOOL* );
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Version**

Available since version 10.

**I.6.1.2.4 IsSelected****C++**

```
HRESULT get_IsSelected(VARIANT_BOOL* );
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

**I.6.1.2.5 Name****C++**

```
HRESULT get_Name(BSTR* );
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

**I.6.1.2.6 Reinforcement****C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement** );
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

**I.6.1.2.7 Steel****C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

**I.6.1.2.8 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Version**

Available since version 10.

**I.6.1.3 IRConcrDeepBeam Methods**

The methods of the IRConcrDeepBeam class are listed here.

**Public Methods**

	Name	Description
💡	Activate (see page 1224)	
💡	CreateCalculationNoteRtf (see page 1224)	
💡	CreateFromObjects (see page 1225)	
💡	Save (see page 1225)	

**I.6.1.3.1 Activate****C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

**I.6.1.3.2 CreateCalculationNoteRtf****C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

**I.6.1.3.3 CreateFromObjects****C++**

```
HRESULT CreateFromObjects(BSTR _selText, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromObjects(String _selText);
```

**Visual Basic**

```
Public Function CreateFromObjects(_selText As String) As Boolean
```

**Version**

Available since version 10.

**I.6.1.3.4 Save****C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

**I.6.2 IRConcrDeepBeamGeometry****Class Hierarchy****C++**

```
interface IRConcrDeepBeamGeometry : IDispatch;
```

**C#**

```
public interface IRConcrDeepBeamGeometry;
```

## Visual Basic

```
Public Interface IRConcrDeepBeamGeometry
```

## Version

Available since version 10.

# I.7 RC Wall

Available since version 10.

## Interfaces

	Name	Description
↳	IRConcrWall (see page 1226)	
↳	IRConcrWallGeometry (see page 1230)	
↳	IRConcrWallOpening (see page 1234)	
↳	IRConcrWallOpenings (see page 1236)	

## I.7.1 IRConcrWall

### Class Hierarchy

#### C++

```
interface IRConcrWall : IDispatch;
```

#### C#

```
public interface IRConcrWall;
```

## Visual Basic

```
Public Interface IRConcrWall
```

## Version

Available since version 10.

### I.7.1.1 IRConcrWall Members

The following tables list the members exposed by IRConcrWall.

#### Public Fields

	Name	Description
◆	Concrete (see page 1227)	
◆	Geometry (see page 1227)	
◆	IsSelected (see page 1227)	
◆	Name (see page 1228)	
◆	Reinforcement (see page 1228)	
◆	Steel (see page 1228)	
◆	UniqueId (see page 1228)	

#### Public Methods

	Name	Description
◆	Activate (see page 1229)	

	Calculate (see page 1229)	
	CreateCalculationNoteRtf (see page 1229)	
	CreateFromObjects (see page 1230)	
	Save (see page 1230)	
	Verify (see page 1230)	

### I.7.1.2 IRConcrWall Fields

The fields of the IRConcrWall class are listed here.

#### Public Fields

	Name	Description
	Concrete (see page 1227)	
	Geometry (see page 1227)	
	IsSelected (see page 1227)	
	Name (see page 1228)	
	Reinforcement (see page 1228)	
	Steel (see page 1228)	
	UniqueId (see page 1228)	

#### I.7.1.2.1 Concrete

##### C++

```
HRESULT get_Concrete(IRConcrConcrete**);
```

##### C#

```
public IRConcrConcrete Concrete { get; }
```

##### Visual Basic

```
Public ReadOnly Concrete As IRConcrConcrete
```

#### Version

Available since version 10.

#### I.7.1.2.2 Geometry

##### C++

```
HRESULT get_Geometry(IRConcrWallGeometry**);
```

##### C#

```
public IRConcrWallGeometry Geometry { get; }
```

##### Visual Basic

```
Public ReadOnly Geometry As IRConcrWallGeometry
```

#### Version

Available since version 10.

#### I.7.1.2.3 IsSelected

##### C++

```
HRESULT get_IsSelected(VARIANT_BOOL*);  
HRESULT put_IsSelected(VARIANT_BOOL);
```

**C#**

```
public bool IsSelected { get; set; }
```

**Visual Basic**

```
Public IsSelected As Boolean
```

**Version**

Available since version 12.

**I.7.1.2.4 Name****C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Version**

Available since version 10.

**I.7.1.2.5 Reinforcement****C++**

```
HRESULT get_Reinforcement(IRConcrReinforcement**);
```

**C#**

```
public IRConcrReinforcement Reinforcement { get; }
```

**Visual Basic**

```
Public ReadOnly Reinforcement As IRConcrReinforcement
```

**Version**

Available since version 10.

**I.7.1.2.6 Steel****C++**

```
HRESULT get_Steel(IRConcrSteel**);
```

**C#**

```
public IRConcrSteel Steel { get; }
```

**Visual Basic**

```
Public ReadOnly Steel As IRConcrSteel
```

**Version**

Available since version 10.

**I.7.1.2.7 UniqueId****C++**

```
HRESULT get_UniqueId(long*);
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As Long
```

**Version**

Available since version 10.

**I.7.1.3 IRConcrWall Methods**

The methods of the IRConcrWall class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Activate (🔗 see page 1229)	
💡	Calculate (🔗 see page 1229)	
💡	CreateCalculationNoteRtf (🔗 see page 1229)	
💡	CreateFromObjects (🔗 see page 1230)	
💡	Save (🔗 see page 1230)	
💡	Verify (🔗 see page 1230)	

**I.7.1.3.1 Activate****C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Version**

Available since version 10.

**I.7.1.3.2 Calculate****C++**

```
HRESULT Calculate(VARIANT_BOOL _createReinforcement, VARIANT_BOOL* ret);
```

**C#**

```
public bool Calculate(bool _createReinforcement);
```

**Visual Basic**

```
Public Function Calculate(_createReinforcement As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.7.1.3.3 CreateCalculationNoteRtf

**C++**

```
HRESULT CreateCalculationNoteRtf(BSTR _fileName, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCalculationNoteRtf(String _fileName);
```

**Visual Basic**

```
Public Function CreateCalculationNoteRtf(_fileName As String) As Boolean
```

**Version**

Available since version 10.

### I.7.1.3.4 CreateFromObjects

**C++**

```
HRESULT CreateFromObjects(BSTR _selTaxt, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateFromObjects(String _selTaxt);
```

**Visual Basic**

```
Public Function CreateFromObjects(_selTaxt As String) As Boolean
```

**Version**

Available since version 10.

### I.7.1.3.5 Save

**C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Version**

Available since version 10.

### I.7.1.3.6 Verify

**C++**

```
HRESULT Verify(VARIANT_BOOL _showErrors, long* ret);
```

**C#**

```
public long Verify(bool _showErrors);
```

**Visual Basic**

```
Public Function Verify(_showErrors As Boolean) As long
```

## Version

Available since version 10.

## I.7.2 IRConcrWallGeometry

### Class Hierarchy

#### C++

```
interface IRConcrWallGeometry : IDispatch;
```

#### C#

```
public interface IRConcrWallGeometry;
```

### Visual Basic

```
Public Interface IRConcrWallGeometry
```

## Version

Available since version 10.

### I.7.2.1 IRConcrWallGeometry Members

The following tables list the members exposed by IRConcrWallGeometry.

#### Public Fields

	Name	Description
◆	Height (see page 1231)	
◆	LeftBoundaryLength (see page 1232)	
◆	LeftBoundaryThickness (see page 1232)	
◆	Length (see page 1232)	
◆	Name (see page 1233)	
◆	Openings (see page 1233)	
◆	RightBoundaryLength (see page 1233)	
◆	RightBoundaryThickness (see page 1233)	
◆	Thickness (see page 1234)	

### I.7.2.2 IRConcrWallGeometry Fields

The fields of the IRConcrWallGeometry class are listed here.

#### Public Fields

	Name	Description
◆	Height (see page 1231)	
◆	LeftBoundaryLength (see page 1232)	
◆	LeftBoundaryThickness (see page 1232)	
◆	Length (see page 1232)	
◆	Name (see page 1233)	
◆	Openings (see page 1233)	
◆	RightBoundaryLength (see page 1233)	

◆	RightBoundaryThickness (see page 1233)	
◆	Thickness (see page 1234)	

### I.7.2.2.1 Height

#### C++

```
HRESULT get_Height(double* );
HRESULT put_Height(double);
```

#### C#

```
public double Height { get; set; }
```

#### Visual Basic

```
Public Height As Double
```

#### Version

Available since version 10.

### I.7.2.2.2 LeftBoundaryLength

#### C++

```
HRESULT get_LeftBoundaryLength(double* );
HRESULT put_LeftBoundaryLength(double);
```

#### C#

```
public double LeftBoundaryLength { get; set; }
```

#### Visual Basic

```
Public LeftBoundaryLength As Double
```

#### Version

Available since version 10.

### I.7.2.2.3 LeftBoundaryThickness

#### C++

```
HRESULT get_LeftBoundaryThickness(double* );
HRESULT put_LeftBoundaryThickness(double);
```

#### C#

```
public double LeftBoundaryThickness { get; set; }
```

#### Visual Basic

```
Public LeftBoundaryThickness As Double
```

#### Version

Available since version 10.

### I.7.2.2.4 Length

#### C++

```
HRESULT get_Length(double* );
HRESULT put_Length(double);
```

#### C#

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Version**

Available since version 10.

**I.7.2.2.5 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Version**

Available since version 10.

**I.7.2.2.6 Openings****C++**

```
HRESULT get_Openings(IRConcrWallOpenings** );
```

**C#**

```
public IRConcrWallOpenings Openings { get; }
```

**Visual Basic**

```
Public ReadOnly Openings As IRConcrWallOpenings
```

**Version**

Available since version 10.

**I.7.2.2.7 RightBoundaryLength****C++**

```
HRESULT get_RightBoundaryLength(double* );
HRESULT put_RightBoundaryLength(double);
```

**C#**

```
public double RightBoundaryLength { get; set; }
```

**Visual Basic**

```
Public RightBoundaryLength As double
```

**Version**

Available since version 10.

**I.7.2.2.8 RightBoundaryThickness****C++**

```
HRESULT get_RightBoundaryThickness(double* );
HRESULT put_RightBoundaryThickness(double);
```

**C#**

```
public double RightBoundaryThickness { get; set; }
```

**Visual Basic**

```
Public RightBoundaryThickness As Double
```

**Version**

Available since version 10.

**I.7.2.2.9 Thickness****C++**

```
HRESULT get_Thickness(double* );
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thickness As Double
```

**Version**

Available since version 10.

**I.7.3 IRConcrWallOpening****Class Hierarchy****C++**

```
interface IRConcrWallOpening : IDispatch;
```

**C#**

```
public interface IRConcrWallOpening;
```

**Visual Basic**

```
Public Interface IRConcrWallOpening
```

**Version**

Available since version 10.

**I.7.3.1 IRConcrWallOpening Members**

The following tables list the members exposed by IRConcrWallOpening.

**Public Fields**

	Name	Description
◆	Lx (↗ see page 1235)	
◆	Lz (↗ see page 1235)	
◆	Name (↗ see page 1235)	
◆	PosX (↗ see page 1235)	
◆	PosZ (↗ see page 1236)	

**I.7.3.2 IRConcrWallOpening Fields**

The fields of the IRConcrWallOpening class are listed here.

## Public Fields

	Name	Description
◆	Lx ( <a href="#">see page 1235</a> )	
◆	Lz ( <a href="#">see page 1235</a> )	
◆	Name ( <a href="#">see page 1235</a> )	
◆	PosX ( <a href="#">see page 1235</a> )	
◆	PosZ ( <a href="#">see page 1236</a> )	

### I.7.3.2.1 Lx

#### C++

```
HRESULT get_Lx(double* );
HRESULT put_Lx(double);
```

#### C#

```
public double Lx { get; set; }
```

#### Visual Basic

```
Public Lx As double
```

#### Version

Available since version 10.

### I.7.3.2.2 Lz

#### C++

```
HRESULT get_Lz(double* );
HRESULT put_Lz(double);
```

#### C#

```
public double Lz { get; set; }
```

#### Visual Basic

```
Public Lz As double
```

#### Version

Available since version 10.

### I.7.3.2.3 Name

#### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Version

Available since version 10.

### I.7.3.2.4 PosX

#### C++

```
HRESULT get_PosX(double* );
HRESULT put_PosX(double);
```

#### C#

```
public double PosX { get; set; }
```

#### Visual Basic

```
Public PosX As double
```

#### Version

Available since version 10.

### I.7.3.2.5 PosZ

#### C++

```
HRESULT get_PosZ(double* );
HRESULT put_PosZ(double);
```

#### C#

```
public double PosZ { get; set; }
```

#### Visual Basic

```
Public PosZ As double
```

#### Version

Available since version 10.

## I.7.4 IRConcrWallOpenings

#### Class Hierarchy

#### C++

```
interface IRConcrWallOpenings : IDispatch;
```

#### C#

```
public interface IRConcrWallOpenings;
```

#### Visual Basic

```
Public Interface IRConcrWallOpenings
```

#### Version

Available since version 10.

### I.7.4.1 IRConcrWallOpenings Members

The following tables list the members exposed by IRConcrWallOpenings.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1237</a> )	
◆	Item ( <a href="#">see page 1237</a> )	

## Public Methods

	Name	Description
>Add (see page 1237)	Add (see page 1237)	

### I.7.4.2 IRConcrWallOpenings Fields

The fields of the IRConcrWallOpenings class are listed here.

#### Public Fields

	Name	Description
Count	Count (see page 1237)	
Item	Item (see page 1237)	

#### I.7.4.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As long
```

##### Version

Available since version 10.

#### I.7.4.2.2 Item

##### C++

```
HRESULT get_Item(IRConcrWallOpening**);
```

##### C#

```
public IRConcrWallOpening Item { get; }
```

##### Visual Basic

```
Public ReadOnly Item As IRConcrWallOpening
```

##### Version

Available since version 10.

### I.7.4.3 IRConcrWallOpenings Methods

The methods of the IRConcrWallOpenings class are listed here.

#### Public Methods

	Name	Description
Add	Add (see page 1237)	

#### I.7.4.3.1 Add

##### C++

```
HRESULT Add(IRConcrWallOpening** ret);
```

##### C#

```
public IRConcrWallOpening Add();
```

## Visual Basic

```
Public Function Add() As IRConcrWallOpening
```

### Version

Available since version 10.

## I.8 Reinforcement

Available since version 10.

### Enumerations

	Name	Description
	IRConcrBarType (see page 1250)	
	IRConcrBarSubtype (see page 1250)	
	IRConcrSteelType (see page 1252)	
	IRConcrSteelSurfaceType (see page 1256)	

### Interfaces

	Name	Description
	IRConcrReinforcement (see page 1238)	
	IRConcrReinforcingBar (see page 1242)	
	IRConcrReinforcingBars (see page 1248)	
	IRConcrSteelNames (see page 1251)	
	IRConcrSteel (see page 1253)	
	IRConcrConcrete (see page 1255)	
	IRConcrSteelStrengths (see page 1257)	
	IRConcrBarDiameters (see page 1257)	

## I.8.1 IRConcrReinforcement

### Class Hierarchy

#### C++

```
interface IRConcrReinforcement : IDispatch;
```

#### C#

```
public interface IRConcrReinforcement;
```

## Visual Basic

```
Public Interface IRConcrReinforcement
```

### Version

Available since version 10.

### I.8.1.1 IRConcrReinforcement Members

The following tables list the members exposed by IRConcrReinforcement.

#### Public Fields

	Name	Description
◆	Bars ( [ see page 1239 )	
◆	Comment ( [ see page 1239 )	
◆	GroupGlobally ( [ see page 1240 )	
◆	GroupingMethod ( [ see page 1240 )	
◆	IsFrozen ( [ see page 1240 )	

#### Public Methods

	Name	Description
◆	Freeze ( [ see page 1241 )	
◆	LockRefresh ( [ see page 1241 )	
◆	RemoveAll ( [ see page 1241 )	
◆	RemoveParametric ( [ see page 1242 )	

### I.8.1.2 IRConcrReinforcement Fields

The fields of the IRConcrReinforcement class are listed here.

#### Public Fields

	Name	Description
◆	Bars ( [ see page 1239 )	
◆	Comment ( [ see page 1239 )	
◆	GroupGlobally ( [ see page 1240 )	
◆	GroupingMethod ( [ see page 1240 )	
◆	IsFrozen ( [ see page 1240 )	

#### I.8.1.2.1 Bars

##### C++

```
HRESULT get_Bars( IRConcrReinforcingBars** );
```

##### C#

```
public IRConcrReinforcingBars Bars { get; }
```

##### Visual Basic

```
Public ReadOnly Bars As IRConcrReinforcingBars
```

##### Version

Available since version 10.

#### I.8.1.2.2 Comment

##### C++

```
HRESULT get_Comment(BSTR*);
HRESULT put_Comment(BSTR);
```

##### C#

```
public String Comment { get; set; }
```

**Visual Basic**

```
Public Comment As String
```

**Version**

Available since version 10.

**I.8.1.2.3 GroupGlobally****C++**

```
HRESULT get_GroupGlobally(VARIANT_BOOL* );
HRESULT put_GroupGlobally(VARIANT_BOOL);
```

**C#**

```
public bool GroupGlobally { get; set; }
```

**Visual Basic**

```
Public GroupGlobally As Boolean
```

**Version**

Available since version 10.

**I.8.1.2.4 GroupingMethod****C++**

```
HRESULT get_GroupingMethod(long* );
HRESULT put_GroupingMethod(long);
```

**C#**

```
public long GroupingMethod { get; set; }
```

**Visual Basic**

```
Public GroupingMethod As long
```

**Version**

Available since version 10.

**I.8.1.2.5 IsFrozen****C++**

```
HRESULT get_IsFrozen(VARIANT_BOOL* );
```

**C#**

```
public bool IsFrozen { get; }
```

**Visual Basic**

```
Public ReadOnly IsFrozen As Boolean
```

**Version**

Available since version 11.

**I.8.1.3 IRConcrReinforcement Methods**

The methods of the IRConcrReinforcement class are listed here.

**Public Methods**

	Name	Description
	Freeze (see page 1241)	

	LockRefresh (see page 1241)	
	RemoveAll (see page 1241)	
	RemoveParametric (see page 1242)	

### I.8.1.3.1 Freeze

**C++**

```
HRESULT Freeze(VARIANT_BOOL _freeze, VARIANT_BOOL* ret);
```

**C#**

```
public bool Freeze(bool _freeze);
```

**Visual Basic**

```
Public Function Freeze(_freeze As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.8.1.3.2 LockRefresh

**C++**

```
HRESULT LockRefresh(VARIANT_BOOL _lock, VARIANT_BOOL* ret);
```

**C#**

```
public bool LockRefresh(bool _lock);
```

**Visual Basic**

```
Public Function LockRefresh(_lock As Boolean) As Boolean
```

**Version**

Available since version 10.

### I.8.1.3.3 RemoveAll

**C++**

```
HRESULT RemoveAll();
```

**C#**

```
public void RemoveAll();
```

**Visual Basic**

```
Public Sub RemoveAll()
```

**Version**

Available since version 10.

### I.8.1.3.4 RemoveParametric

**C++**

```
HRESULT RemoveParametric();
```

**C#**

```
public void RemoveParametric();
```

**Visual Basic**

```
Public Sub RemoveParametric()
```

**Version**

Available since version 11.

**I.8.2 IRConcrReinforcingBar****Class Hierarchy****C++**

```
interface IRConcrReinforcingBar : IDispatch;
```

**C#**

```
public interface IRConcrReinforcingBar;
```

**Visual Basic**

```
Public Interface IRConcrReinforcingBar
```

**Version**

Available since version 10.

**I.8.2.1 IRConcrReinforcingBar Members**

The following tables list the members exposed by IRConcrReinforcingBar.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DirectionX (↗ see page 1243)	
◆	DirectionY (↗ see page 1244)	
◆	DirectionZ (↗ see page 1244)	
◆	GroupNumber (↗ see page 1244)	
◆	IsParametric (↗ see page 1244)	
◆	Position (↗ see page 1245)	
◆	ROBarEditorId (↗ see page 1245)	
◆	Span (↗ see page 1245)	
◆	SteelType (↗ see page 1245)	
◆	Subtype (↗ see page 1246)	
◆	Type (↗ see page 1246)	
◆	Weight (↗ see page 1246)	

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡◆	CreateCopy (↗ see page 1247)	
≡◆	GetROBarEditor (↗ see page 1247)	
≡◆	SetBarShape (↗ see page 1247)	
≡◆	SetDirections (↗ see page 1247)	
≡◆	SetROBarEditor (↗ see page 1248)	

**I.8.2.2 IRConcrReinforcingBar Fields**

The fields of the IRConcrReinforcingBar class are listed here.

## Public Fields

	Name	Description
◆	DirectionX ( [ see page 1243) )	
◆	DirectionY ( [ see page 1244) )	
◆	DirectionZ ( [ see page 1244) )	
◆	GroupNumber ( [ see page 1244) )	
◆	IsParametric ( [ see page 1244) )	
◆	Position ( [ see page 1245) )	
◆	ROBarEditorId ( [ see page 1245) )	
◆	Span ( [ see page 1245) )	
◆	SteelType ( [ see page 1245) )	
◆	Subtype ( [ see page 1246) )	
◆	Type ( [ see page 1246) )	
◆	Weight ( [ see page 1246) )	

### I.8.2.2.1 DirectionX

#### C++

```
HRESULT get_DirectionX(IRobotGeoPoint3D**);
```

#### C#

```
public IRobotGeoPoint3D DirectionX { get; }
```

#### Visual Basic

```
Public ReadOnly DirectionX As IRobotGeoPoint3D
```

#### Version

Available since version 10.

### I.8.2.2.2 DirectionY

#### C++

```
HRESULT get_DirectionY(IRobotGeoPoint3D**);
```

#### C#

```
public IRobotGeoPoint3D DirectionY { get; }
```

#### Visual Basic

```
Public ReadOnly DirectionY As IRobotGeoPoint3D
```

#### Version

Available since version 10.

### I.8.2.2.3 DirectionZ

#### C++

```
HRESULT get_DirectionZ(IRobotGeoPoint3D**);
```

#### C#

```
public IRobotGeoPoint3D DirectionZ { get; }
```

#### Visual Basic

```
Public ReadOnly DirectionZ As IRobotGeoPoint3D
```

**Version**

Available since version 10.

**I.8.2.2.4 GroupNumber****C++**

```
HRESULT get_GroupNumber( int* );
HRESULT put_GroupNumber( int );
```

**C#**

```
public int GroupNumber { get; set; }
```

**Visual Basic**

```
Public GroupNumber As Integer
```

**Version**

Available since version 10.

**I.8.2.2.5 IsParametric****C++**

```
HRESULT get_IsParametric(VARIANT_BOOL* );
HRESULT put_IsParametric(VARIANT_BOOL );
```

**C#**

```
public bool IsParametric { get; set; }
```

**Visual Basic**

```
Public IsParametric As Boolean
```

**Version**

Available since version 11.

**I.8.2.2.6 Position****C++**

```
HRESULT get_Position(IRobotGeoPoint3D** );
```

**C#**

```
public IRobotGeoPoint3D Position { get; }
```

**Visual Basic**

```
Public ReadOnly Position As IRobotGeoPoint3D
```

**Version**

Available since version 10.

**I.8.2.2.7 ROBarEditorId****C++**

```
HRESULT get_ROBarEditorId(long* );
```

**C#**

```
public long ROBarEditorId { get; }
```

**Visual Basic**

```
Public ReadOnly ROBarEditorId As Long
```

**Version**

Available since version 10.

**I.8.2.2.8 Span****C++**

```
HRESULT get_Span(short* );
HRESULT put_Span(short);
```

**C#**

```
public short Span { get; set; }
```

**Visual Basic**

```
Public Span As short
```

**Version**

Available since version 10.

**I.8.2.2.9 SteelType****C++**

```
HRESULT get_SteelType(IRConcrSteelType* );
HRESULT put_SteelType(IRConcrSteelType);
```

**C#**

```
public IRConcrSteelType SteelType { get; set; }
```

**Visual Basic**

```
Public SteelType As IRConcrSteelType
```

**Version**

Available since version 10.

**I.8.2.2.10 Subtype****C++**

```
HRESULT get_Subtype(IRConcrBarSubtype* );
```

**C#**

```
public IRConcrBarSubtype Subtype { get; }
```

**Visual Basic**

```
Public ReadOnly Subtype As IRConcrBarSubtype
```

**Version**

Available since version 10.

**I.8.2.2.11 Type****C++**

```
HRESULT get_Type(IRConcrBarType* );
```

**C#**

```
public IRConcrBarType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRConcrBarType
```

**Version**

Available since version 10.

**I.8.2.2.12 Weight****C++**

```
HRESULT get_Weight( double* );
```

**C#**

```
public double Weight { get; }
```

**Visual Basic**

```
Public ReadOnly Weight As Double
```

**Version**

Available since version 10.

**I.8.2.3 IRConcrReinforcingBar Methods**

The methods of the IRConcrReinforcingBar class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	CreateCopy ( <a href="#">see page 1247</a> )	
≡	GetROBarEditor ( <a href="#">see page 1247</a> )	
≡	SetBarShape ( <a href="#">see page 1247</a> )	
≡	SetDirections ( <a href="#">see page 1247</a> )	
≡	SetROBarEditor ( <a href="#">see page 1248</a> )	

**I.8.2.3.1 CreateCopy****C++**

```
HRESULT CreateCopy(double _offX, double _offY, double _offZ, long _number, VARIANT_BOOL* ret);
```

**C#**

```
public bool CreateCopy(double _offX, double _offY, double _offZ, long _number);
```

**Visual Basic**

```
Public Function CreateCopy(_offX As Double, _offY As Double, _offZ As Double, _number As Long) As Boolean
```

**Version**

Available since version 10.

**I.8.2.3.2 GetROBarEditor****C++**

```
HRESULT GetROBarEditor(VARIANT_BOOL _readOnly, IDispatch** ret);
```

**C#**

```
public IDispatch* GetROBarEditor(bool _readOnly);
```

**Visual Basic**

```
Public Function GetROBarEditor(_readOnly As Boolean) As IDispatch*
```

**Version**

Available since version 10.

**I.8.2.3.3 SetBarShape****C++**

```
HRESULT SetBarShape(VARIANT_BOOL* ret);
```

**C#**

```
public bool SetBarShape();
```

**Visual Basic**

```
Public Function SetBarShape() As Boolean
```

**Version**

Available since version 10.

**I.8.2.3.4 SetDirections****C++**

```
HRESULT SetDirections(double _Xx, double _Xy, double _Xz, double _Yx, double _Yy, double  
_Yz, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetDirections(double _Xx, double _Xy, double _Xz, double _Yx, double _Yy,  
double _Yz);
```

**Visual Basic**

```
Public Function SetDirections(_Xx As double, _Xy As double, _Xz As double, _Yx As double,  
_Yy As double, _Yz As double) As Boolean
```

**Version**

Available since version 10.

**I.8.2.3.5 SetROBarEditor****C++**

```
HRESULT SetROBarEditor(IDispatch* _editor);
```

**C#**

```
public void SetROBarEditor(IDispatch* _editor);
```

**Visual Basic**

```
Public Sub SetROBarEditor(ByRef _editor As IDispatch*)
```

**Version**

Available since version 10.

**I.8.3 IRConcrReinforcingBars****Class Hierarchy**

**C++**

```
interface IRConcrReinforcingBars : IDispatch;
```

**C#**

```
public interface IRConcrReinforcingBars;
```

**Visual Basic**

```
Public Interface IRConcrReinforcingBars
```

**Version**

Available since version 10.

**I.8.3.1 IRConcrReinforcingBars Members**

The following tables list the members exposed by IRConcrReinforcingBars.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1249</a> )	
◆	Item ( <a href="#">see page 1249</a> )	

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add ( <a href="#">see page 1249</a> )	

**I.8.3.2 IRConcrReinforcingBars Fields**

The fields of the IRConcrReinforcingBars class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1249</a> )	
◆	Item ( <a href="#">see page 1249</a> )	

**I.8.3.2.1 Count****C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Version**

Available since version 10.

**I.8.3.2.2 Item****C++**

```
HRESULT get_Item( IRConcrReinforcingBar** );
```

**C#**

```
public IRConcrReinforcingBar Item { get; }
```

## Visual Basic

```
Public ReadOnly Item As IRConcrReinforcingBar
```

### Version

Available since version 10.

## I.8.3.3 IRConcrReinforcingBars Methods

The methods of the IRConcrReinforcingBars class are listed here.

### Public Methods

	Name	Description
	Add (see page 1249)	

### I.8.3.3.1 Add

#### C++

```
HRESULT Add(BSTR _shapeCode, IRConcrBarType _type, IRConcrBarSubType _subType, double _diameter, double _bendDiam = 0, IRConcrReinforcingBar** ret);
```

#### C#

```
public IRConcrReinforcingBar Add(String _shapeCode, IRConcrBarType _type, IRConcrBarSubType _subType, double _diameter, double _bendDiam = 0);
```

## Visual Basic

```
Public Function Add(_shapeCode As String, _type As IRConcrBarType, _subType As IRConcrBarSubType, _diameter As Double, Optional _bendDiam As Double = 0) As IRConcrReinforcingBar
```

### Version

Available since version 10.

## I.8.4 IRConcrBarType

#### C++

```
enum IRConcrBarType;
```

#### C#

```
public enum IRConcrBarType;
```

## Visual Basic

```
Public Enum IRConcrBarType
```

### Members

Members	Description
I_CBT_BAR_TYPE_UNDEFINED = 0	Available since version 10.
I_CBT_BAR_TYPE_LON_MAIN = 1	Available since version 10.
I_CBT_BAR_TYPE_TRA_MAIN = 2	Available since version 10.
I_CBT_BAR_TYPE_LON_HOLE = 4	Available since version 10.
I_CBT_BAR_TYPE_TRA_HOLE = 8	Available since version 10.
I_CBT_BAR_TYPE_LON_SURF = 16	Available since version 10.
I_CBT_BAR_TYPE_TRA_SURF = 32	Available since version 10.
I_CBT_BAR_TYPE_LON_CONS = 64	Available since version 10.
I_CBT_BAR_TYPE_TRA_CONS = 128	Available since version 10.

I_CBT_BAR_TYPE_LON_CORN = 256	Available since version 10.
I_CBT_BAR_TYPE_TRA_CORN = 512	Available since version 10.
I_CBT_BAR_TYPE_LON_JOIN = 1024	Available since version 10.
I_CBT_BAR_TYPE_TRA_JOIN = 2048	Available since version 10.
I_CBT_BAR_TYPE_LON_SPEC = 4096	Available since version 10.
I_CBT_BAR_TYPE_TRA_SPEC = 8192	Available since version 10.
I_CBT_BAR_TYPE_LON_DIST = 32768	Available since version 10.
I_CBT_BAR_TYPE_LON_BOAT = 65536	Available since version 10.
I_CBT_BAR_TYPE_LON_TORS = 131072	Available since version 10.
I_CBT_BAR_TYPE_LON_MAIN_FIRST = 262144	Available since version 10.
I_CBT_BAR_TYPE_LON_JOIN_STRUT = 524288	Available since version 10.
I_CBT_BAR_TYPE_LON_FLAN = 1048576	Available since version 10.

**Version**

Available since version 10.

**I.8.5 IRConcrBarSubtype****C++**

```
enum IRConcrBarSubtype;
```

**C#**

```
public enum IRConcrBarSubtype;
```

**Visual Basic**

```
Public Enum IRConcrBarSubtype
```

**Members**

Members	Description
I_CBS_BAR_SUBTYPE_ANY = 0	Available since version 10.
I_CBS_BAR_SUBTYPE_UP = 1	Available since version 10.
I_CBS_BAR_SUBTYPE_UP_RIGHT = 2	Available since version 10.
I_CBS_BAR_SUBTYPE_DOWN = 4	Available since version 10.
I_CBS_BAR_SUBTYPE_INC_LEFT = 8	Available since version 10.
I_CBS_BAR_SUBTYPE_INC_RIGHT = 16	Available since version 10.
I_CBS_BAR_SUBTYPE_STRAIGHT = 32	Available since version 10.
I_CBS_BAR_SUBTYPE_INC_45 = 64	Available since version 10.
I_CBS_BAR_SUBTYPE_DIR_VER = 128	Available since version 10.
I_CBS_BAR_SUBTYPE_DIR_HOR = 256	Available since version 10.
I_CBS_BAR_SUBTYPE_EXT = 512	Available since version 10.
I_CBS_BAR_SUBTYPE_INT = 1024	Available since version 10.
I_CBS_BAR_SUBTYPE_SHA = 2048	Available since version 10.
I_CBS_BAR_SUBTYPE_PIN = 4096	Available since version 10.
I_CBS_BAR_SUBTYPE_JOIN = 8192	Available since version 10.
I_CBS_BAR_SUBTYPE_SPEC = 16384	Available since version 10.
I_CBS_BAR_SUBTYPE_UBAR = 32768	Available since version 10.
I_CBS_BAR_SUBTYPE_DIST = 65536	Available since version 10.
I_CBS_BAR_SUBTYPE_WALL = 131072	Available since version 10.
I_CBS_BAR_SUBTYPE_FOOT = 262144	Available since version 10.
I_CBS_BAR_SUBTYPE_TABLE = 524288	Available since version 10.

I_CBS_BAR_SUBTYPE_GROIN = 1048576	Available since version 10.
I_CBS_BAR_SUBTYPE_FRONT = 2097152	Available since version 10.
I_CBS_BAR_SUBTYPE_BACK = 4194304	Available since version 10.
I_CBS_BAR_SUBTYPE_LEFT = 8388608	Available since version 10.
I_CBS_BAR_SUBTYPE_RIGHT = 16777216	Available since version 10.

**Version**

Available since version 10.

## I.8.6 IRConcrSteelNames

**Class Hierarchy****C++**

```
interface IRConcrSteelNames : IDispatch;
```

**C#**

```
public interface IRConcrSteelNames;
```

**Visual Basic**

```
Public Interface IRConcrSteelNames
```

**Version**

Available since version 10.

### I.8.6.1 IRConcrSteelNames Members

The following tables list the members exposed by IRConcrSteelNames.

**Public Fields**

	Name	Description
❖	Count (↗ see page 1252)	
❖	Item (↗ see page 1252)	

### I.8.6.2 IRConcrSteelNames Fields

The fields of the IRConcrSteelNames class are listed here.

**Public Fields**

	Name	Description
❖	Count (↗ see page 1252)	
❖	Item (↗ see page 1252)	

#### I.8.6.2.1 Count

**C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Version**

Available since version 10.

### I.8.6.2.2 Item

**C++**

```
HRESULT get_Item(BSTR*);
```

**C#**

```
public String Item { get; }
```

**Visual Basic**

```
Public ReadOnly Item As String
```

**Version**

Available since version 10.

## I.8.7 IRConcrSteelType

**C++**

```
enum IRConcrSteelType;
```

**C#**

```
public enum IRConcrSteelType;
```

**Visual Basic**

```
Public Enum IRConcrSteelType
```

**Members**

Members	Description
I_CST_STEEL_TYPE_LONG = 1	Available since version 10.
I_CST_STEEL_TYPE_TRAN = 2	Available since version 10.
I_CST_STEEL_TYPE_WIRE = 3	Available since version 10.
I_CST_STEEL_TYPE_SPEC = 4	Available since version 10.

**Version**

Available since version 10.

## I.8.8 IRConcrSteel

**Class Hierarchy**

**C++**

```
interface IRConcrSteel : IDispatch;
```

**C#**

```
public interface IRConcrSteel;
```

**Visual Basic**

```
Public Interface IRConcrSteel
```

**Version**

Available since version 10.

### I.8.8.1 IRConcrSteel Members

The following tables list the members exposed by IRConcrSteel.

## Public Fields

	Name	Description
◆	AvailableNames ( [ see page 1254) ]	
◆	AvailableStrengths ( [ see page 1254) ]	
◆	BarDiameters ( [ see page 1254) ]	
◆	BarsDatabasePath ( [ see page 1255) ]	Available since version 10.
◆	CharacteristicStrength ( [ see page 1255) ]	
◆	SelectedName ( [ see page 1255) ]	
◆	SurfaceType ( [ see page 1255) ]	

### I.8.8.2 IRConcrSteel Fields

The fields of the IRConcrSteel class are listed here.

## Public Fields

	Name	Description
◆	AvailableNames ( [ see page 1254) ]	
◆	AvailableStrengths ( [ see page 1254) ]	
◆	BarDiameters ( [ see page 1254) ]	
◆	BarsDatabasePath ( [ see page 1255) ]	Available since version 10.
◆	CharacteristicStrength ( [ see page 1255) ]	
◆	SelectedName ( [ see page 1255) ]	
◆	SurfaceType ( [ see page 1255) ]	

#### I.8.8.2.1 AvailableNames

##### C++

```
HRESULT get_AvailableNames( IRConcrSteelNames** );
```

##### C#

```
public IRConcrSteelNames AvailableNames { get; }
```

##### Visual Basic

```
Public ReadOnly AvailableNames As IRConcrSteelNames
```

##### Version

Available since version 10.

#### I.8.8.2.2 AvailableStrengths

##### C++

```
HRESULT get_AvailableStrengths( IRConcrSteelStrengths** );
```

##### C#

```
public IRConcrSteelStrengths AvailableStrengths { get; }
```

##### Visual Basic

```
Public ReadOnly AvailableStrengths As IRConcrSteelStrengths
```

**Version**

Available since version 10.

**I.8.8.2.3 BarDiameters****C++**

```
HRESULT get_BarDowners(IRConcrBarDiameters**);
```

**C#**

```
public IRConcrBarDiameters BarDiameters { get; }
```

**Visual Basic**

```
Public ReadOnly BarDiameters As IRConcrBarDiameters
```

**Version**

Available since version 10.

**I.8.8.2.4 BarsDatabasePath****C#**

```
void BarsDatabasePath;
```

**Description**

Available since version 10.

**I.8.8.2.5 CharacteristicStrength****C++**

```
HRESULT get_CharacteristicStrength(double*);  
HRESULT put_CharacteristicStrength(double);
```

**C#**

```
public double CharacteristicStrength { get; set; }
```

**Visual Basic**

```
Public CharacteristicStrength As Double
```

**Version**

Available since version 10.

**I.8.8.2.6 SelectedName****C++**

```
HRESULT get_SelectedName(BSTR*);  
HRESULT put_SelectedName(BSTR);
```

**C#**

```
public String SelectedName { get; set; }
```

**Visual Basic**

```
Public SelectedName As String
```

**Version**

Available since version 10.

### I.8.8.2.7 SurfaceType

#### C++

```
HRESULT get_SurfaceType(IRConcrSteelSurfaceType* );
HRESULT put_SurfaceType(IRConcrSteelSurfaceType);
```

#### C#

```
public IRConcrSteelSurfaceType SurfaceType { get; set; }
```

#### Visual Basic

```
Public SurfaceType As IRConcrSteelSurfaceType
```

#### Version

Available since version 10.

## I.8.9 IRConcrConcrete

#### Class Hierarchy

#### C++

```
interface IRConcrConcrete : IDispatch;
```

#### C#

```
public interface IRConcrConcrete;
```

#### Visual Basic

```
Public Interface IRConcrConcrete
```

#### Version

Available since version 10.

### I.8.9.1 IRConcrConcrete Members

The following tables list the members exposed by IRConcrConcrete.

#### Public Fields

	Name	Description
◆	CharacteristicStrength (see page 1256)	

### I.8.9.2 IRConcrConcrete Fields

The fields of the IRConcrConcrete class are listed here.

#### Public Fields

	Name	Description
◆	CharacteristicStrength (see page 1256)	

### I.8.9.2.1 CharacteristicStrength

#### C++

```
HRESULT get_CharacteristicStrength(double* );
HRESULT put_CharacteristicStrength(double);
```

#### C#

```
public double CharacteristicStrength { get; set; }
```

**Visual Basic**

```
Public CharacteristicStrength As double
```

**Version**

Available since version 10.

**I.8.10 IRConcrSteelSurfaceType****C++**

```
enum IRConcrSteelSurfaceType;
```

**C#**

```
public enum IRConcrSteelSurfaceType;
```

**Visual Basic**

```
Public Enum IRConcrSteelSurfaceType
```

**Members**

Members	Description
I_SST_SURFACE_RIBBED = 1	Available since version 10.
I_SST_SURFACE_PLAIN = 2	Available since version 10.

**Version**

Available since version 10.

**I.8.11 IRConcrSteelStrengths****Class Hierarchy****C++**

```
interface IRConcrSteelStrengths : IDispatch;
```

**C#**

```
public interface IRConcrSteelStrengths;
```

**Visual Basic**

```
Public Interface IRConcrSteelStrengths
```

**Version**

Available since version 10.

**I.8.12 IRConcrBarDiameters****Class Hierarchy****C++**

```
interface IRConcrBarDiameters : IDispatch;
```

**C#**

```
public interface IRConcrBarDiameters;
```

**Visual Basic**

```
Public Interface IRConcrBarDiameters
```

## Version

Available since version 10.

### I.8.12.1 IRConcrBarDiameters Members

The following tables list the members exposed by IRConcrBarDiameters.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1258</a> )	
◆	Item ( <a href="#">see page 1258</a> )	

#### Public Methods

	Name	Description
◆	Add ( <a href="#">see page 1258</a> )	
◆	RemoveAll ( <a href="#">see page 1259</a> )	

### I.8.12.2 IRConcrBarDiameters Fields

The fields of the IRConcrBarDiameters class are listed here.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1258</a> )	
◆	Item ( <a href="#">see page 1258</a> )	

### I.8.12.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Version

Available since version 10.

### I.8.12.2.2 Item

#### C++

```
HRESULT get_Item(double* );
```

#### C#

```
public double Item { get; }
```

#### Visual Basic

```
Public ReadOnly Item As double
```

#### Version

Available since version 10.

### I.8.12.3 IRConcrBarDiameters Methods

The methods of the IRConcrBarDiameters class are listed here.

#### Public Methods

	Name	Description
+	Add ( [ see page 1258 )	
+	RemoveAll ( [ see page 1259 )	

#### I.8.12.3.1 Add

##### C++

```
HRESULT Add(double _newDiam);
```

##### C#

```
public void Add(double _newDiam);
```

##### Visual Basic

```
Public Sub Add(_newDiam As double)
```

#### Version

Available since version 10.

#### I.8.12.3.2 RemoveAll

##### C++

```
HRESULT RemoveAll();
```

##### C#

```
public void RemoveAll();
```

##### Visual Basic

```
Public Sub RemoveAll()
```

#### Version

Available since version 10.

## I.9 Drawing

Available since version 10.

#### Interfaces

	Name	Description
+	IRConcrDrawing ([ see page 1259 )	
+	IRConcrDrawingBarGroup ([ see page 1260 )	

### I.9.1 IRConcrDrawing

#### Class Hierarchy

##### C++

```
interface IRConcrDrawing : IDispatch;
```

**C#**

```
public interface IRConcrDrawing;
```

**Visual Basic**

```
Public Interface IRConcrDrawing
```

**Version**

Available since version 10.

**I.9.1.1 IRConcrDrawing Members**

The following tables list the members exposed by IRConcrDrawing.

**Public Fields**

	Name	Description
◆	Group ( <a href="#">see page 1260</a> )	
◆	GroupCount ( <a href="#">see page 1260</a> )	

**I.9.1.2 IRConcrDrawing Fields**

The fields of the IRConcrDrawing class are listed here.

**Public Fields**

	Name	Description
◆	Group ( <a href="#">see page 1260</a> )	
◆	GroupCount ( <a href="#">see page 1260</a> )	

**I.9.1.2.1 Group****C++**

```
HRESULT get_Group( IRConcrDrawingBarGroup** );
```

**C#**

```
public IRConcrDrawingBarGroup Group { get; }
```

**Visual Basic**

```
Public ReadOnly Group As IRConcrDrawingBarGroup
```

**Version**

Available since version 10.

**I.9.1.2.2 GroupCount****C++**

```
HRESULT get_GroupCount( long* );
```

**C#**

```
public long GroupCount { get; }
```

**Visual Basic**

```
Public ReadOnly GroupCount As long
```

**Version**

Available since version 10.

## I.9.2 IRConcrDrawingBarGroup

### Class Hierarchy

#### C++

```
interface IRConcrDrawingBarGroup : IDispatch;
```

#### C#

```
public interface IRConcrDrawingBarGroup;
```

### Visual Basic

```
Public Interface IRConcrDrawingBarGroup
```

### Version

Available since version 10.

## I.9.2.1 IRConcrDrawingBarGroup Members

The following tables list the members exposed by IRConcrDrawingBarGroup.

### Public Fields

	Name	Description
◆	Bar (see page 1261)	
◆	BarCount (see page 1261)	

## I.9.2.2 IRConcrDrawingBarGroup Fields

The fields of the IRConcrDrawingBarGroup class are listed here.

### Public Fields

	Name	Description
◆	Bar (see page 1261)	
◆	BarCount (see page 1261)	

## I.9.2.2.1 Bar

#### C++

```
HRESULT get_Bar(IRConcrReinforcingBar**);
```

#### C#

```
public IRConcrReinforcingBar Bar { get; }
```

### Visual Basic

```
Public ReadOnly Bar As IRConcrReinforcingBar
```

### Version

Available since version 10.

## I.9.2.2.2 BarCount

#### C++

```
HRESULT get_BarCount(long*);
```

#### C#

```
public long BarCount { get; }
```

**Visual Basic**

```
Public ReadOnly BarCount As long
```

**Version**

Available since version 10.

**I.10 IRobotProjectComponentType****C++**

```
enum IRobotProjectComponentType;
```

**C#**

```
public enum IRobotProjectComponentType;
```

**Visual Basic**

```
Public Enum IRobotProjectComponentType
```

**Members**

Members	Description
I_PCT_BEAM = 3	Concrete beam .
I_PCT_COLUMN = 4	Concrete column .
I_PCT_FOOT = 5	Concrete footing .
I_PCT_JOINT = 6	Steel connection .
I_PCT_DRAWING = 7	Plotter drawing .
I_PCT_BWALL = 8	Concrete wall .
I_PCT_RETAINING_WALL = 10	Retaining wall. Available since version 3.
I_PCT_SLAB = 9	Available since version 3.
I_PCT_CONTINUOUS_FOOT = 11	Available since version 3.

**Description**

A set of identifiers is defined to describe different types of components that may be saved in a project. .

**I.11 IRobotProjectComponent****Class Hierarchy****C++**

```
interface IRobotProjectComponent : IDispatch;
```

**C#**

```
public interface IRobotProjectComponent;
```

**Visual Basic**

```
Public Interface IRobotProjectComponent
```

**Description**

Common interface granting access to any project component. .

**I.11.1 IRobotProjectComponent Members**

The following tables list the members exposed by IRobotProjectComponent.

## Public Fields

	Name	Description
◆	Data (see page 1263)	Object providing access to data on a project component; the type of the object depends on the component type .
◆	Name (see page 1263)	Component name .
◆	Type (see page 1263)	Project component type .

### I.11.2 IRobotProjectComponent Fields

The fields of the IRobotProjectComponent class are listed here.

## Public Fields

	Name	Description
◆	Data (see page 1263)	Object providing access to data on a project component; the type of the object depends on the component type .
◆	Name (see page 1263)	Component name .
◆	Type (see page 1263)	Project component type .

#### I.11.2.1 Data

##### C++

```
HRESULT get_Data(IDispatch* );
HRESULT put_Data(IDispatch);
```

##### C#

```
public IDispatch Data { get; set; }
```

##### Visual Basic

```
Public Data As IDispatch
```

##### Description

Object providing access to data on a project component; the type of the object depends on the component type .

#### I.11.2.2 Name

##### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

##### C#

```
public String Name { get; set; }
```

##### Visual Basic

```
Public Name As String
```

##### Description

Component name .

#### I.11.2.3 Type

##### C++

```
HRESULT get_Type(IRobotProjectComponentType* );
HRESULT put_Type(IRobotProjectComponentType);
```

##### C#

```
public IRobotProjectComponentType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotProjectComponentType
```

**Description**

Project component type .

**I.12 IRobotProjectComponentMngr****Class Hierarchy****C++**

```
interface IRobotProjectComponentMngr : IDispatch;
```

**C#**

```
public interface IRobotProjectComponentMngr;
```

**Visual Basic**

```
Public Interface IRobotProjectComponentMngr
```

**Description**

Manager of project components manages tree-like structures containing all components defined in a project. Project components are divided into different levels. The manager makes available a list of all levels that are indexed with numbers from 1 to the number of levels. Additionally, each level has its name. Project components may be taken by levels. It is possible to take from the project all the components of an indicated type from the level of the indicated name. .

**I.12.1 IRobotProjectComponentMngr Members**

The following tables list the members exposed by IRobotProjectComponentMngr.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	LevelCount ( <a href="#">see page 1264</a> )	Number of levels in the structure containing all project components. .
◆	StdLevelName ( <a href="#">see page 1265</a> )	Name of standard level Available since version 2.0.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Create ( <a href="#">see page 1265</a> )	Function creates a project component of the defined type and locate it on a specified level. Available since version 2.0.
◆	Get ( <a href="#">see page 1265</a> )	The function returns a collection of components of the indicated type from the level of the indicated name. Collection components are objects of the RobotProjectComponent type. .
◆	GetLevelName ( <a href="#">see page 1266</a> )	The function returns the name of the level with the indicated index. Levels are indexed with numbers from 1 to LevelCount ( <a href="#">see page 1264</a> ). .

**I.12.2 IRobotProjectComponentMngr Fields**

The fields of the IRobotProjectComponentMngr class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	LevelCount ( <a href="#">see page 1264</a> )	Number of levels in the structure containing all project components. .
◆	StdLevelName ( <a href="#">see page 1265</a> )	Name of standard level Available since version 2.0.

### I.12.2.1 LevelCount

**C++**

```
HRESULT get_LevelCount(long* );
HRESULT put_LevelCount(long);
```

**C#**

```
public long LevelCount { get; set; }
```

**Visual Basic**

```
Public LevelCount As long
```

**Description**

Number of levels in the structure containing all project components..

### I.12.2.2 StdLevelName

**C++**

```
HRESULT get_StdLevelName(BSTR* );
```

**C#**

```
public String StdLevelName { get; }
```

**Visual Basic**

```
Public ReadOnly StdLevelName As String
```

**Description**

Name of standard level Available since version 2.0.

## I.12.3 IRobotProjectComponentMngr Methods

The methods of the IRobotProjectComponentMngr class are listed here.

### Public Methods

	<b>Name</b>	<b>Description</b>
💡	Create (see page 1265)	Function creates a project component of the defined type and locate it on a specified level. Available since version 2.0.
💡	Get (see page 1265)	The function returns a collection of components of the indicated type from the level of the indicated name. Collection components are objects of the RobotProjectComponent type..
💡	GetLevelName (see page 1266)	The function returns the name of the level with the indicated index. Levels are indexed with numbers from 1 to LevelCount (see page 1264)..

### I.12.3.1 Create

**C++**

```
HRESULT Create(IRobotProjectComponentType _cmpnt_type, BSTR _cmpnt_name, BSTR _level_name,
IRobotProjectComponent** ret);
```

**C#**

```
public IRobotProjectComponent Create(IRobotProjectComponentType _cmpnt_type, String
_cmpnt_name, String _level_name);
```

**Visual Basic**

```
Public Function Create(_cmpnt_type As IRobotProjectComponentType, _cmpnt_name As String,
_level_name As String) As IRobotProjectComponent
```

## Description

Function creates a project component of the defined type and locate it on a specified level. Available since version 2.0.

### I.12.3.2 Get

#### C++

```
HRESULT Get(IRobotProjectComponentType _cmpnt_type, BSTR _level_name, IRobotCollection** ret);
```

#### C#

```
public IRobotCollection Get(IRobotProjectComponentType _cmpnt_type, String _level_name);
```

#### Visual Basic

```
Public Function Get(_cmpnt_type As IRobotProjectComponentType, _level_name As String) As IRobotCollection
```

#### Description

The function returns a collection of components of the indicated type from the level of the indicated name. Collection components are objects of the RobotProjectComponent type. .

### I.12.3.3 GetLevelName

#### C++

```
HRESULT GetLevelName(long _level_idx, BSTR* ret);
```

#### C#

```
public String GetLevelName(long _level_idx);
```

#### Visual Basic

```
Public Function GetLevelName(_level_idx As long) As String
```

#### Description

The function returns the name of the level with the indicated index. Levels are indexed with numbers from 1 to LevelCount (see page 1264). .

## II Support of external file formats

A structure and an entire project is saved to the file (RTD format). It is, however, possible to open and save structures defined in other external file formats. .

### Enumerations

	Name	Description
☝	IRobotExternalFileFormat (see page 1266)	A set of identifiers is defined to describe different external file formats accepted by Robot. .
☝	IRobotSTRParamType (see page 1273)	The Robot text file allows one to define variables of different types. Depending on variable type, they may assume values of different character. A set of identifiers is defined to refer to different types of variables available in STR files. .

### Interfaces

	Name	Description
☞	IRobotSTRParams (see page 1267)	Interface granting access to the set of parameters (variables) defined by the STR format files. .

	IRobotSTRParameter ( <a href="#">see page 1268</a> )	The interface describing the variable of text file in STR format. .
-----------------------------------------------------------------------------------	------------------------------------------------------	---------------------------------------------------------------------

## II.1 IRobotExternalFileFormat

### C++

```
enum IRobotExternalFileFormat;
```

### C#

```
public enum IRobotExternalFileFormat;
```

### Visual Basic

```
Public Enum IRobotExternalFileFormat
```

### Members

Members	Description
I_EFF_STR = 1	Text file STR format .
I_EFF_DXF = 2	DXF format .

### Description

A set of identifiers is defined to describe different external file formats accepted by Robot. .

## II.2 IRobotSTRParams

### Class Hierarchy

### C++

```
interface IRobotSTRParams : IDispatch;
```

### C#

```
public interface IRobotSTRParams;
```

### Visual Basic

```
Public Interface IRobotSTRParams
```

### Description

Interface granting access to the set of parameters (variables) defined by the STR format files. .

### II.2.1 IRobotSTRParams Members

The following tables list the members exposed by IRobotSTRParams.

#### Public Fields

	Name	Description
	Count ( <a href="#">see page 1267</a> )	Number of managed parameters .

#### Public Methods

	Name	Description
	FindParameter ( <a href="#">see page 1268</a> )	The function returns the index associated with the parameter of the indicated name .
	GetParameter ( <a href="#">see page 1268</a> )	The function returns a structure describing a parameter with the indicated index .

## II.2.2 IRobotSTRParams Fields

The fields of the IRobotSTRParams class are listed here.

### Public Fields

	Name	Description
◆	Count ( [ see page 1267 )	Number of managed parameters .

### II.2.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of managed parameters .

## II.2.3 IRobotSTRParams Methods

The methods of the IRobotSTRParams class are listed here.

### Public Methods

	Name	Description
◆	FindParameter ([ see page 1268 )	The function returns the index associated with the parameter of the indicated name .
◆	GetParameter ([ see page 1268 )	The function returns a structure describing a parameter with the indicated index .

### II.2.3.1 FindParameter

#### C++

```
HRESULT FindParameter(BSTR _param_name, long* ret);
```

#### C#

```
public long FindParameter(String _param_name);
```

#### Visual Basic

```
Public Function FindParameter(_param_name As String) As long
```

#### Description

The function returns the index associated with the parameter of the indicated name .

### II.2.3.2 GetParameter

#### C++

```
HRESULT GetParameter(long _idx, IRobotSTRParameter** ret);
```

#### C#

```
public IRobotSTRParameter GetParameter(long _idx);
```

## Visual Basic

```
Public Function GetParameter(_idx As long) As IRobotSTRParameter
```

### Description

The function returns a structure describing a parameter with the indicated index .

## II.3 IRobotSTRParameter

### Class Hierarchy

#### C++

```
interface IRobotSTRParameter : IDispatch;
```

#### C#

```
public interface IRobotSTRParameter;
```

## Visual Basic

```
Public Interface IRobotSTRParameter
```

### Description

The interface describing the variable of text file in STR format. .

### II.3.1 IRobotSTRParameter Members

The following tables list the members exposed by IRobotSTRParameter.

#### Public Fields

	Name	Description
◆	Description ( <a href="#">see page 1270</a> )	Variable description .
◆	DoubleVal_1 ( <a href="#">see page 1270</a> )	Value of the variable of the I_STR_PT_DOUBLE type. It is also the first out of three values of the variable of the I_STR_PT_DOUBLE_3 type. .
◆	DoubleVal_2 ( <a href="#">see page 1270</a> )	The second out of three variables of the I_STR_PT_DOUBLE_3 type .
◆	DoubleVal_3 ( <a href="#">see page 1271</a> )	The third out of three variables of the I_STR_PT_DOUBLE_3 type .
◆	FilePathVal ( <a href="#">see page 1271</a> )	Value of the variable of the I_STR_PT_GROUP type representing the access path to the file with the appropriate bitmap. .
◆	IntegerVal ( <a href="#">see page 1271</a> )	Value of the variable of the I_STR_PT_INTEGER type .
◆	IsActive ( <a href="#">see page 1271</a> )	Flag indicating if the variable is active .
◆	Name ( <a href="#">see page 1272</a> )	Variable name .
◆	SelectionVal ( <a href="#">see page 1272</a> )	Value of the variable of the I_STR_PT_SELECTION type describing a selection by means of text .
◆	TextList ( <a href="#">see page 1272</a> )	String of characters containing all possible values (separated by commas) of the variable of the I_STR_PT_TEXT_LIST type. .
◆	TextVal ( <a href="#">see page 1272</a> )	Value of the variable of the I_STR_PT_TEXT type. It is also the value currently selected from the list for the variable of the I_STR_PT_TEXT_LIST type. .
◆	Type ( <a href="#">see page 1273</a> )	Variable type .

### II.3.2 IRobotSTRParameter Fields

The fields of the IRobotSTRParameter class are listed here.

#### Public Fields

	Name	Description
◆	Description ( <a href="#">see page 1270</a> )	Variable description .

◆	DoubleVal_1 (see page 1270)	Value of the variable of the I_STR_PT_DOUBLE type. It is also the first out of three values of the variable of the I_STR_PT_DOUBLE_3 type. .
◆	DoubleVal_2 (see page 1270)	The second out of three variables of the I_STR_PT_DOUBLE_3 type .
◆	DoubleVal_3 (see page 1271)	The third out of three variables of the I_STR_PT_DOUBLE_3 type .
◆	FilePathVal (see page 1271)	Value of the variable of the I_STR_PT_GROUP type representing the access path to the file with the appropriate bitmap..
◆	IntegerVal (see page 1271)	Value of the variable of the I_STR_PT_INTEGER type .
◆	IsActive (see page 1271)	Flag indicating if the variable is active .
◆	Name (see page 1272)	Variable name .
◆	SelectionVal (see page 1272)	Value of the variable of the I_STR_PT_SELECTION type describing a selection by means of text .
◆	TextList (see page 1272)	String of characters containing all possible values (separated by commas) of the variable of the I_STR_PT_TEXT_LIST type. .
◆	TextVal (see page 1272)	Value of the variable of the I_STR_PT_TEXT type. It is also the value currently selected from the list for the variable of the I_STR_PT_TEXT_LIST type. .
◆	Type (see page 1273)	Variable type .

### II.3.2.1 Description

#### C++

```
HRESULT get_Description(BSTR*);
```

#### C#

```
public String Description { get; }
```

#### Visual Basic

```
Public ReadOnly Description As String
```

#### Description

Variable description .

### II.3.2.2 DoubleVal\_1

#### C++

```
HRESULT get_DoubleVal_1(double*);  
HRESULT put_DoubleVal_1(double);
```

#### C#

```
public double DoubleVal_1 { get; set; }
```

#### Visual Basic

```
Public DoubleVal_1 As double
```

#### Description

Value of the variable of the I\_STR\_PT\_DOUBLE type. It is also the first out of three values of the variable of the I\_STR\_PT\_DOUBLE\_3 type. .

### II.3.2.3 DoubleVal\_2

#### C++

```
HRESULT get_DoubleVal_2(double*);  
HRESULT put_DoubleVal_2(double);
```

#### C#

```
public double DoubleVal_2 { get; set; }
```

**Visual Basic**

```
Public DoubleVal_2 As double
```

**Description**

The second out of three variables of the I\_STR\_PT\_DOUBLE\_3 type .

**II.3.2.4 DoubleVal\_3****C++**

```
HRESULT get_DoubleVal_3(double* );
HRESULT put_DoubleVal_3(double);
```

**C#**

```
public double DoubleVal_3 { get; set; }
```

**Visual Basic**

```
Public DoubleVal_3 As double
```

**Description**

The third out of three variables of the I\_STR\_PT\_DOUBLE\_3 type .

**II.3.2.5 FilePathVal****C++**

```
HRESULT get_FilePathVal(BSTR* );
HRESULT put_FilePathVal(BSTR);
```

**C#**

```
public String FilePathVal { get; set; }
```

**Visual Basic**

```
Public FilePathVal As String
```

**Description**

Value of the variable of the I\_STR\_PT\_GROUP type representing the access path to the file with the appropriate bitmap. .

**II.3.2.6 IntegerVal****C++**

```
HRESULT get_IntegerVal(long* );
HRESULT put_IntegerVal(long);
```

**C#**

```
public long IntegerVal { get; set; }
```

**Visual Basic**

```
Public IntegerVal As long
```

**Description**

Value of the variable of the I\_STR\_PT\_INTEGER type .

**II.3.2.7 IsActive****C++**

```
HRESULT get_IsActive(VARIANT_BOOL* );
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Description**

Flag indicating if the variable is active .

**II.3.2.8 Name****C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Description**

Variable name .

**II.3.2.9 SelectionVal****C++**

```
HRESULT get_SelectionVal(BSTR*);  
HRESULT put_SelectionVal(BSTR);
```

**C#**

```
public String SelectionVal { get; set; }
```

**Visual Basic**

```
Public SelectionVal As String
```

**Description**

Value of the variable of the I\_STR\_PT\_SELECTION type describing a selection by means of text .

**II.3.2.10 TextList****C++**

```
HRESULT get_TextList(BSTR*);  
HRESULT put_TextList(BSTR);
```

**C#**

```
public String TextList { get; set; }
```

**Visual Basic**

```
Public TextList As String
```

**Description**

String of characters containing all possible values (separated by commas) of the variable of the I\_STR\_PT\_TEXT\_LIST type. .

### II.3.2.11 TextVal

**C++**

```
HRESULT get_TextVal(BSTR* );
HRESULT put_TextVal(BSTR);
```

**C#**

```
public String TextVal { get; set; }
```

**Visual Basic**

```
Public TextVal As String
```

**Description**

Value of the variable of the I\_STR\_PT\_TEXT type. It is also the value currently selected from the list for the variable of the I\_STR\_PT\_TEXT\_LIST type. .

### II.3.2.12 Type

**C++**

```
HRESULT get_Type(IRobotSTRParamType* );
```

**C#**

```
public IRobotSTRParamType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotSTRParamType
```

**Description**

Variable type .

## II.4 IRobotSTRParamType

**C++**

```
enum IRobotSTRParamType;
```

**C#**

```
public enum IRobotSTRParamType;
```

**Visual Basic**

```
Public Enum IRobotSTRParamType
```

**Members**

Members	Description
I_STR_PT_INTEGER = 1	Variable assuming the value in the form of an integer .
I_STR_PT_DOUBLE = 2	Variable assuming the value in the form of a real number .
I_STR_PT_TEXT = 3	Variable assuming the value in the form of a text .
I_STR_PT_SELECTION = 4	Variable assuming the value in the form of a text describing selection .
I_STR_PT_DOUBLE_3 = 5	Variable assuming the value described by means of three real numbers .
I_STR_PT_TEXT_LIST = 6	Variable assuming the value in the form of a specially formatted text. It consists of fragments separated by commas, defining the list of possible text values, and the (repeated) currently-selected fragment..
I_STR_PT_SECTION = 7	Variable containing the section name (as a text) .

I_STR_PT_GROUP = 8	Variable representing a group of variables. Its value consists of two elements: group name (stored in the TextVal variable), and the path to the file containing the bitmap associated with the group (FilePathVal) .
--------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Description

The Robot text file allows one to define variables of different types. Depending on variable type, they may assume values of different character. A set of identifiers is defined to refer to different types of variables available in STR files. .

# III Job preferences

## Enumerations

	Name	Description
	IRobotCodeType ( <a href="#">see page 1303</a> )	Set of identifiers determining the types of codes that may be set in the Robot program.
	IRobotDatabaseType ( <a href="#">see page 1317</a> )	Available database types.
	IRobotCalculationModelCoherence ( <a href="#">see page 1318</a> )	

## Interfaces

	Name	Description
	IRobotProjectPreferences ( <a href="#">see page 1288</a> )	Interface describing the settings for the current project (job).
	IRobotSectionDatabaseList ( <a href="#">see page 1297</a> )	The interface describing a list of section databases .
	IRobotMaterialDatabase ( <a href="#">see page 1300</a> )	The interface providing access to material database .
	IRobotSectionDatabase ( <a href="#">see page 1304</a> )	The interface providing access to section database.
	IRobotProjectPreferencesEvents ( <a href="#">see page 1307</a> )	Events related to the change of settings for the current project.
	IRobotEurocodeSteelDesignFactors ( <a href="#">see page 1308</a> )	Partial factors for the steel design according to Eurocode.
	IRobotEurocodeSteelConnectionFactors ( <a href="#">see page 1311</a> )	Partial factors for the steel connection design according to Eurocode.
	IRobotEurocodeFactors ( <a href="#">see page 1316</a> )	Partial factors for Eurocodes.

## III.1 Units and formats

Available since version 3.5.

## Enumerations

	Name	Description
	IRobotUnitType ( <a href="#">see page 1275</a> )	Unit types.
	IRobotUnitEditionType ( <a href="#">see page 1277</a> )	Types of editable units. .

## Interfaces

	Name	Description
	IRobotUnitData ( <a href="#">see page 1275</a> )	Unit definition interface.
	IRobotUnitMngr ( <a href="#">see page 1278</a> )	Interface of unit support management. .

	IRobotUnitComplexData (see page 1282)	Interface of composed unit definition. A composed unit is a unit being a function of other two units. For example, unit of the moment is a product of force unit and length unit. .
	IRobotUnitEditionServer (see page 1283)	Interface enabling unit edition. .
	IRobotUnitEditionData (see page 1286)	Interface of unit edit definition. An edit definition is a definition which determines a base unit and a proportionality factor. .
	IRobotUnitMngrEvents (see page 1287)	Events (notifications) generated by the unit manager.

### III.1.1 IRobotUnitType

#### C++

```
enum IRobotUnitType;
```

#### C#

```
public enum IRobotUnitType;
```

#### Visual Basic

```
Public Enum IRobotUnitType
```

#### Members

Members	Description
I_UT_STRUCTURE_DIMENSION = 1	Available since version 3.5.
I_UT_SECTION_DIMENSION = 2	Available since version 3.5.
I_UT_SECTION_PROPERTIES = 3	Available since version 3.5.
I_UT_STEEL_CONECTIONS = 4	Available since version 3.5.
I_UT_DIAMETER_RC_BASE = 5	Available since version 3.5.
I_UT_REINFORCEMENT_AREAS = 6	Available since version 3.5.
I_UT_FORCE = 7	Available since version 3.5.
I_UT_MOMENT = 8	Available since version 3.5.
I_UT_STRESS = 9	Available since version 3.5.
I_UT_DISPLACEMENT = 10	Available since version 3.5.
I_UT_ANGLE_ROTATION_DATA = 11	Available since version 3.5.
I_UT_ANGLE_ROTATION_RESULT = 12	Available since version 3.5.
I_UT_TEMPERATURE = 13	Available since version 3.5.
I_UT_WEIGHT = 14	Available since version 3.5.
I_UT_MASS = 15	Available since version 3.5.
I_UT_DIMENSIONLESS_QUALITY = 16	Available since version 3.5.
I_UT_RULER = 17	Available since version 3.5.

#### Description

Unit types.

#### Version

Available since version 3.5.

### III.1.2 IRobotUnitData

#### Class Hierarchy

#### C++

```
interface IRobotUnitData : IDispatch;
```

**C#**

```
public interface IRobotUnitData;
```

**Visual Basic**

```
Public Interface IRobotUnitData
```

**Description**

Unit definition interface.

**Version**

Available since version 3.5.

**III.1.2.1 IRobotUnitData Members**

The following tables list the members exposed by IRobotUnitData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	E ( <a href="#">see page 1276</a> )	Type ( <a href="#">see page 1277</a> ) of unit presentation: true value - exponential presentation, false value - decimal presentation .
◆	Name ( <a href="#">see page 1276</a> )	Unit name.
◆	Precision ( <a href="#">see page 1277</a> )	Precision of unit presentation - number of decimal places.
◆	Type ( <a href="#">see page 1277</a> )	Unit type.

**III.1.2.2 IRobotUnitData Fields**

The fields of the IRobotUnitData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	E ( <a href="#">see page 1276</a> )	Type ( <a href="#">see page 1277</a> ) of unit presentation: true value - exponential presentation, false value - decimal presentation .
◆	Name ( <a href="#">see page 1276</a> )	Unit name.
◆	Precision ( <a href="#">see page 1277</a> )	Precision of unit presentation - number of decimal places.
◆	Type ( <a href="#">see page 1277</a> )	Unit type.

**III.1.2.2.1 E****C++**

```
HRESULT get_E(VARIANT_BOOL* );
HRESULT put_E(VARIANT_BOOL);
```

**C#**

```
public bool E { get; set; }
```

**Visual Basic**

```
Public E As Boolean
```

**Description**

Type ([see page 1277](#)) of unit presentation: true value - exponential presentation, false value - decimal presentation .

**Version**

Available since version 3.5.

### III.1.2.2.2 Name

#### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

#### C#

```
public String Name { get; set; }
```

#### Visual Basic

```
Public Name As String
```

#### Description

Unit name.

#### Version

Available since version 3.5.

### III.1.2.2.3 Precision

#### C++

```
HRESULT get_Precision(long* );
HRESULT put_Precision(long);
```

#### C#

```
public long Precision { get; set; }
```

#### Visual Basic

```
Public Precision As long
```

#### Description

Precision of unit presentation - number of decimal places.

#### Version

Available since version 3.5.

### III.1.2.2.4 Type

#### C++

```
HRESULT get_Type(IRobotUnitType* );
```

#### C#

```
public IRobotUnitType Type { get; }
```

#### Visual Basic

```
Public ReadOnly Type As IRobotUnitType
```

#### Description

Unit type.

#### Version

Available since version 3.5.

## III.1.3 IRobotUnitEditionType

#### C++

```
enum IRobotUnitEditionType;
```

**C#**

```
public enum IRobotUnitEditionType;
```

**Visual Basic**

```
Public Enum IRobotUnitEditionType
```

**Members**

Members	Description
I_UMT_LENGTH = 0	Available since version 3.5.
I_UMT_FORCE = 1	Available since version 3.5.
I_UMT_MASS = 2	Available since version 3.5.

**Description**

Types of editable units. .

**Version**

Available since version 3.5.

**III.1.4 IRobotUnitMngr****Class Hierarchy****C++**

```
interface IRobotUnitMngr : IDispatch;
```

**C#**

```
public interface IRobotUnitMngr;
```

**Visual Basic**

```
Public Interface IRobotUnitMngr
```

**Description**

Interface of unit support management. .

**Version**

Available since version 3.5.

**III.1.4.1 IRobotUnitMngr Members**

The following tables list the members exposed by IRobotUnitMngr.

**Public Fields**

	Name	Description
!	UnitEdition ( [ see page 1279)	Edition server.
!	UseMetricAsDefault ( [ see page 1279)	Flag indicating if the software uses metric units by default.

**Public Methods**

	Name	Description
!	Count ( [ see page 1280)	Function returns the number of available units of the specified type. .
!	Get ( [ see page 1280)	Function takes definition of the active unit for a given type.
!	GetCoeff ( [ see page 1280)	The function returns the coefficient of conversion of the currently-selected unit of a given type to the base unit.

	GetCoeff2 (see page 1281)	The function returns the conversion factor for a unit of a given type and specified index to the base unit.
	GetName (see page 1281)	Function takes name of the unit of the specified type with the determined index. .
	Refresh (see page 1281)	Function forces refreshing current information about units in the whole application.
	Set (see page 1282)	Function determines active unit of the specified type based on the provided definition. .

### III.1.4.2 IRobotUnitMngr Fields

The fields of the IRobotUnitMngr class are listed here.

#### Public Fields

	Name	Description
	UnitEdition (see page 1279)	Edition server.
	UseMetricAsDefault (see page 1279)	Flag indicating if the software uses metric units by default.

#### III.1.4.2.1 UnitEdition

##### C++

```
HRESULT get_UnitEdition(IRobotUnitEditionServer**);
```

##### C#

```
public IRobotUnitEditionServer UnitEdition { get; }
```

##### Visual Basic

```
Public ReadOnly UnitEdition As IRobotUnitEditionServer
```

#### Description

Edition server.

#### Version

Available since version 3.5.

#### III.1.4.2.2 UseMetricAsDefault

##### C++

```
HRESULT get_UseMetricAsDefault(VARIANT_BOOL*);  
HRESULT put_UseMetricAsDefault(VARIANT_BOOL);
```

##### C#

```
public bool UseMetricAsDefault { get; set; }
```

##### Visual Basic

```
Public UseMetricAsDefault As Boolean
```

#### Description

Flag indicating if the software uses metric units by default.

#### Version

Available since version 9.7.

### III.1.4.3 IRobotUnitMngr Methods

The methods of the IRobotUnitMngr class are listed here.

## Public Methods

	Name	Description
≡	Count ( [ see page 1280)	Function returns the number of available units of the specified type. .
≡	Get ( [ see page 1280)	Function takes definition of the active unit for a given type.
≡	GetCoeff ( [ see page 1280)	The function returns the coefficient of conversion of the currently-selected unit of a given type to the base unit.
≡	GetCoeff2 ( [ see page 1281)	The function returns the conversion factor for a unit of a given type and specified index to the base unit.
≡	GetName ( [ see page 1281)	Function takes name of the unit of the specified type with the determined index. .
≡	Refresh ( [ see page 1281)	Function forces refreshing current information about units in the whole application.
≡	Set ( [ see page 1282)	Function determines active unit of the specified type based on the provided definition. .

### III.1.4.3.1 Count

#### C++

```
HRESULT Count(IRobotUnitType _type, long* ret);
```

#### C#

```
public long Count(IRobotUnitType _type);
```

#### Visual Basic

```
Public Function Count(_type As IRobotUnitType) As long
```

#### Description

Function returns the number of available units of the specified type. .

#### Version

Available since version 3.5.

### III.1.4.3.2 Get

#### C++

```
HRESULT Get(IRobotUnitType _type, IRobotUnitData** ret);
```

#### C#

```
public IRobotUnitData Get(IRobotUnitType _type);
```

#### Visual Basic

```
Public Function Get(_type As IRobotUnitType) As IRobotUnitData
```

#### Description

Function takes definition of the active unit for a given type.

#### Version

Available since version 3.5.

### III.1.4.3.3 GetCoeff

#### C++

```
HRESULT GetCoeff(IRobotUnitType _unit_type, double* ret);
```

**C#**

```
public double GetCoeff(IRobotUnitType _unit_type);
```

**Visual Basic**

```
Public Function GetCoeff(_unit_type As IRobotUnitType) As double
```

**Description**

The function returns the coefficient of conversion of the currently-selected unit of a given type to the base unit.

**III.1.4.3.4 GetCoeff2****C++**

```
HRESULT GetCoeff2(IRobotUnitType _unit_type, long _unit_idx, double* ret);
```

**C#**

```
public double GetCoeff2(IRobotUnitType _unit_type, long _unit_idx);
```

**Visual Basic**

```
Public Function GetCoeff2(_unit_type As IRobotUnitType, _unit_idx As long) As double
```

**Description**

The function returns the conversion factor for a unit of a given type and specified index to the base unit.

**Version**

Available since version 12.

**III.1.4.3.5 GetName****C++**

```
HRESULT GetName(IRobotUnitType _type, long _idx);
```

**C#**

```
public void GetName(IRobotUnitType _type, long _idx);
```

**Visual Basic**

```
Public Sub GetName(_type As IRobotUnitType, _idx As long)
```

**Description**

Function takes name of the unit of the specified type with the determined index. .

**Version**

Available since version 3.5.

**III.1.4.3.6 Refresh****C++**

```
HRESULT Refresh();
```

**C#**

```
public void Refresh();
```

**Visual Basic**

```
Public Sub Refresh()
```

## Description

Function forces refreshing current information about units in the whole application.

### III.1.4.3.7 Set

#### C++

```
HRESULT Set(IRobotUnitType _type, IRobotUnitData _data);
```

#### C#

```
public void Set(IRobotUnitType _type, IRobotUnitData _data);
```

#### Visual Basic

```
Public Sub Set(_type As IRobotUnitType, _data As IRobotUnitData)
```

#### Description

Function determines active unit of the specified type based on the provided definition. .

#### Version

Available since version 3.5.

## III.1.5 IRobotUnitComplexData

#### Class Hierarchy

#### C++

```
interface IRobotUnitComplexData : IDispatch;
```

#### C#

```
public interface IRobotUnitComplexData;
```

#### Visual Basic

```
Public Interface IRobotUnitComplexData
```

#### Description

Interface of composed unit definition. A composed unit is a unit being a function of other two units. For example, unit of the moment is a product of force unit and length unit. .

#### Version

Available since version 3.5.

### III.1.5.1 IRobotUnitComplexData Members

The following tables list the members exposed by IRobotUnitComplexData.

#### Public Fields

	Name	Description
◆	Name2 (see page 1283)	Name of the second component unit. .

### III.1.5.2 IRobotUnitComplexData Fields

The fields of the IRobotUnitComplexData class are listed here.

#### Public Fields

	Name	Description
◆	Name2 (see page 1283)	Name of the second component unit. .

### III.1.5.2.1 Name2

#### C++

```
HRESULT get_Name2(BSTR* );
HRESULT put_Name2(BSTR);
```

#### C#

```
public String Name2 { get; set; }
```

#### Visual Basic

```
Public Name2 As String
```

#### Description

Name of the second component unit. .

#### Version

Available since version 3.5.

## III.1.6 IRobotUnitEditionServer

#### Class Hierarchy

#### C++

```
interface IRobotUnitEditionServer : IDispatch;
```

#### C#

```
public interface IRobotUnitEditionServer;
```

#### Visual Basic

```
Public Interface IRobotUnitEditionServer
```

#### Description

Interface enabling unit edition. .

#### Version

Available since version 3.5.

### III.1.6.1 IRobotUnitEditionServer Members

The following tables list the members exposed by IRobotUnitEditionServer.

#### Public Methods

	Name	Description
💡	Count (🔗 see page 1284)	Function returns the number of units of the specified type. .
💡	Delete (🔗 see page 1284)	Function deletes a unit of the specified type with the determined index. .
💡	Find (🔗 see page 1284)	Function returns index of the unit of the specified type with the determined name. .
💡	Get (🔗 see page 1285)	Function takes the edit definition of unit of the specified type with the specified index. .
💡	New (🔗 see page 1285)	Function generates a new edit unit based on given parameters. .
💡	Set (🔗 see page 1285)	Function updates unit definition. .

### III.1.6.2 IRobotUnitEditionServer Methods

The methods of the IRobotUnitEditionServer class are listed here.

## Public Methods

	Name	Description
ESH	Count ( [ see page 1284)	Function returns the number of units of the specified type.. .
ESH	Delete ( [ see page 1284)	Function deletes a unit of the specified type with the determined index.. .
ESH	Find ( [ see page 1284)	Function returns index of the unit of the specified type with the determined name.. .
ESH	Get ( [ see page 1285)	Function takes the edit definition of unit of the specified type with the specified index.. .
ESH	New ( [ see page 1285)	Function generates a new edit unit based on given parameters.. .
ESH	Set ( [ see page 1285)	Function updates unit definition.. .

### III.1.6.2.1 Count

#### C++

```
HRESULT Count( IRobotUnitEditionType _type, long* ret);
```

#### C#

```
public long Count( IRobotUnitEditionType _type);
```

#### Visual Basic

```
Public Function Count(_type As IRobotUnitEditionType) As long
```

#### Description

Function returns the number of units of the specified type.. .

#### Version

Available since version 3.5.

### III.1.6.2.2 Delete

#### C++

```
HRESULT Delete( IRobotUnitEditionType _type, long _idx);
```

#### C#

```
public void Delete( IRobotUnitEditionType _type, long _idx);
```

#### Visual Basic

```
Public Sub Delete(_type As IRobotUnitEditionType, _idx As long)
```

#### Description

Function deletes a unit of the specified type with the determined index.. .

#### Version

Available since version 3.5.

### III.1.6.2.3 Find

#### C++

```
HRESULT Find( IRobotUnitEditionType _type, BSTR _name, long* ret);
```

#### C#

```
public long Find( IRobotUnitEditionType _type, String _name);
```

**Visual Basic**

```
Public Function Find(_type As IRobotUnitEditionType, _name As String) As long
```

**Description**

Function returns index of the unit of the specified type with the determined name.. .

**Version**

Available since version 3.5.

**III.1.6.2.4 Get****C++**

```
HRESULT Get(IRobotUnitEditionType _type, long _idx, IRobotUnitEditionData** ret);
```

**C#**

```
public IRobotUnitEditionData Get(IRobotUnitEditionType _type, long _idx);
```

**Visual Basic**

```
Public Function Get(_type As IRobotUnitEditionType, _idx As long) As IRobotUnitEditionData
```

**Description**

Function takes the edit definition of unit of the specified type with the specified index. .

**Version**

Available since version 3.5.

**III.1.6.2.5 New****C++**

```
HRESULT New(IRobotUnitEditionType _type, name _unit, double _coeff, long* ret);
```

**C#**

```
public long New(IRobotUnitEditionType _type, name _unit, double _coeff);
```

**Visual Basic**

```
Public Function New(_type As IRobotUnitEditionType, _unit As name, _coeff As double) As long
```

**Description**

Function generates a new edit unit based on given parameters. .

**Version**

Available since version 3.5.

**III.1.6.2.6 Set****C++**

```
HRESULT Set(IRobotUnitEditionData _data);
```

**C#**

```
public void Set(IRobotUnitEditionData _data);
```

**Visual Basic**

```
Public Sub Set(_data As IRobotUnitEditionData)
```

**Description**

Function updates unit definition. .

**Version**

Available since version 3.5.

**III.1.7 IRobotUnitEditionData****Class Hierarchy****C++**

```
interface IRobotUnitEditionData : IDispatch;
```

**C#**

```
public interface IRobotUnitEditionData;
```

**Visual Basic**

```
Public Interface IRobotUnitEditionData
```

**Description**

Interface of unit edit definition. An edit definition is a definition which determines a base unit and a proportionality factor. .

**Version**

Available since version 3.5.

**III.1.7.1 IRobotUnitEditionData Members**

The following tables list the members exposed by IRobotUnitEditionData.

**Public Fields**

	Name	Description
◆	Coefficient (see page 1286)	Number determining the ratio of defined unit to base unit.
◆	Type (see page 1287)	Type of edit unit.
◆	Unit (see page 1287)	Name of base unit .

**III.1.7.2 IRobotUnitEditionData Fields**

The fields of the IRobotUnitEditionData class are listed here.

**Public Fields**

	Name	Description
◆	Coefficient (see page 1286)	Number determining the ratio of defined unit to base unit.
◆	Type (see page 1287)	Type of edit unit.
◆	Unit (see page 1287)	Name of base unit .

**III.1.7.2.1 Coefficient****C++**

```
HRESULT get_Coefficient(double* );
HRESULT put_Coefficient(double);
```

**C#**

```
public double Coefficient { get; set; }
```

**Visual Basic**

```
Public Coefficient As double
```

**Description**

Number determining the ratio of defined unit to base unit.

**Version**

Available since version 3.5.

### III.1.7.2.2 Type

**C++**

```
HRESULT get_Type(IRobotUnitEditionType*);
```

**C#**

```
public IRobotUnitEditionType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotUnitEditionType
```

**Description**

Type of edit unit.

**Version**

Available since version 3.5.

### III.1.7.2.3 Unit

**C++**

```
HRESULT get_Unit(BSTR*);  
HRESULT put_Unit(BSTR);
```

**C#**

```
public String Unit { get; set; }
```

**Visual Basic**

```
Public Unit As String
```

**Description**

Name of base unit .

**Version**

Available since version 3.5.

## III.1.8 IRobotUnitMngrEvents

**Class Hierarchy****C++**

```
interface IRobotUnitMngrEvents : IDispatch;
```

**C#**

```
public interface IRobotUnitMngrEvents;
```

**Visual Basic**

```
Public Interface IRobotUnitMngrEvents
```

**Description**

Events (notifications) generated by the unit manager.

### III.1.8.1 IRobotUnitMngrEvents Members

The following tables list the members exposed by IRobotUnitMngrEvents.

#### Public Methods

	Name	Description
💡	UnitsChanged (🔗 see page 1288)	Event generated after any change in the settings of units.

### III.1.8.2 IRobotUnitMngrEvents Methods

The methods of the IRobotUnitMngrEvents class are listed here.

#### Public Methods

	Name	Description
💡	UnitsChanged (🔗 see page 1288)	Event generated after any change in the settings of units.

### III.1.8.2.1 UnitsChanged

#### C++

```
HRESULT UnitsChanged();
```

#### C#

```
public void UnitsChanged();
```

#### Visual Basic

```
Public Sub UnitsChanged()
```

#### Description

Event generated after any change in the settings of units.

## III.2 IRobotProjectPreferences

#### Class Hierarchy

#### C++

```
interface IRobotProjectPreferences : IDispatch;
```

#### C#

```
public interface IRobotProjectPreferences;
```

#### Visual Basic

```
Public Interface IRobotProjectPreferences
```

#### Description

Interface describing the settings for the current project (job).

### III.2.1 IRobotProjectPreferences Members

The following tables list the members exposed by IRobotProjectPreferences.

#### Public Fields

	Name	Description
💡	CalcModelCoherence (🔗 see page 1290)	
💡	EurocodeFactors (🔗 see page 1290)	Partial factors for Eurocode.

◆	KinematicConstraints ( <a href="#">see page 1291</a> )	
◆	Materials ( <a href="#">see page 1291</a> )	Active material database .
◆	MeshAutoAdjust ( <a href="#">see page 1291</a> )	Automatic adjustment of finite element mesh.
◆	MeshAutoAdjustIterationCount ( <a href="#">see page 1291</a> )	
◆	MeshParams ( <a href="#">see page 1292</a> )	Meshing parameters.
◆	MeshParamsFloors ( <a href="#">see page 1292</a> )	Mesh generation parameters for slabs.
◆	MeshParamsWalls ( <a href="#">see page 1292</a> )	Mesh generation parameters for walls.
◆	SectionsActive ( <a href="#">see page 1293</a> )	The list of active (selected) section databases .
◆	SectionsFound ( <a href="#">see page 1293</a> )	The list of available (found) names for the section database .
◆	Units ( <a href="#">see page 1293</a> )	Object managing units and formats.
◆	VehiclesActive ( <a href="#">see page 1293</a> )	List of names of active (selected) vehicle databases.
◆	VehiclesFound ( <a href="#">see page 1294</a> )	List of names of available (found) vehicle databases.

#### Public Methods

	Name	Description
◆	GetActiveCode ( <a href="#">see page 1294</a> )	The function returns the name of the active code of the indicated type.
◆	GetActiveCodeNumber ( <a href="#">see page 1295</a> )	Function returns the numerical identifier of an active code of a given type.
◆	GetCurrentDatabase ( <a href="#">see page 1295</a> )	Function returns a name of the current database of the given type. .
◆	Save ( <a href="#">see page 1295</a> )	Function forces saving changes in the project settings. .
◆	SetActiveCode ( <a href="#">see page 1296</a> )	The function sets the active code of the indicated type. If selection of an indicated code is not possible, then, zero value is returned (False). .
◆	SetActiveCodeNumber ( <a href="#">see page 1296</a> )	Function sets an active code of a given type based on the specified numerical identifier of the code.
◆	SetCurrentDatabase ( <a href="#">see page 1296</a> )	Function sets the given database as a current database of the given type.

### III.2.2 IRobotProjectPreferences Fields

The fields of the IRobotProjectPreferences class are listed here.

#### Public Fields

	Name	Description
◆	CalcModelCoherence ( <a href="#">see page 1290</a> )	
◆	EurocodeFactors ( <a href="#">see page 1290</a> )	Partial factors for Eurocode.
◆	KinematicConstraints ( <a href="#">see page 1291</a> )	
◆	Materials ( <a href="#">see page 1291</a> )	Active material database .
◆	MeshAutoAdjust ( <a href="#">see page 1291</a> )	Automatic adjustment of finite element mesh.
◆	MeshAutoAdjustIterationCount ( <a href="#">see page 1291</a> )	
◆	MeshParams ( <a href="#">see page 1292</a> )	Meshing parameters.
◆	MeshParamsFloors ( <a href="#">see page 1292</a> )	Mesh generation parameters for slabs.
◆	MeshParamsWalls ( <a href="#">see page 1292</a> )	Mesh generation parameters for walls.
◆	SectionsActive ( <a href="#">see page 1293</a> )	The list of active (selected) section databases .
◆	SectionsFound ( <a href="#">see page 1293</a> )	The list of available (found) names for the section database .

◆	Units (see page 1293)	Object managing units and formats.
◆	VehiclesActive (see page 1293)	List of names of active (selected) vehicle databases.
◆	VehiclesFound (see page 1294)	List of names of available (found) vehicle databases.

### III.2.2.1 CalcModelCoherence

#### C++

```
HRESULT get_CalcModelCoherence(IRobotCalculationModelCoherence* );
HRESULT put_CalcModelCoherence(IRobotCalculationModelCoherence);
```

#### C#

```
public IRobotCalculationModelCoherence CalcModelCoherence { get; set; }
```

#### Visual Basic

```
Public CalcModelCoherence As IRobotCalculationModelCoherence
```

#### Version

Available since version 12.

### III.2.2.2 EurocodeFactors

#### C++

```
HRESULT get_EurocodeFactors(IRobotEurocodeFactors** );
```

#### C#

```
public IRobotEurocodeFactors EurocodeFactors { get; }
```

#### Visual Basic

```
Public ReadOnly EurocodeFactors As IRobotEurocodeFactors
```

#### Description

Partial factors for Eurocode.

#### Version

Available since version 9.7.

### III.2.2.3 KinematicConstraints

#### C++

```
HRESULT get_KinematicConstraints(VARIANT_BOOL* );
HRESULT put_KinematicConstraints(VARIANT_BOOL);
```

#### C#

```
public bool KinematicConstraints { get; set; }
```

#### Visual Basic

```
Public KinematicConstraints As Boolean
```

#### Version

Available since version 12.

### III.2.2.4 Materials

#### C++

```
HRESULT get_Materials(IRobotMaterialDatabase** );
```

**C#**

```
public IRobotMaterialDatabase Materials { get; }
```

**Visual Basic**

```
Public ReadOnly Materials As IRobotMaterialDatabase
```

**Description**

Active material database .

**III.2.2.5 MeshAutoAdjust****C++**

```
HRESULT get_MeshAutoAdjust(VARIANT_BOOL* );
HRESULT put_MeshAutoAdjust(VARIANT_BOOL);
```

**C#**

```
public bool MeshAutoAdjust { get; set; }
```

**Visual Basic**

```
Public MeshAutoAdjust As Boolean
```

**Description**

Automatic adjustment of finite element mesh.

**Version**

Available since version 4.5.

**III.2.2.6 MeshAutoAdjustIterationCount****C++**

```
HRESULT get_MeshAutoAdjustIterationCount(long* );
HRESULT put_MeshAutoAdjustIterationCount(long);
```

**C#**

```
public long MeshAutoAdjustIterationCount { get; set; }
```

**Visual Basic**

```
Public MeshAutoAdjustIterationCount As long
```

**Version**

Available since version 13.4.

**III.2.2.7 MeshParams****C++**

```
HRESULT get_MeshParams(IRobotMeshParams** );
```

**C#**

```
public IRobotMeshParams MeshParams { get; }
```

**Visual Basic**

```
Public ReadOnly MeshParams As IRobotMeshParams
```

**Description**

Meshing parameters.

**Version**

Available since version 3.

### III.2.2.8 MeshParamsFloors

#### C++

```
HRESULT get_MeshParamsFloors(IRobotMeshParams**);
```

#### C#

```
public IRobotMeshParams MeshParamsFloors { get; }
```

#### Visual Basic

```
Public ReadOnly MeshParamsFloors As IRobotMeshParams
```

#### Description

Mesh generation parameters for slabs.

#### Version

Available since version 12.

### III.2.2.9 MeshParamsWalls

#### C++

```
HRESULT get_MeshParamsWalls(IRobotMeshParams**);
```

#### C#

```
public IRobotMeshParams MeshParamsWalls { get; }
```

#### Visual Basic

```
Public ReadOnly MeshParamsWalls As IRobotMeshParams
```

#### Description

Mesh generation parameters for walls.

#### Version

Available since version 12.

### III.2.2.10 SectionsActive

#### C++

```
HRESULT get_SectionsActive(IRobotSectionDatabaseList**);
```

#### C#

```
public IRobotSectionDatabaseList SectionsActive { get; }
```

#### Visual Basic

```
Public ReadOnly SectionsActive As IRobotSectionDatabaseList
```

#### Description

The list of active (selected) section databases .

### III.2.2.11 SectionsFound

#### C++

```
HRESULT get_SectionsFound(IRobotSectionDatabaseList**);
```

#### C#

```
public IRobotSectionDatabaseList SectionsFound { get; }
```

**Visual Basic**

```
Public ReadOnly SectionsFound As IRobotSectionDatabaseList
```

**Description**

The list of available (found) names for the section database .

**III.2.2.12 Units****C++**

```
HRESULT get_Units(IRobotUnitMngr**);
```

**C#**

```
public IRobotUnitMngr Units { get; }
```

**Visual Basic**

```
Public ReadOnly Units As IRobotUnitMngr
```

**Description**

Object managing units and formats.

**Version**

Available since version 3.

**III.2.2.13 VehiclesActive****C++**

```
HRESULT get_VehiclesActive(IRobotVehicleDatabaseList**);
```

**C#**

```
public IRobotVehicleDatabaseList VehiclesActive { get; }
```

**Visual Basic**

```
Public ReadOnly VehiclesActive As IRobotVehicleDatabaseList
```

**Description**

List of names of active (selected) vehicle databases.

**Version**

Available since version 3.

**III.2.2.14 VehiclesFound****C++**

```
HRESULT get_VehiclesFound(IRobotVehicleDatabaseList**);
```

**C#**

```
public IRobotVehicleDatabaseList VehiclesFound { get; }
```

**Visual Basic**

```
Public ReadOnly VehiclesFound As IRobotVehicleDatabaseList
```

**Description**

List of names of available (found) vehicle databases.

**Version**

Available since version 3.

### III.2.3 IRobotProjectPreferences Methods

The methods of the IRobotProjectPreferences class are listed here.

#### Public Methods

	Name	Description
ESH	GetActiveCode ( [ see page 1294] )	The function returns the name of the active code of the indicated type.
ESH	GetActiveCodeNumber ( [ see page 1295] )	Function returns the numerical identifier of an active code of a given type.
ESH	GetCurrentDatabase ( [ see page 1295] )	Function returns a name of the current database of the given type. .
ESH	Save ( [ see page 1295] )	Function forces saving changes in the project settings. .
ESH	SetActiveCode ( [ see page 1296] )	The function sets the active code of the indicated type. If selection of an indicated code is not possible, then, zero value is returned (False). .
ESH	SetActiveCodeNumber ( [ see page 1296] )	Function sets an active code of a given type based on the specified numerical identifier of the code.
ESH	SetCurrentDatabase ( [ see page 1296] )	Function sets the given database as a current database of the given type.

#### III.2.3.1 GetActiveCode

##### C++

```
HRESULT GetActiveCode(IRobotCodeType _code_type, BSTR* ret);
```

##### C#

```
public String GetActiveCode(IRobotCodeType _code_type);
```

##### Visual Basic

```
Public Function GetActiveCode(_code_type As IRobotCodeType) As String
```

##### Description

The function returns the name of the active code of the indicated type.

#### III.2.3.2 GetActiveCodeNumber

##### C++

```
HRESULT GetActiveCodeNumber(IRobotCodeType _code_type, long* ret);
```

##### C#

```
public long GetActiveCodeNumber(IRobotCodeType _code_type);
```

##### Visual Basic

```
Public Function GetActiveCodeNumber(_code_type As IRobotCodeType) As long
```

##### Description

Function returns the numerical identifier of an active code of a given type.

##### Version

Available since version 6.5.

#### III.2.3.3 GetCurrentDatabase

##### C++

```
HRESULT GetCurrentDatabase(IRobotDatabaseType _db_type, BSTR* ret);
```

**C#**

```
public String GetCurrentDatabase(IRobotDatabaseType _db_type);
```

**Visual Basic**

```
Public Function GetCurrentDatabase(_db_type As IRobotDatabaseType) As String
```

**Description**

Function returns a name of the current database of the given type. .

**Version**

Available since version 11.

**III.2.3.4 Save****C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Description**

Function forces saving changes in the project settings. .

**Version**

Available since version 4.5.

**III.2.3.5 SetActiveCode****C++**

```
HRESULT SetActiveCode(IRobotCodeType _code_type, BSTR _code_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetActiveCode(IRobotCodeType _code_type, String _code_name);
```

**Visual Basic**

```
Public Function SetActiveCode(_code_type As IRobotCodeType, _code_name As String) As Boolean
```

**Description**

The function sets the active code of the indicated type. If selection of an indicated code is not possible, then, zero value is returned (False). .

**III.2.3.6 SetActiveCodeNumber****C++**

```
HRESULT SetActiveCodeNumber(IRobotCodeType _code_type, long _code_number, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetActiveCodeNumber(IRobotCodeType _code_type, long _code_number);
```

## Visual Basic

```
Public Function SetActiveCodeNumber(_code_type As IRobotCodeType, _code_number As long) As Boolean
```

### Description

Function sets an active code of a given type based on the specified numerical identifier of the code.

### Version

Available since version 6.5.

## III.2.3.7 SetCurrentDatabase

### C++

```
HRESULT SetCurrentDatabase(IRobotDatabaseType _db_type, BSTR _db_name, VARIANT_BOOL* ret);
```

### C#

```
public bool SetCurrentDatabase(IRobotDatabaseType _db_type, String _db_name);
```

## Visual Basic

```
Public Function SetCurrentDatabase(_db_type As IRobotDatabaseType, _db_name As String) As Boolean
```

### Description

Function sets the given database as a current database of the given type.

### Version

Available since version 11.

## III.3 IRobotSectionDatabaseList

### Class Hierarchy

### C++

```
interface IRobotSectionDatabaseList : IDispatch;
```

### C#

```
public interface IRobotSectionDatabaseList;
```

## Visual Basic

```
Public Interface IRobotSectionDatabaseList
```

### Description

The interface describing a list of section databases .

## III.3.1 IRobotSectionDatabaseList Members

The following tables list the members exposed by IRobotSectionDatabaseList.

### Public Fields

	Name	Description
◆	Count (see page 1298)	Number of section databases in the list.

## Public Methods

	Name	Description
✳	Add (see page 1298)	The function adds a section database with a given name at the end of the list. It returns the message about carrying out the operation successfully.
✳	AddFromFile (see page 1299)	The function adds a section database defined in the indicated file (.mdb) at the end of the list. It returns the message about carrying out the operation successfully.
✳	ChangeIndex (see page 1299)	The function imposes a new position in the list for the selected database. By means of the function one may define a required order of section databases in the list. The order is important while searching through the databases for a section with a given name. .
✳	Find (see page 1299)	The function returns the list index of the database with the indicated name. If the required database is absent from the list, the function returns zero. .
✳	Get (see page 1299)	The function returns the name of section database saved in the list under the indicated index. The list elements are indexed from 1 to Count (see page 1298).
✳	GetDatabase (see page 1300)	The function returns the section database whose name is saved in the list in the indicated position.
✳	Remove (see page 1300)	The function deletes a database from the list. .

## III.3.2 IRobotSectionDatabaseList Fields

The fields of the IRobotSectionDatabaseList class are listed here.

### Public Fields

	Name	Description
✳	Count (see page 1298)	Number of section databases in the list.

### III.3.2.1 Count

#### C++

```
HRESULT get_Count(int*);
```

#### C#

```
public int Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Integer
```

#### Description

Number of section databases in the list.

## III.3.3 IRobotSectionDatabaseList Methods

The methods of the IRobotSectionDatabaseList class are listed here.

### Public Methods

	Name	Description
✳	Add (see page 1298)	The function adds a section database with a given name at the end of the list. It returns the message about carrying out the operation successfully.
✳	AddFromFile (see page 1299)	The function adds a section database defined in the indicated file (.mdb) at the end of the list. It returns the message about carrying out the operation successfully.

	ChangeIndex (see page 1299)	The function imposes a new position in the list for the selected database. By means of the function one may define a required order of section databases in the list. The order is important while searching through the databases for a section with a given name. .
	Find (see page 1299)	The function returns the list index of the database with the indicated name. If the required database is absent from the list, the function returns zero. .
	Get (see page 1299)	The function returns the name of section database saved in the list under the indicated index. The list elements are indexed from 1 to Count (see page 1298).
	GetDatabase (see page 1300)	The function returns the section database whose name is saved in the list in the indicated position.
	Remove (see page 1300)	The function deletes a database from the list. .

### III.3.3.1 Add

C++

```
HRESULT Add(BSTR _db_name, VARIANT_BOOL* ret);
```

C#

```
public bool Add(String _db_name);
```

Visual Basic

```
Public Function Add(_db_name As String) As Boolean
```

Description

The function adds a section database with a given name at the end of the list. It returns the message about carrying out the operation successfully.

### III.3.3.2 AddFromFile

C++

```
HRESULT AddFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

C#

```
public bool AddFromFile(String _file_path);
```

Visual Basic

```
Public Function AddFromFile(_file_path As String) As Boolean
```

Description

The function adds a section database defined in the indicated file (.mdb) at the end of the list. It returns the message about carrying out the operation successfully.

### III.3.3.3 ChangeIndex

C++

```
HRESULT ChangeIndex(int _cur_idx, int _new_idx);
```

C#

```
public void ChangeIndex(int _cur_idx, int _new_idx);
```

Visual Basic

```
Public Sub ChangeIndex(_cur_idx As int, _new_idx As int)
```

## Description

The function imposes a new position in the list for the selected database. By means of the function one may define a required order of section databases in the list. The order is important while searching through the databases for a section with a given name. .

### III.3.3.4 Find

#### C++

```
HRESULT Find(BSTR _db_name, int* ret);
```

#### C#

```
public int Find(String _db_name);
```

#### Visual Basic

```
Public Function Find(_db_name As String) As int
```

## Description

The function returns the list index of the database with the indicated name. If the required database is absent from the list, the function returns zero. .

### III.3.3.5 Get

#### C++

```
HRESULT Get(int _idx, BSTR* ret);
```

#### C#

```
public String Get(int _idx);
```

#### Visual Basic

```
Public Function Get(_idx As int) As String
```

## Description

The function returns the name of section database saved in the list under the indicated index. The list elements are indexed from 1 to Count (see page 1298).

### III.3.3.6 GetDatabase

#### C++

```
HRESULT GetDatabase(int _idx, IRobotSectionDatabase** ret);
```

#### C#

```
public IRobotSectionDatabase GetDatabase(int _idx);
```

#### Visual Basic

```
Public Function GetDatabase(_idx As int) As IRobotSectionDatabase
```

## Description

The function returns the section database whose name is saved in the list in the indicated position.

### III.3.3.7 Remove

#### C++

```
HRESULT Remove(int _idx);
```

**C#**

```
public void Remove(int _idx);
```

**Visual Basic**

```
Public Sub Remove(_idx As int)
```

**Description**

The function deletes a database from the list. .

**III.4 IRobotMaterialDatabase****Class Hierarchy****C++**

```
interface IRobotMaterialDatabase : IDispatch;
```

**C#**

```
public interface IRobotMaterialDatabase;
```

**Visual Basic**

```
Public Interface IRobotMaterialDatabase
```

**Description**

The interface providing access to material database .

**III.4.1 IRobotMaterialDatabase Members**

The following tables list the members exposed by IRobotMaterialDatabase.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Name ( <a href="#">see page 1301</a> )	Material database name .

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Get ( <a href="#">see page 1302</a> )	The function returns a table of names of materials of the indicated type that are defined in the database .
❖	GetAll ( <a href="#">see page 1302</a> )	The function returns a table of names of all materials defined in the database .
❖	GetDefault ( <a href="#">see page 1302</a> )	Function returns the name of a selected type of default material. .
❖	Load ( <a href="#">see page 1303</a> )	The function reads materials from the database with the indicated name. If there occurs an error during the operation of reading from the database or the database with the name does not exist, the function returns zero (False).
❖	LoadFromFile ( <a href="#">see page 1303</a> )	The function reads the materials from the indicated file. If there occurs an error during the operation of reading from the database, the function returns zero (False)..
❖	SetDefault ( <a href="#">see page 1303</a> )	Function sets the name of a selected type of default material..

**III.4.2 IRobotMaterialDatabase Fields**

The fields of the IRobotMaterialDatabase class are listed here.

## Public Fields

	Name	Description
◆	Name (see page 1301)	Material database name .

### III.4.2.1 Name

#### C++

```
HRESULT get_Name(BSTR*);
```

#### C#

```
public String Name { get; }
```

#### Visual Basic

```
Public ReadOnly Name As String
```

#### Description

Material database name .

## III.4.3 IRobotMaterialDatabase Methods

The methods of the IRobotMaterialDatabase class are listed here.

#### Public Methods

	Name	Description
◆	Get (see page 1302)	The function returns a table of names of materials of the indicated type that are defined in the database .
◆	GetAll (see page 1302)	The function returns a table of names of all materials defined in the database .
◆	GetDefault (see page 1302)	Function returns the name of a selected type of default material. .
◆	Load (see page 1303)	The function reads materials from the database with the indicated name. If there occurs an error during the operation of reading from the database or the database with the name does not exist, the function returns zero (False).
◆	LoadFromFile (see page 1303)	The function reads the materials from the indicated file. If there occurs an error during the operation of reading from the database, the function returns zero (False). .
◆	SetDefault (see page 1303)	Function sets the name of a selected type of default material. .

### III.4.3.1 Get

#### C++

```
HRESULT Get(IRobotMaterialType _mater_type, IRobotNamesArray** ret);
```

#### C#

```
public IRobotNamesArray Get(IRobotMaterialType _mater_type);
```

#### Visual Basic

```
Public Function Get(_mater_type As IRobotMaterialType) As IRobotNamesArray
```

#### Description

The function returns a table of names of materials of the indicated type that are defined in the database .

### III.4.3.2 GetAll

**C++**

```
HRESULT GetAll(IRobotNamesArray** ret);
```

**C#**

```
public IRobotNamesArray GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotNamesArray
```

**Description**

The function returns a table of names of all materials defined in the database .

### III.4.3.3 GetDefault

**C++**

```
HRESULT GetDefault(IRobotMaterialType _mater_type, BSTR* ret);
```

**C#**

```
public String GetDefault(IRobotMaterialType _mater_type);
```

**Visual Basic**

```
Public Function GetDefault(_mater_type As IRobotMaterialType) As String
```

**Description**

Function returns the name of a selected type of default material. .

**Version**

Available since version 3.

### III.4.3.4 Load

**C++**

```
HRESULT Load(BSTR _dbase_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Load(String _dbase_name);
```

**Visual Basic**

```
Public Function Load(_dbase_name As String) As Boolean
```

**Description**

The function reads materials from the database with the indicated name. If there occurs an error during the operation of reading from the database or the database with the name does not exist, the function returns zero (False).

### III.4.3.5 LoadFromFile

**C++**

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromFile(String _file_path);
```

**Visual Basic**

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

**Description**

The function reads the materials from the indicated file. If there occurs an error during the operation of reading from the database, the function returns zero (False). .

**III.4.3.6 SetDefault****C++**

```
HRESULT SetDefault(IRobotMaterialType _mater_type, BSTR _mater_name);
```

**C#**

```
public void SetDefault(IRobotMaterialType _mater_type, String _mater_name);
```

**Visual Basic**

```
Public Sub SetDefault(_mater_type As IRobotMaterialType, _mater_name As String)
```

**Description**

Function sets the name of a selected type of default material. .

**Version**

Available since version 3.

**III.5 IRobotCodeType****C++**

```
enum IRobotCodeType;
```

**C#**

```
public enum IRobotCodeType;
```

**Visual Basic**

```
Public Enum IRobotCodeType
```

**Members**

Members	Description
I_CT_STEEL_STRUCTURES = 0	The code for steel structures .
I_CT_STEEL_CONNECTIONS = 1	The code for steel connections .
I_CT_TIMBER_STRUCTURES = 2	The code for timber structures .
I_CT_RC_REAL_REINF = 3	The code for real reinforcement .
I_CT_RC_THEORETICAL_REINF = 4	The code for theoretical reinforcement .
I_CT_FOUNDATIONS_DESIGN = 5	Code for spread footing design.
I_CT_CODE_COMBINATIONS = 9	Code combinations.
I_CT_SNOW_WIND_LOADS = 6	Code for snow/wind loads.
I_CT_SEISMIC_LOADS = 7	Code for seismic loads.
I_CT_MOVING_LOADS = 8	Code for moving loads.

**Description**

Set of identifiers determining the types of codes that may be set in the Robot program.

## III.6 IRobotSectionDatabase

### Class Hierarchy

#### C++

```
interface IRobotSectionDatabase : IDispatch;
```

#### C#

```
public interface IRobotSectionDatabase;
```

### Visual Basic

```
Public Interface IRobotSectionDatabase
```

#### Description

The interface providing access to section database.

### III.6.1 IRobotSectionDatabase Members

The following tables list the members exposed by IRobotSectionDatabase.

#### Public Fields

	Name	Description
❖	Description (see page 1305)	Section database description.
❖	FullName (see page 1305)	Full name of section database.
❖	Name (see page 1305)	Section database name .

#### Public Methods

	Name	Description
✳	GetAll (see page 1306)	The function returns a table containing names of all sections described in the database.
✳	Load (see page 1306)	The function reads sections from the section database with the indicated name. If such section database does not exist or there occurs an error during the reading process, the function returns zero. .
✳	LoadFromFile (see page 1307)	The function reads sections from the database saved in the indicated file. If there occurred an error during the reading process, the function returns zero.

### III.6.2 IRobotSectionDatabase Fields

The fields of the IRobotSectionDatabase class are listed here.

#### Public Fields

	Name	Description
❖	Description (see page 1305)	Section database description.
❖	FullName (see page 1305)	Full name of section database.
❖	Name (see page 1305)	Section database name .

#### III.6.2.1 Description

##### C++

```
HRESULT get_Description(BSTR*);
```

##### C#

```
public String Description { get; }
```

**Visual Basic**

```
Public ReadOnly Description As String
```

**Description**

Section database description.

**Version**

Available since version 7.5.

**III.6.2.2 FullName****C++**

```
HRESULT get_FullName(BSTR* );
```

**C#**

```
public String FullName { get; }
```

**Visual Basic**

```
Public ReadOnly FullName As String
```

**Description**

Full name of section database.

**Version**

Available since version 4.

**III.6.2.3 Name****C++**

```
HRESULT get_Name(BSTR* );
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Description**

Section database name .

**III.6.3 IRobotSectionDatabase Methods**

The methods of the IRobotSectionDatabase class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	GetAll ( <a href="#">see page 1306</a> )	The function returns a table containing names of all sections described in the database.
≡	Load ( <a href="#">see page 1306</a> )	The function reads sections from the section database with the indicated name. If such section database does not exist or there occurs an error during the reading process, the function returns zero. .
≡	LoadFromFile ( <a href="#">see page 1307</a> )	The function reads sections from the database saved in the indicated file. If there occurred an error during the reading process, the function returns zero.

### III.6.3.1 GetAll

**C++**

```
HRESULT GetAll(IRobotNamesArray** ret);
```

**C#**

```
public IRobotNamesArray GetAll();
```

**Visual Basic**

```
Public Function GetAll() As IRobotNamesArray
```

**Description**

The function returns a table containing names of all sections described in the database.

### III.6.3.2 Load

**C++**

```
HRESULT Load(BSTR _db_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Load(String _db_name);
```

**Visual Basic**

```
Public Function Load(_db_name As String) As Boolean
```

**Description**

The function reads sections from the section database with the indicated name. If such section database does not exist or there occurs an error during the reading process, the function returns zero. .

### III.6.3.3 LoadFromFile

**C++**

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromFile(String _file_path);
```

**Visual Basic**

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

**Description**

The function reads sections from the database saved in the indicated file. If there occurred an error during the reading process, the function returns zero.

## III.7 IRobotProjectPreferencesEvents

**Class Hierarchy**

**C++**

```
interface IRobotProjectPreferencesEvents : IDispatch;
```

**C#**

```
public interface IRobotProjectPreferencesEvents;
```

**Visual Basic**

```
Public Interface IRobotProjectPreferencesEvents
```

**Description**

Events related to the change of settings for the current project.

**III.7.1 IRobotProjectPreferencesEvents Members**

The following tables list the members exposed by IRobotProjectPreferencesEvents.

**Public Methods**

	<b>Name</b>	<b>Description</b>
=	OnDialogOK ( see page 1307)	Event generated after accepting changes made in the project settings dialog box by the user.

**III.7.2 IRobotProjectPreferencesEvents Methods**

The methods of the IRobotProjectPreferencesEvents class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
=	OnDialogOK ( see page 1307)	Event generated after accepting changes made in the project settings dialog box by the user.

**III.7.2.1 OnDialogOK****C++**

```
HRESULT OnDialogOK();
```

**C#**

```
public void OnDialogOK();
```

**Visual Basic**

```
Public Sub OnDialogOK()
```

**Description**

Event generated after accepting changes made in the project settings dialog box by the user.

**III.8 IRobotEurocodeSteelDesignFactors****Class Hierarchy****C++**

```
interface IRobotEurocodeSteelDesignFactors : IDispatch;
```

**C#**

```
public interface IRobotEurocodeSteelDesignFactors;
```

**Visual Basic**

```
Public Interface IRobotEurocodeSteelDesignFactors
```

**Description**

Partial factors for the steel design according to Eurocode.

## Version

Available since version 9.7.

### III.8.1 IRobotEurocodeSteelDesignFactors Members

The following tables list the members exposed by IRobotEurocodeSteelDesignFactors.

#### Public Fields

	Name	Description
❖	Gamma0 (see page 1309)	Resistance of elements and sections gM0.
❖	Gamma1 (see page 1309)	Resistance of elements and sections gM1.
❖	Gamma2 (see page 1309)	Resistance of elements and sections gM2.
❖	GammaFire (see page 1310)	Safety factor for fire conditions.

#### Public Methods

	Name	Description
❖	LoadFromCode (see page 1310)	Function loads values of coefficients from the code with the specified name.
❖	LoadFromCodeNumber (see page 1310)	Function loads values of coefficients from the specified code.

### III.8.2 IRobotEurocodeSteelDesignFactors Fields

The fields of the IRobotEurocodeSteelDesignFactors class are listed here.

#### Public Fields

	Name	Description
❖	Gamma0 (see page 1309)	Resistance of elements and sections gM0.
❖	Gamma1 (see page 1309)	Resistance of elements and sections gM1.
❖	Gamma2 (see page 1309)	Resistance of elements and sections gM2.
❖	GammaFire (see page 1310)	Safety factor for fire conditions.

#### III.8.2.1 Gamma0

##### C++

```
HRESULT get_Gamma0(double* );
HRESULT put_Gamma0(double);
```

##### C#

```
public double Gamma0 { get; set; }
```

##### Visual Basic

```
Public Gamma0 As Double
```

##### Description

Resistance of elements and sections gM0.

#### Version

Available since version 9.7.

#### III.8.2.2 Gamma1

##### C++

```
HRESULT get_Gamma1(double* );
HRESULT put_Gamma1(double);
```

**C#**

```
public double Gamma1 { get; set; }
```

**Visual Basic**

```
Public Gamma1 As Double
```

**Description**

Resistance of elements and sections gM1.

**Version**

Available since version 9.7.

**III.8.2.3 Gamma2****C++**

```
HRESULT get_Gamma2(double*);  
HRESULT put_Gamma2(double);
```

**C#**

```
public double Gamma2 { get; set; }
```

**Visual Basic**

```
Public Gamma2 As Double
```

**Description**

Resistance of elements and sections gM2.

**Version**

Available since version 9.7.

**III.8.2.4 GammaFire****C++**

```
HRESULT get_GammaFire(double*);  
HRESULT put_GammaFire(double);
```

**C#**

```
public double GammaFire { get; set; }
```

**Visual Basic**

```
Public GammaFire As Double
```

**Description**

Safety factor for fire conditions.

**Version**

Available since version 9.7.

**III.8.3 IRobotEurocodeSteelDesignFactors Methods**

The methods of the IRobotEurocodeSteelDesignFactors class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	LoadFromCode (see page 1310)	Function loads values of coefficients from the code with the specified name.

	LoadFromCodeNumber ( <a href="#">see page 1310</a> )	Function loads values of coefficients from the specified code.
-----------------------------------------------------------------------------------	------------------------------------------------------	----------------------------------------------------------------

### III.8.3.1 LoadFromCode

#### C++

```
HRESULT LoadFromCode(BSTR _code_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromCode(String _code_name);
```

#### Visual Basic

```
Public Function LoadFromCode(_code_name As String) As Boolean
```

#### Description

Function loads values of coefficients from the code with the specified name.

#### Version

Available since version 9.7.

### III.8.3.2 LoadFromCodeNumber

#### C++

```
HRESULT LoadFromCodeNumber(long _code_number, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromCodeNumber(long _code_number);
```

#### Visual Basic

```
Public Function LoadFromCodeNumber(_code_number As long) As Boolean
```

#### Description

Function loads values of coefficients from the specified code.

#### Version

Available since version 9.7.

## III.9 IRobotEurocodeSteelConnectionFactors

#### Class Hierarchy

#### C++

```
interface IRobotEurocodeSteelConnectionFactors : IDispatch;
```

#### C#

```
public interface IRobotEurocodeSteelConnectionFactors;
```

#### Visual Basic

```
Public Interface IRobotEurocodeSteelConnectionFactors
```

#### Description

Partial factors for the steel connection design according to Eurocode.

#### Version

Available since version 9.7.

### III.9.1 IRobotEurocodeSteelConnectionFactors Members

The following tables list the members exposed by IRobotEurocodeSteelConnectionFactors.

#### Public Fields

	Name	Description
❖	Gamma0 (see page 1312)	Resistance of elements and sections gM0.
❖	Gamma1 (see page 1312)	Resistance of elements and sections gM1.
❖	Gamma2 (see page 1313)	Resistance of fasteners and plates in bearing gM2.
❖	Gamma3 (see page 1313)	Slip resistance - at ultimate limit state gM3.
❖	Gamma3Ser (see page 1313)	Slip resistance - at serviceability limit state gM3ser.
❖	Gamma4 (see page 1314)	Bearing resistance of an injection bolt gM4.
❖	Gamma5 (see page 1314)	Resistance of joints in hollow section lattice girder gM5.
❖	Gamma6 (see page 1314)	Resistance of pins at serviceability limit state gM6.
❖	Gamma7 (see page 1315)	Preload of high strength bolts gM7.
❖	GammaC (see page 1315)	Concrete capacity gC .

#### Public Methods

	Name	Description
❖	LoadFromCode (see page 1315)	Function loads values of coefficients from the code with the specified name.
❖	LoadFromCodeNumber (see page 1316)	Function loads values of coefficients from the specified code.

### III.9.2 IRobotEurocodeSteelConnectionFactors Fields

The fields of the IRobotEurocodeSteelConnectionFactors class are listed here.

#### Public Fields

	Name	Description
❖	Gamma0 (see page 1312)	Resistance of elements and sections gM0.
❖	Gamma1 (see page 1312)	Resistance of elements and sections gM1.
❖	Gamma2 (see page 1313)	Resistance of fasteners and plates in bearing gM2.
❖	Gamma3 (see page 1313)	Slip resistance - at ultimate limit state gM3.
❖	Gamma3Ser (see page 1313)	Slip resistance - at serviceability limit state gM3ser.
❖	Gamma4 (see page 1314)	Bearing resistance of an injection bolt gM4.
❖	Gamma5 (see page 1314)	Resistance of joints in hollow section lattice girder gM5.
❖	Gamma6 (see page 1314)	Resistance of pins at serviceability limit state gM6.
❖	Gamma7 (see page 1315)	Preload of high strength bolts gM7.
❖	GammaC (see page 1315)	Concrete capacity gC .

#### III.9.2.1 Gamma0

##### C++

```
HRESULT get_Gamma0(double* );
HRESULT put_Gamma0(double );
```

##### C#

```
public double Gamma0 { get; set; }
```

##### Visual Basic

```
Public Gamma0 As Double
```

**Description**

Resistance of elements and sections gM0.

**Version**

Available since version 9.7.

**III.9.2.2 Gamma1****C++**

```
HRESULT get_Gamma1(double*);  
HRESULT put_Gamma1(double);
```

**C#**

```
public double Gamma1 { get; set; }
```

**Visual Basic**

```
Public Gamma1 As Double
```

**Description**

Resistance of elements and sections gM1.

**Version**

Available since version 9.7.

**III.9.2.3 Gamma2****C++**

```
HRESULT get_Gamma2(double*);  
HRESULT put_Gamma2(double);
```

**C#**

```
public double Gamma2 { get; set; }
```

**Visual Basic**

```
Public Gamma2 As Double
```

**Description**

Resistance of fasteners and plates in bearing gM2.

**Version**

Available since version 9.7.

**III.9.2.4 Gamma3****C++**

```
HRESULT get_Gamma3(double*);  
HRESULT put_Gamma3(double);
```

**C#**

```
public double Gamma3 { get; set; }
```

**Visual Basic**

```
Public Gamma3 As Double
```

**Description**

Slip resistance - at ultimate limit state gM3.

**Version**

Available since version 9.7.

**III.9.2.5 Gamma3Ser****C++**

```
HRESULT get_Gamma3Ser(double*);  
HRESULT put_Gamma3Ser(double);
```

**C#**

```
public double Gamma3Ser { get; set; }
```

**Visual Basic**

```
Public Gamma3Ser As double
```

**Description**

Slip resistance - at serviceability limit state gM3ser.

**Version**

Available since version 9.7.

**III.9.2.6 Gamma4****C++**

```
HRESULT get_Gamma4(double*);  
HRESULT put_Gamma4(double);
```

**C#**

```
public double Gamma4 { get; set; }
```

**Visual Basic**

```
Public Gamma4 As double
```

**Description**

Bearing resistance of an injection bolt gM4.

**Version**

Available since version 9.7.

**III.9.2.7 Gamma5****C++**

```
HRESULT get_Gamma5(double*);  
HRESULT put_Gamma5(double);
```

**C#**

```
public double Gamma5 { get; set; }
```

**Visual Basic**

```
Public Gamma5 As double
```

**Description**

Resistance of joints in hollow section lattice girder gM5.

**Version**

Available since version 9.7.

### III.9.2.8 Gamma6

#### C++

```
HRESULT get_Gamma6(double*);  
HRESULT put_Gamma6(double);
```

#### C#

```
public double Gamma6 { get; set; }
```

#### Visual Basic

```
Public Gamma6 As double
```

#### Description

Resistance of pins at serviceability limit state gM6.

#### Version

Available since version 9.7.

### III.9.2.9 Gamma7

#### C++

```
HRESULT get_Gamma7(double*);  
HRESULT put_Gamma7(double);
```

#### C#

```
public double Gamma7 { get; set; }
```

#### Visual Basic

```
Public Gamma7 As double
```

#### Description

Preload of high strength bolts gM7.

#### Version

Available since version 9.7.

### III.9.2.10 GammaC

#### C++

```
HRESULT get_GammaC(double*);  
HRESULT put_GammaC(double);
```

#### C#

```
public double GammaC { get; set; }
```

#### Visual Basic

```
Public GammaC As double
```

#### Description

Concrete capacity gC .

#### Version

Available since version 9.7.

## III.9.3 IRobotEurocodeSteelConnectionFactors Methods

The methods of the IRobotEurocodeSteelConnectionFactors class are listed here.

## Public Methods

	Name	Description
💡	LoadFromCode (see page 1315)	Function loads values of coefficients from the code with the specified name.
💡	LoadFromCodeNumber (see page 1316)	Function loads values of coefficients from the specified code.

### III.9.3.1 LoadFromCode

#### C++

```
HRESULT LoadFromCode(BSTR _code_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromCode(String _code_name);
```

#### Visual Basic

```
Public Function LoadFromCode(_code_name As String) As Boolean
```

#### Description

Function loads values of coefficients from the code with the specified name.

#### Version

Available since version 9.7.

### III.9.3.2 LoadFromCodeNumber

#### C++

```
HRESULT LoadFromCodeNumber(long _code_number, VARIANT_BOOL* ret);
```

#### C#

```
public bool LoadFromCodeNumber(long _code_number);
```

#### Visual Basic

```
Public Function LoadFromCodeNumber(_code_number As long) As Boolean
```

#### Description

Function loads values of coefficients from the specified code.

#### Version

Available since version 9.7.

## III.10 IRobotEurocodeFactors

#### Class Hierarchy

#### C++

```
interface IRobotEurocodeFactors : IDispatch;
```

#### C#

```
public interface IRobotEurocodeFactors;
```

#### Visual Basic

```
Public Interface IRobotEurocodeFactors
```

**Description**

Partial factors for Eurocodes.

**Version**

Available since version 9.7.

**III.10.1 IRobotEurocodeFactors Members**

The following tables list the members exposed by IRobotEurocodeFactors.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	SteelConnections ( <a href="#">see page 1317</a> )	Partial factors for steel connections according to Eurocode.
❖	SteelDesign ( <a href="#">see page 1317</a> )	Partial factors for the steel design according to Eurocode.

**III.10.2 IRobotEurocodeFactors Fields**

The fields of the IRobotEurocodeFactors class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	SteelConnections ( <a href="#">see page 1317</a> )	Partial factors for steel connections according to Eurocode.
❖	SteelDesign ( <a href="#">see page 1317</a> )	Partial factors for the steel design according to Eurocode.

**III.10.2.1 SteelConnections****C++**

```
HRESULT get_SteelConnections(IRobotEurocodeSteelConnectionFactors**);
```

**C#**

```
public IRobotEurocodeSteelConnectionFactors SteelConnections { get; }
```

**Visual Basic**

```
Public ReadOnly SteelConnections As IRobotEurocodeSteelConnectionFactors
```

**Description**

Partial factors for steel connections according to Eurocode.

**Version**

Available since version 9.7.

**III.10.2.2 SteelDesign****C++**

```
HRESULT get_SteelDesign(IRobotEurocodeSteelDesignFactors**);
```

**C#**

```
public IRobotEurocodeSteelDesignFactors SteelDesign { get; }
```

**Visual Basic**

```
Public ReadOnly SteelDesign As IRobotEurocodeSteelDesignFactors
```

**Description**

Partial factors for the steel design according to Eurocode.

**Version**

Available since version 9.7.

## III.11 IRobotDatabaseType

**C++**

```
enum IRobotDatabaseType;
```

**C#**

```
public enum IRobotDatabaseType;
```

**Visual Basic**

```
Public Enum IRobotDatabaseType
```

**Members**

Members	Description
I_DT_SECTIONS = 1	Database of steel and timber sections. Available since version 11.
I_DT_VEHICLE_LOADS = 2	Vehicle database. Available since version 11.
I_DT_STANDARD_LOADS = 3	Database of loads. Available since version 11.
I_DT_BUILDING_SOILS = 4	Soil database. Available since version 11.
I_DT_BOLTS = 5	Bolt database. Available since version 11.
I_DT_ANCHOR_BOLTS = 6	Anchor bolt database. Available since version 11.
I_DT_REINFORCING_BARS = 7	Database of reinforcing bars. Available since version 11.
I_DT_WIRE_FABRICS = 8	Database of wire fabrics. Available since version 11.

**Description**

Available database types.

**Version**

Available since version 11.

## III.12 IRobotCalculationModelCoherence

**C++**

```
enum IRobotCalculationModelCoherence;
```

**C#**

```
public enum IRobotCalculationModelCoherence;
```

**Visual Basic**

```
Public Enum IRobotCalculationModelCoherence
```

## Members

Members	Description
I_CMC_COHERENT_FE_MESH = 1	Available since version 12.
I_CMC_USE_OF_KINEMATIC_CONSTRAINTS = 2	Available since version 12.

## Version

Available since version 12.

# IV Backgrounds

Available since version 11.

## Enumerations

	Name	Description
	IRobotBackgroundVisibilityRangeType	Available types of background visibility range. ( <a href="#">see page 1329</a> )

## Interfaces

	Name	Description
	IRobotBackgroundLayers ( <a href="#">see page 1319</a> )	Background layers.
	IRobotBackgroundInsertParams ( <a href="#">see page 1321</a> )	Parameters of locating a background in the project.
	IRobotBackground ( <a href="#">see page 1325</a> )	
	IRobotBackgroundVisibilityRange ( <a href="#">see page 1329</a> )	
	IRobotBackgroundServer ( <a href="#">see page 1331</a> )	Object managing backgrounds in the project.

## IV.1 IRobotBackgroundLayers

### Class Hierarchy

#### C++

```
interface IRobotBackgroundLayers : IDispatch;
```

#### C#

```
public interface IRobotBackgroundLayers;
```

### Visual Basic

```
Public Interface IRobotBackgroundLayers
```

### Description

Background layers.

## Version

Available since version 11.

### IV.1.1 IRobotBackgroundLayers Members

The following tables list the members exposed by IRobotBackgroundLayers.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1320)	Number of layers.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	FindName (see page 1320)	Function returns index of the layer of the given name.
◆	Get (see page 1321)	Function makes data for a layer of the given index available.
◆	IsImported (see page 1321)	Function returns True if a layer with the given index is imported with the background to the project.
◆	SetImported (see page 1321)	Function allows for changing the status for the layer of the given index.

**IV.1.2 IRobotBackgroundLayers Fields**

The fields of the IRobotBackgroundLayers class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1320)	Number of layers.

**IV.1.2.1 Count****C++**

```
HRESULT get_Count(long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of layers.

**Version**

Available since version 11.

**IV.1.3 IRobotBackgroundLayers Methods**

The methods of the IRobotBackgroundLayers class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	FindName (see page 1320)	Function returns index of the layer of the given name.
◆	Get (see page 1321)	Function makes data for a layer of the given index available.
◆	IsImported (see page 1321)	Function returns True if a layer with the given index is imported with the background to the project.
◆	SetImported (see page 1321)	Function allows for changing the status for the layer of the given index.

**IV.1.3.1 FindName****C++**

```
HRESULT FindName(BSTR _layer_name, long* ret);
```

**C#**

```
public long FindName(String _layer_name);
```

**Visual Basic**

```
Public Function FindName(_layer_name As String) As long
```

**Description**

Function returns index of the layer of the given name.

**Version**

Available since version 11.

**IV.1.3.2 Get****C++**

```
HRESULT Get(long _layer_idx, BSTR _layer_name, bool* _is_imported);
```

**C#**

```
public void Get(long _layer_idx, String _layer_name, bool* _is_imported);
```

**Visual Basic**

```
Public Sub Get(_layer_idx As long, ByRef _layer_name As String, ByRef _is_imported As bool*)
```

**Description**

Function makes data for a layer of the given index available.

**Version**

Available since version 11.

**IV.1.3.3 IsImported****C++**

```
HRESULT IsImported(long _layer_idx, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsImported(long _layer_idx);
```

**Visual Basic**

```
Public Function IsImported(_layer_idx As long) As Boolean
```

**Description**

Function returns True if a layer with the given index is imported with the background to the project.

**Version**

Available since version 11.

**IV.1.3.4 SetImported****C++**

```
HRESULT SetImported(long _layer_idx, VARIANT_BOOL _is_imported);
```

**C#**

```
public void SetImported(long _layer_idx, bool _is_imported);
```

**Visual Basic**

```
Public Sub SetImported(_layer_idx As long, _is_imported As Boolean)
```

**Description**

Function allows for changing the status for the layer of the given index.

**Version**

Available since version 11.

## IV.2 IRobotBackgroundInsertParams

**Class Hierarchy****C++**

```
interface IRobotBackgroundInsertParams : IDispatch;
```

**C#**

```
public interface IRobotBackgroundInsertParams;
```

**Visual Basic**

```
Public Interface IRobotBackgroundInsertParams
```

**Description**

Parameters of locating a background in the project.

**Version**

Available since version 11.

### IV.2.1 IRobotBackgroundInsertParams Members

The following tables list the members exposed by IRobotBackgroundInsertParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	InsertionPoint ( <a href="#">see page 1323</a> )	A point on the plane in which the characteristic point of the background will be located.
❖	Layers ( <a href="#">see page 1323</a> )	Background layers.
❖	Mirror ( <a href="#">see page 1323</a> )	Background mirror.
❖	Plane ( <a href="#">see page 1324</a> )	Plane of background insertion.
❖	PositionOnNormalAxis ( <a href="#">see page 1324</a> )	Location of the selected plane on the axis perpendicular to it.
❖	ReferencePoint ( <a href="#">see page 1324</a> )	Characteristic point on the background.
❖	RotationAngle ( <a href="#">see page 1324</a> )	
❖	ScalingFactor ( <a href="#">see page 1325</a> )	Scale factor.
❖	Units ( <a href="#">see page 1325</a> )	

### IV.2.2 IRobotBackgroundInsertParams Fields

The fields of the IRobotBackgroundInsertParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	InsertionPoint ( <a href="#">see page 1323</a> )	A point on the plane in which the characteristic point of the background will be located.

Layers (see page 1323)	Background layers.
Mirror (see page 1323)	Background mirror.
Plane (see page 1324)	Plane of background insertion.
PositionOnNormalAxis (see page 1324)	Location of the selected plane on the axis perpendicular to it.
ReferencePoint (see page 1324)	Characteristic point on the background.
RotationAngle (see page 1324)	
ScalingFactor (see page 1325)	Scale factor.
Units (see page 1325)	

#### IV.2.2.1 InsertionPoint

##### C++

```
HRESULT get_InsertionPoint(IRobotGeoPoint2D**);
```

##### C#

```
public IRobotGeoPoint2D InsertionPoint { get; }
```

##### Visual Basic

```
Public ReadOnly InsertionPoint As IRobotGeoPoint2D
```

##### Description

A point on the plane in which the characteristic point of the background will be located.

##### Version

Available since version 11.

#### IV.2.2.2 Layers

##### C++

```
HRESULT get_Layers(IRobotBackgroundLayers**);
```

##### C#

```
public IRobotBackgroundLayers Layers { get; }
```

##### Visual Basic

```
Public ReadOnly Layers As IRobotBackgroundLayers
```

##### Description

Background layers.

##### Version

Available since version 11.

#### IV.2.2.3 Mirror

##### C++

```
HRESULT get_Mirror(VARIANT_BOOL*);  
HRESULT put_Mirror(VARIANT_BOOL);
```

##### C#

```
public bool Mirror { get; set; }
```

##### Visual Basic

```
Public Mirror As Boolean
```

**Description**

Background mirror.

**Version**

Available since version 11.

**IV.2.2.4 Plane****C++**

```
HRESULT get_Plane(IRobotGeoPlane*);  
HRESULT put_Plane(IRobotGeoPlane);
```

**C#**

```
public IRobotGeoPlane Plane { get; set; }
```

**Visual Basic**

```
Public Plane As IRobotGeoPlane
```

**Description**

Plane of background insertion.

**Version**

Available since version 11.

**IV.2.2.5 PositionOnNormalAxis****C++**

```
HRESULT get_PositionOnNormalAxis(double*);  
HRESULT put_PositionOnNormalAxis(double);
```

**C#**

```
public double PositionOnNormalAxis { get; set; }
```

**Visual Basic**

```
Public PositionOnNormalAxis As Double
```

**Description**

Location of the selected plane on the axis perpendicular to it.

**Version**

Available since version 11.

**IV.2.2.6 ReferencePoint****C++**

```
HRESULT get_ReferencePoint(IRobotGeoPoint2D**);
```

**C#**

```
public IRobotGeoPoint2D ReferencePoint { get; }
```

**Visual Basic**

```
Public ReadOnly ReferencePoint As IRobotGeoPoint2D
```

**Description**

Characteristic point on the background.

**Version**

Available since version 11.

**IV.2.2.7 RotationAngle****C++**

```
HRESULT get_RotationAngle(double*);  
HRESULT put_RotationAngle(double);
```

**C#**

```
public double RotationAngle { get; set; }
```

**Visual Basic**

```
Public RotationAngle As Double
```

**Version**

Available since version 11.

**IV.2.2.8 ScalingFactor****C++**

```
HRESULT get_ScalingFactor(double*);  
HRESULT put_ScalingFactor(double);
```

**C#**

```
public double ScalingFactor { get; set; }
```

**Visual Basic**

```
Public ScalingFactor As Double
```

**Description**

Scale factor.

**Version**

Available since version 11.

**IV.2.2.9 Units****C++**

```
HRESULT get_Units(BSTR*);  
HRESULT put_Units(BSTR);
```

**C#**

```
public String Units { get; set; }
```

**Visual Basic**

```
Public Units As String
```

**Version**

Available since version 11.

**IV.3 IRobotBackground****Class Hierarchy****C++**

```
interface IRobotBackground : IDispatch;
```

**C#**

```
public interface IRobotBackground;
```

**Visual Basic**

```
Public Interface IRobotBackground
```

**Version**

Available since version 11.

**IV.3.1 IRobotBackground Members**

The following tables list the members exposed by IRobotBackground.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Color (see page 1326)	
◆	FilePath (see page 1327)	Path to a file with definition of background imported to project as a link.
◆	InsertParams (see page 1327)	Parameters of placing a background in the project.
◆	IsOn (see page 1327)	Flag switching on/off background displaying.
◆	Name (see page 1327)	
◆	Number (see page 1328)	Unique number identifying background.
◆	VisibilityRange (see page 1328)	Visibility range.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Save (see page 1328)	Function saves changed background definition to the project.

**IV.3.2 IRobotBackground Fields**

The fields of the IRobotBackground class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Color (see page 1326)	
◆	FilePath (see page 1327)	Path to a file with definition of background imported to project as a link.
◆	InsertParams (see page 1327)	Parameters of placing a background in the project.
◆	IsOn (see page 1327)	Flag switching on/off background displaying.
◆	Name (see page 1327)	
◆	Number (see page 1328)	Unique number identifying background.
◆	VisibilityRange (see page 1328)	Visibility range.

**IV.3.2.1 Color****C++**

```
HRESULT get_Color(long* );
HRESULT put_Color(long);
```

**C#**

```
public long Color { get; set; }
```

**Visual Basic**

```
Public Color As long
```

**Version**

Available since version 11.

**IV.3.2.2 FilePath****C++**

```
HRESULT get_FilePath(BSTR* );
HRESULT put_FilePath(BSTR);
```

**C#**

```
public String FilePath { get; set; }
```

**Visual Basic**

```
Public FilePath As String
```

**Description**

Path to a file with definition of background imported to project as a link.

**Version**

Available since version 11.

**IV.3.2.3 InsertParams****C++**

```
HRESULT get_InsertParams(IRobotBackgroundInsertParams** );
```

**C#**

```
public IRobotBackgroundInsertParams InsertParams { get; }
```

**Visual Basic**

```
Public ReadOnly InsertParams As IRobotBackgroundInsertParams
```

**Description**

Parameters of placing a background in the project.

**Version**

Available since version 11.

**IV.3.2.4 IsOn****C++**

```
HRESULT get_IsOn(VARIANT_BOOL* );
HRESULT put_IsOn(VARIANT_BOOL);
```

**C#**

```
public bool IsOn { get; set; }
```

**Visual Basic**

```
Public IsOn As Boolean
```

**Description**

Flag switching on/off background displaying.

**Version**

Available since version 11.

#### IV.3.2.5 Name

##### C++

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

##### C#

```
public String Name { get; set; }
```

##### Visual Basic

```
Public Name As String
```

##### Version

Available since version 11.

#### IV.3.2.6 Number

##### C++

```
HRESULT get_Number(long* );
```

##### C#

```
public long Number { get; }
```

##### Visual Basic

```
Public ReadOnly Number As long
```

##### Description

Unique number identifying background.

##### Version

Available since version 11.

#### IV.3.2.7 VisibilityRange

##### C++

```
HRESULT get_VisibilityRange(IRobotBackgroundVisibilityRange** );
```

##### C#

```
public IRobotBackgroundVisibilityRange VisibilityRange { get; }
```

##### Visual Basic

```
Public ReadOnly VisibilityRange As IRobotBackgroundVisibilityRange
```

##### Description

Visibility range.

##### Version

Available since version 11.

### IV.3.3 IRobotBackground Methods

The methods of the IRobotBackground class are listed here.

#### Public Methods

	Name	Description
	Save (see page 1328)	Function saves changed background definition to the project.

### IV.3.3.1 Save

**C++**

```
HRESULT Save(VARIANT_BOOL* ret);
```

**C#**

```
public bool Save();
```

**Visual Basic**

```
Public Function Save() As Boolean
```

#### Description

Function saves changed background definition to the project.

#### Version

Available since version 11.

## IV.4 IRobotBackgroundVisibilityRangeType

**C++**

```
enum IRobotBackgroundVisibilityRangeType;
```

**C#**

```
public enum IRobotBackgroundVisibilityRangeType;
```

**Visual Basic**

```
Public Enum IRobotBackgroundVisibilityRangeType
```

#### Members

Members	Description
I_BVRT_AUTOMATIC = 1	Available since version 11.
I_BVRT_USER_DEFINED = 2	Available since version 11.
I_BVRT_UNLIMITED = 3	Available since version 11.

#### Description

Available types of background visibility range.

#### Version

Available since version 11.

## IV.5 IRobotBackgroundVisibilityRange

#### Class Hierarchy

**C++**

```
interface IRobotBackgroundVisibilityRange : IDispatch;
```

**C#**

```
public interface IRobotBackgroundVisibilityRange;
```

**Visual Basic**

```
Public Interface IRobotBackgroundVisibilityRange
```

**Version**

Available since version 11.

**IV.5.1 IRobotBackgroundVisibilityRange Members**

The following tables list the members exposed by IRobotBackgroundVisibilityRange.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	FromPos ( <a href="#">see page 1330</a> )	
❖	ToPos ( <a href="#">see page 1330</a> )	
❖	Type ( <a href="#">see page 1330</a> )	

**IV.5.2 IRobotBackgroundVisibilityRange Fields**

The fields of the IRobotBackgroundVisibilityRange class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	FromPos ( <a href="#">see page 1330</a> )	
❖	ToPos ( <a href="#">see page 1330</a> )	
❖	Type ( <a href="#">see page 1330</a> )	

**IV.5.2.1 FromPos****C++**

```
HRESULT get_FromPos(double* );
HRESULT put_FromPos(double);
```

**C#**

```
public double FromPos { get; set; }
```

**Visual Basic**

```
Public FromPos As Double
```

**Version**

Available since version 11.

**IV.5.2.2 ToPos****C++**

```
HRESULT get_ToPos(double* );
HRESULT put_ToPos(double);
```

**C#**

```
public double ToPos { get; set; }
```

**Visual Basic**

```
Public ToPos As Double
```

**Version**

Available since version 11.

### IV.5.2.3 Type

#### C++

```
HRESULT get_Type(IRobotBackgroundVisibilityRangeType* );
HRESULT put_Type(IRobotBackgroundVisibilityRangeType);
```

#### C#

```
public IRobotBackgroundVisibilityRangeType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRobotBackgroundVisibilityRangeType
```

#### Version

Available since version 11.

## IV.6 IRobotBackgroundServer

#### Class Hierarchy

#### C++

```
interface IRobotBackgroundServer : IDispatch;
```

#### C#

```
public interface IRobotBackgroundServer;
```

#### Visual Basic

```
Public Interface IRobotBackgroundServer
```

#### Description

Object managing backgrounds in the project.

#### Version

Available since version 11.

### IV.6.1 IRobotBackgroundServer Members

The following tables list the members exposed by IRobotBackgroundServer.

#### Public Methods

	Name	Description
💡	Create (↗ see page 1332)	Function creates an object defining background on the basis of the selected file.
💡	Get (↗ see page 1332)	Function returns definition of the background of the given number.
💡	GetAllNumbers (↗ see page 1332)	Function returns table containing numbers of all backgrounds defined in the project.
💡	Remove (↗ see page 1333)	Function returns the background of the given number.
💡	RemoveAll (↗ see page 1333)	Function deletes all backgrounds from project.

### IV.6.2 IRobotBackgroundServer Methods

The methods of the IRobotBackgroundServer class are listed here.

## Public Methods

	Name	Description
💡	Create (🔗 see page 1332)	Function creates an object defining background on the basis of the selected file.
💡	Get (🔗 see page 1332)	Function returns definition of the background of the given number.
💡	GetAllNumbers (🔗 see page 1332)	Function returns table containing numbers of all backgrounds defined in the project.
💡	Remove (🔗 see page 1333)	Function returns the background of the given number.
💡	RemoveAll (🔗 see page 1333)	Function deletes all backgrounds from project.

### IV.6.2.1 Create

#### C++

```
HRESULT Create(BSTR _file_path, VARIANT_BOOL _as_link, IRobotBackground** ret);
```

#### C#

```
public IRobotBackground Create(String _file_path, bool _as_link);
```

#### Visual Basic

```
Public Function Create(_file_path As String, _as_link As Boolean) As IRobotBackground
```

#### Description

Function creates an object defining background on the basis of the selected file.

#### Version

Available since version 11.

### IV.6.2.2 Get

#### C++

```
HRESULT Get(long _number, IRobotBackground** ret);
```

#### C#

```
public IRobotBackground Get(long _number);
```

#### Visual Basic

```
Public Function Get(_number As long) As IRobotBackground
```

#### Description

Function returns definition of the background of the given number.

#### Version

Available since version 11.

### IV.6.2.3 GetAllNumbers

#### C++

```
HRESULT GetAllNumbers(SAFEARRAY** ret);
```

#### C#

```
public Array GetAllNumbers();
```

**Visual Basic**

```
Public Function GetAllNumbers() As Long()
```

**Description**

Function returns table containing numbers of all backgrounds defined in the project.

**Version**

Available since version 11.

**IV.6.2.4 Remove****C++**

```
HRESULT Remove(long _number);
```

**C#**

```
public void Remove(long _number);
```

**Visual Basic**

```
Public Sub Remove(_number As Long)
```

**Description**

Function returns the background of the given number.

**Version**

Available since version 11.

**IV.6.2.5 RemoveAll****C++**

```
HRESULT RemoveAll();
```

**C#**

```
public void RemoveAll();
```

**Visual Basic**

```
Public Sub RemoveAll()
```

**Description**

Function deletes all backgrounds from project.

**Version**

Available since version 11.

**V IRobotProjectType****C++**

```
enum IRobotProjectType;
```

**C#**

```
public enum IRobotProjectType;
```

**Visual Basic**

```
Public Enum IRobotProjectType
```

## Members

Members	Description
I_PT_CONCRETE_DEEP_BEAM = 18	Deep beam. Available since version 2.5.
I_PT_BUILDING = 19	Available since version 9.5.

## Description

A set of identifiers is defined to refer to structure types recognized by Robot.

# VI IRobotProject

## Class Hierarchy

### C++

```
interface IRobotProject : IDispatch;
```

### C#

```
public interface IRobotProject;
```

## Visual Basic

```
Public Interface IRobotProject
```

## Description

RobotProject interface allows for managing all the data saved in an RTD-format file. .

## VI.1 IRobotProject Members

The following tables list the members exposed by IRobotProject.

### Public Fields

	Name	Description
◆	ActiveModel ( [ see page 1336) )	.
◆	AxisMngr ( [ see page 1337) )	
◆	Backgrounds ( [ see page 1337) )	Backgrounds.
◆	CalcEngine ( [ see page 1337) )	Object representing the calculation module of Robot .
◆	ComponentMngr ( [ see page 1337) )	Manager of all project components .
◆	ConcrReinfEngine ( [ see page 1338) )	Calculation module for the concrete reinforcement.
◆	Connections ( [ see page 1338) )	Server of connections defined in the project. .
◆	DimServer ( [ see page 1338) )	Design and verification server for steel and timber members.
◆	ExtFileName ( [ see page 1338) )	The path to the last external file (read by means of the OpenExtFile ( [ see page 1344) function) in a non-RTD format.
◆	ExtFileParams ( [ see page 1339) )	Set of parameters opened by means of ReadExtFileParams ( [ see page 1344)() function from an external-format file .
◆	FileInsertParams ( [ see page 1339) )	Collection of parameter related to the manner of inserting into the project the currently opened external file .
◆	FileName ( [ see page 1339) )	Path to the file where the project is saved .
◆	IsActive ( [ see page 1339) )	A flag indicating if the project is active or not - opened from a file or defined by the New ( [ see page 1343)() function. It can happen that in the Robot program there is no project opened. In such a situation the IsActive flag equals zero (False).
◆	Name ( [ see page 1340) )	Project name .

◆	Preferences ( <a href="#">see page 1340</a> )	The object describing job parameters .
◆	PrintEngine ( <a href="#">see page 1340</a> )	Module that prints project documentation .
◆	Structure ( <a href="#">see page 1341</a> )	The object representing a structure defined in the project .
◆	Type ( <a href="#">see page 1341</a> )	Project type (type of structure defined in the project) .
◆	Uniqueld ( <a href="#">see page 1341</a> )	Unique string of characters enabling project identification (RTD file) independently of the file name and its location on disk, which may change while working with the project; (available since version 1.7).
◆	ViewMngr ( <a href="#">see page 1341</a> )	Manager controlling all views related to the project .

## Public Methods

	Name	Description
◆	Close ( <a href="#">see page 1342</a> )	The function closes the current project without saving changes. .
◆	InsertExtFile ( <a href="#">see page 1343</a> )	The function inserts a structure defined in the external format (non-RTD format) file to the current project. Values of parameters describing definition and method of structure insertion may be read out (by means of the ReadExtFileParams ( <a href="#">see page 1344</a> ) function) and changed earlier. If the structure is inserted correctly to the current project, then the function returns value different from zero (True).
◆	New ( <a href="#">see page 1343</a> )	The function creates a new project of the indicated type. .
◆	NewFromTemplate ( <a href="#">see page 1343</a> )	Function creates a new document on the basis of a template defined by a specified file. Available since version 2.0.
◆	Open ( <a href="#">see page 1344</a> )	The function opens an existing project from the indicated file in RTD format. .
◆	OpenExtFile ( <a href="#">see page 1344</a> )	The function opens a file in a format different from RTD. When opening the file, the program uses the current values of parameters read earlier from the file by means of the ReadExtFileParams ( <a href="#">see page 1344</a> ) function. The parameters are divided into two sets: FileInsertParams ( <a href="#">see page 1339</a> ) - parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams ( <a href="#">see page 1339</a> ) - set of variables defined in the external file to provide a parametrized structure description. The function returns a non-zero value (True) if the file is opened correctly and inserted into the project. It returns a zero value... more ( <a href="#">see page 1344</a> )
◆	ReadExtFileParams ( <a href="#">see page 1344</a> )	The function reads variables from the indicated external file in a non-RTD format. The values of the variables will be used during the process of opening the file by means of the OpenExtFile ( <a href="#">see page 1344</a> ) function. Once the variables are read by means of the function, two sets of parameters are available: FileInsertParams ( <a href="#">see page 1339</a> ) - parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams ( <a href="#">see page 1339</a> ) - set of variables defined in the external file to provide a parametrized structure description. The value of each parameter from each set may be modified and the modification... more ( <a href="#">see page 1344</a> )
◆	Save ( <a href="#">see page 1345</a> )	The function saves the current project to the file whose full access path is described by the FileName ( <a href="#">see page 1339</a> ) variable. .
◆	SaveAs ( <a href="#">see page 1345</a> )	The function saves the current project to the file accessed by the indicated path. .
◆	SaveAsExtFile ( <a href="#">see page 1345</a> )	The function saves the current project to the file in the indicated external format (other than RTD). .
◆	SaveToFormat ( <a href="#">see page 1345</a> )	Function saves the project to a file of the selected format. .

## VI.2 IRobotProject Fields

The fields of the IRobotProject class are listed here.

## Public Fields

	Name	Description
◆	ActiveModel ( [ see page 1336) ]	.
◆	AxisMngr ( [ see page 1337) ]	
◆	Backgrounds ( [ see page 1337) ]	Backgrounds.
◆	CalcEngine ( [ see page 1337) ]	Object representing the calculation module of Robot .
◆	ComponentMngr ( [ see page 1337) ]	Manager of all project components .
◆	ConcrReinfEngine ( [ see page 1338) ]	Calculation module for the concrete reinforcement.
◆	Connections ( [ see page 1338) ]	Server of connections defined in the project .
◆	DimServer ( [ see page 1338) ]	Design and verification server for steel and timber members.
◆	ExtFileName ( [ see page 1338) ]	The path to the last external file (read by means of the OpenExtFile ( [ see page 1344) ) function in a non-RTD format.
◆	ExtFileParams ( [ see page 1339) ]	Set of parameters opened by means of ReadExtFileParams ( [ see page 1344) () function from an external-format file .
◆	FileInsertParams ( [ see page 1339) ]	Collection of parameter related to the manner of inserting into the project the currently opened external file .
◆	FileName ( [ see page 1339) ]	Path to the file where the project is saved .
◆	IsActive ( [ see page 1339) ]	A flag indicating if the project is active or not - opened from a file or defined by the New ( [ see page 1343) () function. It can happen that in the Robot program there is no project opened. In such a situation the IsActive flag equals zero (False).
◆	Name ( [ see page 1340) ]	Project name .
◆	Preferences ( [ see page 1340) ]	The object describing job parameters .
◆	PrintEngine ( [ see page 1340) ]	Module that prints project documentation .
◆	Structure ( [ see page 1341) ]	The object representing a structure defined in the project .
◆	Type ( [ see page 1341) ]	Project type (type of structure defined in the project) .
◆	UniqueID ( [ see page 1341) ]	Unique string of characters enabling project identification (RTD file) independently of the file name and its location on disk, which may change while working with the project; (available since version 1.7).
◆	ViewMngr ( [ see page 1341) ]	Manager controlling all views related to the project .

## VI.2.1 ActiveModel

### C++

```
HRESULT get_ActiveModel( IRobotActiveModelType* );
HRESULT put_ActiveModel( IRobotActiveModelType );
```

### C#

```
public IRobotActiveModelType ActiveModel { get; set; }
```

### Visual Basic

```
Public ActiveModel As IRobotActiveModelType
```

### Description

### Version

Available since version 15.

## VI.2.2 AxisMngr

### C++

```
HRESULT get_AxisMngr( IRobotStructuralAxisGridMngr** );
```

**C#**

```
public IRobotStructuralAxisGridMngr AxisMngr { get; }
```

**Visual Basic**

```
Public ReadOnly AxisMngr As IRobotStructuralAxisGridMngr
```

## VI.2.3 Backgrounds

**C++**

```
HRESULT get_Backgrounds(IRobotBackgroundServer**);
```

**C#**

```
public IRobotBackgroundServer Backgrounds { get; }
```

**Visual Basic**

```
Public ReadOnly Backgrounds As IRobotBackgroundServer
```

**Description**

Backgrounds.

**Version**

Available since version 11.

## VI.2.4 CalcEngine

**C++**

```
HRESULT get_CalcEngine(IRobotCalcEngine**);
```

**C#**

```
public IRobotCalcEngine CalcEngine { get; }
```

**Visual Basic**

```
Public ReadOnly CalcEngine As IRobotCalcEngine
```

**Description**

Object representing the calculation module of Robot .

## VI.2.5 ComponentMngr

**C++**

```
HRESULT get_ComponentMngr(IRobotProjectComponentMngr**);
```

**C#**

```
public IRobotProjectComponentMngr ComponentMngr { get; }
```

**Visual Basic**

```
Public ReadOnly ComponentMngr As IRobotProjectComponentMngr
```

**Description**

Manager of all project components .

## VI.2.6 ConcrReinfEngine

**C++**

```
HRESULT get_ConcrReinfEngine(IRConcrCalcEngine**);
```

**C#**

```
public IRConcrCalcEngine ConcrReinfEngine { get; }
```

**Visual Basic**

```
Public ReadOnly ConcrReinfEngine As IRConcrCalcEngine
```

**Description**

Calculation module for the concrete reinforcement.

**Version**

Available since version 5.5.

## VI.2.7 Connections

**C++**

```
HRESULT get_Connections(IRJointConnectionServer**);
```

**C#**

```
public IRJointConnectionServer Connections { get; }
```

**Visual Basic**

```
Public ReadOnly Connections As IRJointConnectionServer
```

**Description**

Server of connections defined in the project. .

## VI.2.8 DimServer

**C++**

```
HRESULT get_DimServer(IRDimServer**);
```

**C#**

```
public IRDimServer DimServer { get; }
```

**Visual Basic**

```
Public ReadOnly DimServer As IRDimServer
```

**Description**

Design and verification server for steel and timber members.

**Version**

Available since version 2.5.

## VI.2.9 ExtFileName

**C++**

```
HRESULT get_ExtFileName(BSTR*);
```

**C#**

```
public String ExtFileName { get; }
```

**Visual Basic**

```
Public ReadOnly ExtFileName As String
```

**Description**

The path to the last external file (read by means of the OpenExtFile (see page 1344) function) in a non-RTD format.

## VI.2.10 ExtFileParams

### C++

```
HRESULT get_ExtFileParams(IDispatch*);
```

### C#

```
public IDispatch ExtFileParams { get; }
```

### Visual Basic

```
Public ReadOnly ExtFileParams As IDispatch
```

### Description

Set of parameters opened by means of ReadExtFileParams (see page 1344)() function from an external-format file .

## VI.2.11 FileInsertParams

### C++

```
HRESULT get_FileInsertParams(IRobotFileInsertParams**);
```

### C#

```
public IRobotFileInsertParams FileInsertParams { get; }
```

### Visual Basic

```
Public ReadOnly FileInsertParams As IRobotFileInsertParams
```

### Description

Collection of parameter related to the manner of inserting into the project the currently opened external file .

## VI.2.12 FileName

### C++

```
HRESULT get_FileName(BSTR*);
```

### C#

```
public String FileName { get; }
```

### Visual Basic

```
Public ReadOnly FileName As String
```

### Description

Path to the file where the project is saved .

## VI.2.13 IsActive

### C++

```
HRESULT get_IsActive(VARIANT_BOOL*);
```

### C#

```
public bool IsActive { get; }
```

### Visual Basic

```
Public ReadOnly IsActive As Boolean
```

### Description

A flag indicating if the project is active or not - opened from a file or defined by the New (see page 1343)() function. It can happen that in the Robot program there is no project opened. In such a situation the IsActive flag equals zero (False).

## VI.2.14 Name

**C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**Description**

Project name .

## VI.2.15 Preferences

**C++**

```
HRESULT get_Preferences(IRobotProjectPreferences**);
```

**C#**

```
public IRobotProjectPreferences Preferences { get; }
```

**Visual Basic**

```
Public ReadOnly Preferences As IRobotProjectPreferences
```

**Description**

The object describing job parameters .

## VI.2.16 PrintEngine

**C++**

```
HRESULT get_PrintEngine(IRobotPrintEngine**);
```

**C#**

```
public IRobotPrintEngine PrintEngine { get; }
```

**Visual Basic**

```
Public ReadOnly PrintEngine As IRobotPrintEngine
```

**Description**

Module that prints project documentation .

## VI.2.17 Structure

**C++**

```
HRESULT get_Structure(IRobotStructure**);
```

**C#**

```
public IRobotStructure Structure { get; }
```

**Visual Basic**

```
Public ReadOnly Structure As IRobotStructure
```

**Description**

The object representing a structure defined in the project .

## VI.2.18 Type

### C++

```
HRESULT get_Type(IRobotProjectType*);
```

### C#

```
public IRobotProjectType Type { get; }
```

### Visual Basic

```
Public ReadOnly Type As IRobotProjectType
```

### Description

Project type (type of structure defined in the project).

## VI.2.19 UniqueId

### C++

```
HRESULT get_UniqueId(BSTR*);
```

### C#

```
public String UniqueId { get; }
```

### Visual Basic

```
Public ReadOnly UniqueId As String
```

### Description

Unique string of characters enabling project identification (RTD file) independently of the file name and its location on disk, which may change while working with the project; (available since version 1.7).

## VI.2.20 ViewMngr

### C++

```
HRESULT get_ViewMngr(IRobotViewMngr**);
```

### C#

```
public IRobotViewMngr ViewMngr { get; }
```

### Visual Basic

```
Public ReadOnly ViewMngr As IRobotViewMngr
```

### Description

Manager controlling all views related to the project.

## VI.3 IRobotProject Methods

The methods of the IRobotProject class are listed here.

### Public Methods

	Name	Description
✖	Close (see page 1342)	The function closes the current project without saving changes. .
✖	InsertExtFile (see page 1343)	The function inserts a structure defined in the external format (non-RTD format) file to the current project. Values of parameters describing definition and method of structure insertion may be read out (by means of the ReadExtFileParams (see page 1344) function) and changed earlier. If the structure is inserted correctly to the current project, then the function returns value different from zero (True).

	New ( <a href="#">see page 1343</a> )	The function creates a new project of the indicated type. .
	NewFromTemplate ( <a href="#">see page 1343</a> )	Function creates a new document on the basis of a template defined by a specified file. Available since version 2.0.
	Open ( <a href="#">see page 1344</a> )	The function opens an existing project from the indicated file in RTD format. .
	OpenExtFile ( <a href="#">see page 1344</a> )	The function opens a file in a format different from RTD. When opening the file, the program uses the current values of parameters read earlier from the file by means of the ReadExtFileParams ( <a href="#">see page 1344</a> ) function. The parameters are divided into two sets: FileInsertParams ( <a href="#">see page 1339</a> ) - parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams ( <a href="#">see page 1339</a> ) - set of variables defined in the external file to provide a parametrized structure description. The function returns a non-zero value (True) if the file is opened correctly and inserted into the project. It returns a zero value... more ( <a href="#">see page 1344</a> )
	ReadExtFileParams ( <a href="#">see page 1344</a> )	The function reads variables from the indicated external file in a non-RTD format. The values of the variables will be used during the process of opening the file by means of the OpenExtFile ( <a href="#">see page 1344</a> ) function. Once the variables are read by means of the function, two sets of parameters are available: FileInsertParams ( <a href="#">see page 1339</a> ) - parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams ( <a href="#">see page 1339</a> ) - set of variables defined in the external file to provide a parametrized structure description. The value of each parameter from each set may be modified and the modification... more ( <a href="#">see page 1344</a> )
	Save ( <a href="#">see page 1345</a> )	The function saves the current project to the file whose full access path is described by the FileName ( <a href="#">see page 1339</a> ) variable. .
	SaveAs ( <a href="#">see page 1345</a> )	The function saves the current project to the file accessed by the indicated path. .
	SaveAsExtFile ( <a href="#">see page 1345</a> )	The function saves the current project to the file in the indicated external format (other than RTD). .
	SaveToFormat ( <a href="#">see page 1345</a> )	Function saves the project to a file of the selected format. .

### VI.3.1 Close

#### C++

```
HRESULT Close();
```

#### C#

```
public void Close();
```

#### Visual Basic

```
Public Sub Close()
```

#### Description

The function closes the current project without saving changes. .

### VI.3.2 InsertExtFile

#### C++

```
HRESULT InsertExtFile(BSTR _file_path, IRobotExternalFileFormat _file_format, VARIANT_BOOL _ignore_warnings, VARIANT_BOOL _only_geometry, VARIANT_BOOL* ret);
```

#### C#

```
public bool InsertExtFile(String _file_path, IRobotExternalFileFormat _file_format, bool _ignore_warnings, bool _only_geometry);
```

**Visual Basic**

```
Public Function InsertExtFile(_file_path As String, _file_format As
IRobotExternalFileFormat, _ignore_warnings As Boolean, _only_geometry As Boolean) As Boolean
```

**Description**

The function inserts a structure defined in the external format (non-RTD format) file to the current project. Values of parameters describing definition and method of structure insertion may be read out (by means of the ReadExtFileParams (see page 1344) function) and changed earlier. If the structure is inserted correctly to the current project, then the function returns value different from zero (True).

**Version**

Available since version 2.5.

**VI.3.3 New****C++**

```
HRESULT New(IRobotProjectType _prj_type);
```

**C#**

```
public void New(IRobotProjectType _prj_type);
```

**Visual Basic**

```
Public Sub New(_prj_type As IRobotProjectType)
```

**Description**

The function creates a new project of the indicated type. .

**VI.3.4 NewFromTemplate****C++**

```
HRESULT NewFromTemplate(BSTR _tmpl_file);
```

**C#**

```
public void NewFromTemplate(String _tmpl_file);
```

**Visual Basic**

```
Public Sub NewFromTemplate(_tmpl_file As String)
```

**Description**

Function creates a new document on the basis of a template defined by a specified file. Available since version 2.0.

**VI.3.5 Open****C++**

```
HRESULT Open(BSTR _file_path);
```

**C#**

```
public void Open(String _file_path);
```

**Visual Basic**

```
Public Sub Open(_file_path As String)
```

**Description**

The function opens an existing project from the indicated file in RTD format. .

### VI.3.6 OpenExtFile

**C++**

```
HRESULT OpenExtFile(BSTR _file_path, IRobotExternalFileFormat _format, VARIANT_BOOL  
_ignore_warnings, VARIANT_BOOL* ret);
```

**C#**

```
public bool OpenExtFile(String _file_path, IRobotExternalFileFormat _format, bool  
_ignore_warnings);
```

**Visual Basic**

```
Public Function OpenExtFile(_file_path As String, _format As IRobotExternalFileFormat,  
_ignore_warnings As Boolean) As Boolean
```

**Description**

The function opens a file in a format different from RTD. When opening the file, the program uses the current values of parameters read earlier from the file by means of the ReadExtFileParams (see page 1344) function. The parameters are divided into two sets: FileInsertParams (see page 1339)- parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams (see page 1339) - set of variables defined in the external file to provide a parametrized structure description. The function returns a non-zero value (True) if the file is opened correctly and inserted into the project. It returns a zero value (False) in the case of errors during file opening. .

### VI.3.7 ReadExtFileParams

**C++**

```
HRESULT ReadExtFileParams(BSTR _file_path, IRobotExternalFileFormat _file_format,  
VARIANT_BOOL* ret);
```

**C#**

```
public bool ReadExtFileParams(String _file_path, IRobotExternalFileFormat _file_format);
```

**Visual Basic**

```
Public Function ReadExtFileParams(_file_path As String, _file_format As  
IRobotExternalFileFormat) As Boolean
```

**Description**

The function reads variables from the indicated external file in a non-RTD format. The values of the variables will be used during the process of opening the file by means of the OpenExtFile (see page 1344) function. Once the variables are read by means of the function, two sets of parameters are available: FileInsertParams (see page 1339) - parameters describing the manner of inserting the structure defined in the external file to the current project, and ExtFileParams (see page 1339) - set of variables defined in the external file to provide a parametrized structure description. The value of each parameter from each set may be modified and the modification will be recognized during opening the file by means of the OpenExtFile (see page 1344) function. The function returns zero (False) in the case of errors during analysis of the indicated external file. .

### VI.3.8 Save

**C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Description**

The function saves the current project to the file whose full access path is described by the FileName (see page 1339) variable. .

**VI.3.9 SaveAs****C++**

```
HRESULT SaveAs(BSTR _file_path);
```

**C#**

```
public void SaveAs(String _file_path);
```

**Visual Basic**

```
Public Sub SaveAs(_file_path As String)
```

**Description**

The function saves the current project to the file accessed by the indicated path. .

**VI.3.10 SaveAsExtFile****C++**

```
HRESULT SaveAsExtFile(BSTR _file_path, IRobotExternalFileFormat _format);
```

**C#**

```
public void SaveAsExtFile(String _file_path, IRobotExternalFileFormat _format);
```

**Visual Basic**

```
Public Sub SaveAsExtFile(_file_path As String, _format As IRobotExternalFileFormat)
```

**Description**

The function saves the current project to the file in the indicated external format (other than RTD). .

**VI.3.11 SaveToFormat****C++**

```
HRESULT SaveToFormat(IRobotProjectSaveFormat _file_format, BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool SaveToFormat(IRobotProjectSaveFormat _file_format, String _file_path);
```

**Visual Basic**

```
Public Function SaveToFormat(_file_format As IRobotProjectSaveFormat, _file_path As String) As Boolean
```

**Description**

Function saves the project to a file of the selected format. .

**Version**

Available since version 3.

## VII IRobotFileInsertParams

### Class Hierarchy

#### C++

```
interface IRobotFileInsertParams : IDispatch;
```

#### C#

```
public interface IRobotFileInsertParams;
```

### Visual Basic

```
Public Interface IRobotFileInsertParams
```

### Description

The process of inserting a structure, defined in an external file of a non-RTD format, is carried out according to parameters determining the insertion mode. The RobotInsertParams interface provides access to the parameters determining the manner of inserting the structure into the current project. .

### VII.1 IRobotFileInsertParams Members

The following tables list the members exposed by IRobotFileInsertParams.

#### Public Fields

	Name	Description
❖	AsObject ( [ see page 1347 ] )	The flag indicating if the structure being inserted is to be treated as an object .
❖	ReferenceNode ( [ see page 1347 ] )	Node number - structure insertion point .
❖	ScaleFactor ( [ see page 1347 ] )	Scaling coefficient .

#### Public Methods

	Name	Description
❖	GetInsertionPoint ( [ see page 1348 ] )	The function sets the coordinates of the insertion point. .
❖	GetRotation ( [ see page 1348 ] )	The function provides access to the rotation angle defined for the inserted structure. .
❖	SetInsertionPoint ( [ see page 1348 ] )	The function sets the coordinates of the insertion point. .
❖	SetRotation ( [ see page 1349 ] )	The function allows one to determine the angle of the inserted structure. .

### VII.2 IRobotFileInsertParams Fields

The fields of the IRobotFileInsertParams class are listed here.

#### Public Fields

	Name	Description
❖	AsObject ( [ see page 1347 ] )	The flag indicating if the structure being inserted is to be treated as an object .
❖	ReferenceNode ( [ see page 1347 ] )	Node number - structure insertion point .
❖	ScaleFactor ( [ see page 1347 ] )	Scaling coefficient .

### VII.2.1 AsObject

#### C++

```
HRESULT get_AsObject(VARIANT_BOOL* );
```

```
HRESULT put_AsObject(VARIANT_BOOL);
```

**C#**

```
public bool AsObject { get; set; }
```

**Visual Basic**

```
Public AsObject As Boolean
```

**Description**

The flag indicating if the structure being inserted is to be treated as an object .

**VII.2.2 ReferenceNode****C++**

```
HRESULT get_ReferenceNode(long*);  
HRESULT put_ReferenceNode(long);
```

**C#**

```
public long ReferenceNode { get; set; }
```

**Visual Basic**

```
Public ReferenceNode As long
```

**Description**

Node number - structure insertion point .

**VII.2.3 ScaleFactor****C++**

```
HRESULT get_ScaleFactor(double*);  
HRESULT put_ScaleFactor(double);
```

**C#**

```
public double ScaleFactor { get; set; }
```

**Visual Basic**

```
Public ScaleFactor As double
```

**Description**

Scaling coefficient .

**VII.3 IRobotFileInsertParams Methods**

The methods of the IRobotFileInsertParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetInsertionPoint (🔗 see page 1348)	The function sets the coordinates of the insertion point. .
💡	GetRotation (🔗 see page 1348)	The function provides access to the rotation angle defined for the inserted structure. .
💡	SetInsertionPoint (🔗 see page 1348)	The function sets the coordinates of the insertion point. .
💡	SetRotation (🔗 see page 1349)	The function allows one to determine the angle of the inserted structure. .

### VII.3.1 GetInsertionPoint

C++

```
HRESULT GetInsertionPoint(double* _x, double* _y, double* _z);
```

C#

```
public void GetInsertionPoint(double* _x, double* _y, double* _z);
```

Visual Basic

```
Public Sub GetInsertionPoint(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

Description

The function sets the coordinates of the insertion point. .

### VII.3.2 GetRotation

C++

```
HRESULT GetRotation(double* _alpha, double* _beta, double* _gamma);
```

C#

```
public void GetRotation(double* _alpha, double* _beta, double* _gamma);
```

Visual Basic

```
Public Sub GetRotation(ByRef _alpha As double*, ByRef _beta As double*, ByRef _gamma As double*)
```

Description

The function provides access to the rotation angle defined for the inserted structure. .

### VII.3.3 SetInsertionPoint

C++

```
HRESULT SetInsertionPoint(double _x, double _y, double _z);
```

C#

```
public void SetInsertionPoint(double _x, double _y, double _z);
```

Visual Basic

```
Public Sub SetInsertionPoint(_x As double, _y As double, _z As double)
```

Description

The function sets the coordinates of the insertion point. .

### VII.3.4 SetRotation

C++

```
HRESULT SetRotation(double _alpha, double _beta, double _gamma);
```

C#

```
public void SetRotation(double _alpha, double _beta, double _gamma);
```

Visual Basic

```
Public Sub SetRotation(_alpha As double, _beta As double, _gamma As double)
```

## Description

The function allows one to determine the angle of the inserted structure. .

# VIII IRobotProjectSaveFormat

## C++

```
enum IRobotProjectSaveFormat;
```

## C#

```
public enum IRobotProjectSaveFormat;
```

## Visual Basic

```
Public Enum IRobotProjectSaveFormat
```

## Members

Members	Description
I_PSF_RTD = 0	Available since version 3.
I_PSF_RTD_NORESULTS = 1	Available since version 3.
I_PSF_STR = 2	Available since version 3.
I_PSF_DXF = 3	Available since version 3.
I_PSF_DXF_V14 = 4	Available since version 3.
I_PSF_DWG = 5	Available since version 3.
I_PSF_ANF = 6	Available since version 3.
I_PSF_WRL = 7	Available since version 3.
I_PSF_S = 8	Available since version 3.
I_PSF_SAT = 9	Available since version 3.
I_PSF_STP_CIM = 10	Available since version 3.
I_PSF_STP_DSTV = 11	Available since version 3.

## Description

Available formats for project saving.

## Version

Available since version 3.

# IX IRobotProjectEvents

## Class Hierarchy

## C++

```
interface IRobotProjectEvents : IDispatch;
```

## C#

```
public interface IRobotProjectEvents;
```

## Visual Basic

```
Public Interface IRobotProjectEvents
```

## Description

Events generated by the RobotProject object.

**Version**

Available since version 7.5.

## IX.1 IRobotProjectEvents Members

The following tables list the members exposed by IRobotProjectEvents.

**Public Methods**

	<b>Name</b>	<b>Description</b>
⽰	OnClose ( ⓘ see page 1350)	Event generated directly before closing the document.
⽰	OnSave ( ⓘ see page 1351)	Event generated directly before saving the project to the file of the specified path.

## IX.2 IRobotProjectEvents Methods

The methods of the IRobotProjectEvents class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
⽰	OnClose ( ⓘ see page 1350)	Event generated directly before closing the document.
⽰	OnSave ( ⓘ see page 1351)	Event generated directly before saving the project to the file of the specified path.

### IX.2.1 OnClose

**C++**

```
HRESULT OnClose(VARIANT_BOOL _is_modified);
```

**C#**

```
public void OnClose(bool _is_modified);
```

**Visual Basic**

```
Public Sub OnClose(_is_modified As Boolean)
```

**Description**

Event generated directly before closing the document.

**Version**

Available since version 7.5.

### IX.2.2 OnSave

**C++**

```
HRESULT OnSave(BSTR _file_path, VARIANT_BOOL _is_modified);
```

**C#**

```
public void OnSave(String _file_path, bool _is_modified);
```

**Visual Basic**

```
Public Sub OnSave(_file_path As String, _is_modified As Boolean)
```

**Description**

Event generated directly before saving the project to the file of the specified path.

**Version**

Available since version 7.5.

**X IRobotActiveModelType****C++**

```
enum IRobotActiveModelType;
```

**C#**

```
public enum IRobotActiveModelType;
```

**Visual Basic**

```
Public Enum IRobotActiveModelType
```

**Members**

Members	Description
I_AMT_MAIN = 0	.
I_AMT_DAM = 1	Available since version 15.

**Description****Version**

Available since version 15.

**Presentation of data****I Views and layouts****Enumerations**

	Name	Description
	IRobotLayoutId (see page 1468)	Set of identifiers for the layouts defined in the Robot program.. .
	IRobotDialogId (see page 1469)	Identifiers of available dialogs.

**Interfaces**

	Name	Description
	IRobotViewMngr (see page 1462)	View manager controls all the views and layouts available in Robot. .

**I.1 Tables****Enumerations**

	Name	Description
	IRobotTableType (see page 1352)	One can differentiate a number of table types presenting different types of data. Each table type has got its identifier.. .

	IRobotTableDataType ( <a href="#">see page 1354</a> )	A set of identifiers is defined to refer to types of data presented in tables. .
	IRobotTableConfigFlag ( <a href="#">see page 1355</a> )	Tables may be configured in many ways. A set of flags is defined to help one configure tables. .
	IRobotTableConfigValue ( <a href="#">see page 1356</a> )	Set of values used to configure tables. .
	IRobotTableScreenCaptureUpdateType ( <a href="#">see page 1372</a> )	

**Interfaces**

	Name	Description
	IRobotTableConfig ( <a href="#">see page 1356</a> )	Each table may be configured by means of a configuring object.
	IRobotTable ( <a href="#">see page 1357</a> )	Interface representing tables. .
	IRobotTableFrame ( <a href="#">see page 1366</a> )	Interface representing the viewer composed of many tables. Each table is a separate tab defined within the same main frame.
	IRobotTableScreenCaptureParams ( <a href="#">see page 1369</a> )	Parameters of the screen capture for a table.

**I.1.1 IRobotTableType****C++**

```
enum IRobotTableType;
```

**C#**

```
public enum IRobotTableType;
```

**Visual Basic**

```
Public Enum IRobotTableType
```

**Members**

Members	Description
I_TT_NODES = 1	Nodes table. Available since version 15.2.
I_TT_BARS = 2	Bars table. Available since version 15.2.
I_TT_PANELS = 3	.
I_TTFINITE_ELEMENTS = 4	.
I_TT_SUPPORTS = 5	.
I_TT_PROPERTIES = 6	.
I_TT_MEASURE = 7	.
I_TT_LOADS = 8	.
I_TT_COMBINATIONS = 9	.
I_TT_MASS = 10	.
	Available since version 15.2.

I_TT_REACTIONS = 11	.	Available since version 15.2.
I_TT_NODE_DISPLACEMENTS = 12	.	Available since version 15.2.
I_TT_BAR_DEFLECTIONS = 13	.	Available since version 15.2.
I_TT_FORCES = 14	.	Available since version 15.2.
I_TT_STRESSES = 15	.	Available since version 15.2.
I_TT_DYNAMIC = 16	.	Available since version 15.2.
I_TT_PSEUDOSTATIC = 17	.	Available since version 15.2.
I_TT_BUCKLING = 18	.	Available since version 15.2.
I_TT_FE_RESULTS = 19	.	Available since version 15.2.
I_TT_REINFORCEMENT = 20	.	Available since version 15.2.
I_TT_SUPPORTS_LABEL = 21	.	Available since version 15.2.
I_TT_OFFSETS = 22	.	Available since version 15.2.
I_TT_STEEL_MEMBERS = 23	.	Available since version 15.2.
I_TT_LIARIG_LABEL = 24	Rigid Link labels table. Available since version 15.2.	
I_TT_COREWALLS = 25	.	Available since version 15.2.
I_TT_INTERACTION_FORCE = 26	.	Available since version 15.2.
I_TT_PLASTIC_HISTORY = 27	.	Available since version 15.2.
I_TT_GLOBAL_ANALYSIS_BARS = 28	.	Available since version 15.2.
I_TT_INFLUENCE_LINE = 29	.	Available since version 15.2.
I_TT_STOREYS = 30	.	Available since version 15.2.

## Description

One can differentiate a number of table types presenting different types of data. Each table type has got its identifier. .

### I.1.2 IRobotTableDataType

#### C++

```
enum IRobotTableDataType;
```

#### C#

```
public enum IRobotTableDataType;
```

## Visual Basic

```
Public Enum IRobotTableDataType
```

### Members

Members	Description
I_TDT_DEFAULT = 1	.
	Available since version 15.2.
I_TDT_VALUES = 2	.
	Available since version 15.2.
I_TDT_INFO = 3	.
	Available since version 15.2.
I_TDT_ENVELOPE = 4	.
	Available since version 15.2.
I_TDT_EXTREMES = 5	.
	Available since version 15.2.
I_TDT_NODE = 6	.
	Available since version 15.2.
I_TDT_BAR = 7	.
	Available since version 15.2.
I_TDT_PANEL = 8	.
	Available since version 15.2.
I_TDT_FE = 9	.
	Available since version 15.2.
I_TDT_CABLE = 10	.
	Available since version 15.2.
I_TDT_BRACKET = 11	.
	Available since version 15.2.
I_TDT_MEMBERS = 12	.
	Available since version 15.2.
I_TDT_CODE_GROUPS = 13	.
	Available since version 15.2.
I_TDT_PARA = 14	.
	Available since version 15.2.
I_TDT_VOLUME = 15	.
	Available since version 15.2.
I_TDT_MATERIAL = 16	.
	Available since version 15.2.
I_TDT_DISPLACEMENTS = 17	.
	Available since version 15.2.
I_TDT_REDUCED_FORCES = 18	.
	Available since version 15.2.
I_TDT_TOTAL = 19	.
	Available since version 15.2.

### Description

A set of identifiers is defined to refer to types of data presented in tables. .

## I.1.3 IRobotTableConfigFlag

### C++

```
enum IRobotTableConfigFlag;
```

**C#**

```
public enum IRobotTableConfigFlag;
```

**Visual Basic**

```
Public Enum IRobotTableConfigFlag
```

**Members**

<b>Members</b>	<b>Description</b>
I_TCF_DETAILS = 1	Detailed information Available since version 1.7.
I_TCF_GLOBAL_COORDINATES = 2	Global coordinates Available since version 1.7.
I_TCF_LOCAL_COORDINATES = 3	Local coordinates Available since version 1.7.
I_TCF_COEXISTENT_VALUES = 4	Available since version 1.7.
I_TCF_EXTREME_COMBINATIONS = 5	Available since version 1.7.
I_TCF_PROP_STD_TAPERED_SECTION = 6	Available since version 1.7.
I_TCF_COMB_DEFINITIONS = 7	Available since version 1.7.
I_TCF_COMB_CODE_COMPONENTS = 8	Available since version 1.7.
I_TCF_COMB_MOVING_LOADS = 9	Available since version 1.7.
I_TCF_COMB_TIME_HISTORY_COMPONENTS = 10	Available since version 1.7.
I_TCF_REAC_VALUES = 11	Available since version 1.7.
I_TCF_REAC_APPLIED_FORCES_SUM = 12	Available since version 1.7.
I_TCF_REAC_EQUILIBRIUM_PRECISION = 13	Available since version 1.7.
I_TCF_REAC_RESIDUUM = 14	Available since version 1.7.
I_TCF_BAR_DEFLECTIONS = 15	Available since version 1.7.
I_TCF_BAR_DEFLECTIONS_MAX = 16	Available since version 1.7.
I_TCF_BAR_DEFLECTIONS_DISPLACEMENTS = 17	Available since version 1.7.
I_TCF_FE_DIR_X_AXIS = 18	Available since version 1.7.
I_TCF_FE_DIR_Y_AXIS = 19	Available since version 1.7.
I_TCF_FE_DIR_Z_AXIS = 20	Available since version 1.7.
I_TCF_FE_LAYER_UPPER = 21	Available since version 1.7.
I_TCF_FE_LAYER_MIDDLE = 22	Available since version 1.7.
I_TCF_FE_LAYER_LOWER = 23	Available since version 1.7.
I_TCF_FE_LAYER_MAX = 24	Available since version 1.7.
I_TCF_FE_LAYER_MIN = 25	Available since version 1.7.
I_TCF_FE_LAYER_MAX_ABSOLUTE = 26	Available since version 1.7.
I_TCF_FE_IN_NODES = 27	Available since version 1.7.
I_TCF_FE_IN_ELEMENT_CENTERS = 28	Available since version 1.7.
I_TCF_REINF_IN_NODES = 29	Available since version 1.7.
I_TCF_REINF_IN_ELEMENT_CENTERS = 30	Available since version 1.7.
I_TCF_REAC_DDC = 31	Available since version 1.7.
I_TCF_LOCALS = 32	Available since version 1.7.
I_TCF_FE_IN_ELENODE = 33	Available since version 1.7.
I_TCF_ORDER_BY_CASE = 34	Available since version 1.7.
I_TCF_CASE_INFO = 35	Available since version 1.7.
I_TCF_GLOBAL_ANALYSIS_RESULTS_NPOINTS = 36	Available since version 3.
I_TCF_GLOBAL_ANALYSIS_RESULTS_RELATIVE = 37	Available since version 3.

**Description**

Tables may be configured in many ways. A set of flags is defined to help one configure tables. .

## I.1.4 IRobotTableConfigValue

**C++**

```
enum IRobotTableConfigValue;
```

**C#**

```
public enum IRobotTableConfigValue;
```

**Visual Basic**

```
Public Enum IRobotTableConfigValue
```

**Description**

Set of values used to configure tables..

## I.1.5 IRobotTableConfig

**Class Hierarchy**

**C++**

```
interface IRobotTableConfig : IDispatch;
```

**C#**

```
public interface IRobotTableConfig;
```

**Visual Basic**

```
Public Interface IRobotTableConfig
```

**Description**

Each table may be configured by means of a configuring object.

### I.1.5.1 IRobotTableConfig Members

The following tables list the members exposed by IRobotTableConfig.

**Public Methods**

	Name	Description
💡	SetFlag (🔗 see page 1357)	The function switches {on/off} the indicated flag configuring the table. .
💡	SetValue (🔗 see page 1357)	The function sets the value of the indicated variable configuring the table. .

### I.1.5.2 IRobotTableConfig Methods

The methods of the IRobotTableConfig class are listed here.

**Public Methods**

	Name	Description
💡	SetFlag (🔗 see page 1357)	The function switches {on/off} the indicated flag configuring the table. .
💡	SetValue (🔗 see page 1357)	The function sets the value of the indicated variable configuring the table. .

#### I.1.5.2.1 SetFlag

**C++**

```
HRESULT SetFlag( IRobotTableConfigFlag _flag, VARIANT_BOOL _set);
```

**C#**

```
public void SetFlag(IRobotTableConfigFlag _flag, bool _set);
```

**Visual Basic**

```
Public Sub SetFlag(_flag As IRobotTableConfigFlag, _set As Boolean)
```

**Description**

The function switches {on/off} the indicated flag configuring the table. .

**I.1.5.2.2 SetValue****C++**

```
HRESULT SetValue(IRobotTableConfigValue _value_id, double _val);
```

**C#**

```
public void SetValue(IRobotTableConfigValue _value_id, double _val);
```

**Visual Basic**

```
Public Sub SetValue(_value_id As IRobotTableConfigValue, _val As Double)
```

**Description**

The function sets the value of the indicated variable configuring the table. .

**I.1.6 IRobotTable****Class Hierarchy****C++**

```
interface IRobotTable : IDispatch;
```

**C#**

```
public interface IRobotTable;
```

**Visual Basic**

```
Public Interface IRobotTable
```

**Description**

Interface representing tables. .

**I.1.6.1 IRobotTable Members**

The following tables list the members exposed by IRobotTable.

**Public Fields**

	Name	Description
◆	ColCount (↗ see page 1359)	Number of table columns.
◆	Configuration (↗ see page 1359)	Object controlling table configuration .
◆	Printable (↗ see page 1360)	Object representing tables; it may be added to the printout and printed .
◆	RowCount (↗ see page 1360)	Number of table rows.
◆	Title (↗ see page 1360)	Table title .
◆	UserControl (↗ see page 1360)	Flag that enables giving/taking back table navigation to/from the interactive program user .
◆	Visible (↗ see page 1361)	Variable determining if the table is to be displayed or not .
◆	Window (↗ see page 1361)	

## Public Methods

	Name	Description
◆	AddColumn (see page 1362)	The function adds to a table a new column that will present data of the indicated type .
◆	GetColWidth (see page 1362)	Function returns a width of the column with the specified index.
◆	GetDataType (see page 1362)	Function returns a type of data presented in the column with the specified index.
◆	GetRowHeight (see page 1363)	Function returns a height of the row with the specified index.
◆	MakeScreenCapture (see page 1363)	Function saves a screen capture with the specified parameters.
◆	Select (see page 1363)	The function sets a filter defined by the indicated selection on the table .
◆	SelectAllFromStore (see page 1364)	The function sets a filter on the table, defined by means of all selections formerly saved in the table by means of the StoreSelection (see page 1366)() function. .
◆	SelectFromStore (see page 1364)	The function sets a filter on the table. The filter is defined by means of the selection saved earlier by the StoreSelection (see page 1366)() function.
◆	SelectMode (see page 1364)	The function sets a filter on the table, described by the indicated mode selection .
◆	SelectModeFromStore (see page 1365)	The function sets a filter on the table, defined by means of mode selection formerly saved in the table by means of the StoreModeSelection (see page 1365)()..
◆	SetColWidth (see page 1365)	Function sets a width of the column with the specified index.
◆	SetRowHeight (see page 1365)	Function sets a height of the row with the specified index.
◆	StoreModeSelection (see page 1365)	The function saves the indicated mode selection to the table. .
◆	StoreSelection (see page 1366)	The function saves the indicated selection in the table without modifying its filter. One may save one selection of each type of objects - structure components - in the table. .

### I.1.6.2 IRobotTable Fields

The fields of the IRobotTable class are listed here.

## Public Fields

	Name	Description
◆	ColCount (see page 1359)	Number of table columns.
◆	Configuration (see page 1359)	Object controlling table configuration .
◆	Printable (see page 1360)	Object representing tables; it may be added to the printout and printed .
◆	RowCount (see page 1360)	Number of table rows.
◆	Title (see page 1360)	Table title .
◆	UserControl (see page 1360)	Flag that enables giving/taking back table navigation to/from the interactive program user .
◆	Visible (see page 1361)	Variable determining if the table is to be displayed or not .
◆	Window (see page 1361)	

#### I.1.6.2.1 ColCount

##### C++

```
HRESULT get_ColCount(long*);
```

##### C#

```
public long ColCount { get; }
```

**Visual Basic**

```
Public ReadOnly ColCount As long
```

**Description**

Number of table columns.

**Version**

Available since version 8.2.

### I.1.6.2.2 Configuration

**C++**

```
HRESULT get_Configuration(IRobotTableConfig**);
```

**C#**

```
public IRobotTableConfig Configuration { get; }
```

**Visual Basic**

```
Public ReadOnly Configuration As IRobotTableConfig
```

**Description**

Object controlling table configuration .

### I.1.6.2.3 Printable

**C++**

```
HRESULT get_Printable(IRobotPrintable**);
```

**C#**

```
public IRobotPrintable Printable { get; }
```

**Visual Basic**

```
Public ReadOnly Printable As IRobotPrintable
```

**Description**

Object representing tables; it may be added to the printout and printed .

### I.1.6.2.4 RowCount

**C++**

```
HRESULT get_RowCount(long*);
```

**C#**

```
public long RowCount { get; }
```

**Visual Basic**

```
Public ReadOnly RowCount As long
```

**Description**

Number of table rows.

**Version**

Available since version 8.2.

### I.1.6.2.5 Title

#### C++

```
HRESULT get_Title(BSTR* );
HRESULT put_Title(BSTR);
```

#### C#

```
public String Title { get; set; }
```

#### Visual Basic

```
Public Title As String
```

#### Description

Table title .

### I.1.6.2.6 UserControl

#### C++

```
HRESULT get_UserControl(VARIANT_BOOL* );
HRESULT put_UserControl(VARIANT_BOOL);
```

#### C#

```
public bool UserControl { get; set; }
```

#### Visual Basic

```
Public UserControl As Boolean
```

#### Description

Flag that enables giving/taking back table navigation to/from the interactive program user .

#### Version

Available since version 3.

### I.1.6.2.7 Visible

#### C++

```
HRESULT get_Visible(VARIANT_BOOL* );
HRESULT put_Visible(VARIANT_BOOL);
```

#### C#

```
public bool Visible { get; set; }
```

#### Visual Basic

```
Public Visible As Boolean
```

#### Description

Variable determining if the table is to be displayed or not .

### I.1.6.2.8 Window

#### C++

```
HRESULT get_Window(IRobotWindow** );
```

#### C#

```
public IRobotWindow Window { get; }
```

## Visual Basic

```
Public ReadOnly Window As IRobotWindow
```

### Version

Available since version 3.

#### I.1.6.3 IRobotTable Methods

The methods of the IRobotTable class are listed here.

##### Public Methods

	Name	Description
⊕	AddColumn ( [ see page 1362) )	The function adds to a table a new column that will present data of the indicated type .
⊕	GetColWidth ( [ see page 1362) )	Function returns a width of the column with the specified index.
⊕	GetDataType ( [ see page 1362) )	Function returns a type of data presented in the column with the specified index.
⊕	GetRowHeight ( [ see page 1363) )	Function returns a height of the row with the specified index.
⊕	MakeScreenCapture ( [ see page 1363) )	Function saves a screen capture with the specified parameters.
⊕	Select ( [ see page 1363) )	The function sets a filter defined by the indicated selection on the table .
⊕	SelectAllFromStore ( [ see page 1364) )	The function sets a filter on the table, defined by means of all selections formerly saved in the table by means of the StoreSelection ( [ see page 1366)() ) function. .
⊕	SelectFromStore ( [ see page 1364) )	The function sets a filter on the table. The filter is defined by means of the selection saved earlier by the StoreSelection ( [ see page 1366)() ) function.
⊕	SelectMode ( [ see page 1364) )	The function sets a filter on the table, described by the indicated mode selection. .
⊕	SelectModeFromStore ( [ see page 1365) )	The function sets a filter on the table, defined by means of mode selection formerly saved in the table by means of the StoreModeSelection ( [ see page 1365)() ) .
⊕	SetColWidth ( [ see page 1365) )	Function sets a width of the column with the specified index.
⊕	SetRowHeight ( [ see page 1365) )	Function sets a height of the row with the specified index.
⊕	StoreModeSelection ( [ see page 1365) )	The function saves the indicated mode selection to the table. .
⊕	StoreSelection ( [ see page 1366) )	The function saves the indicated selection in the table without modifying its filter. One may save one selection of each type of objects - structure components - in the table. .

#### I.1.6.3.1 AddColumn

##### C++

```
HRESULT AddColumn( IRobotTableDataType _data_type);
```

##### C#

```
public void AddColumn( IRobotTableDataType _data_type);
```

## Visual Basic

```
Public Sub AddColumn(_data_type As IRobotTableDataType)
```

### Description

The function adds to a table a new column that will present data of the indicated type .

### I.1.6.3.2 GetColWidth

#### C++

```
HRESULT GetColWidth(long _col_idx, long* ret);
```

#### C#

```
public long GetColWidth(long _col_idx);
```

#### Visual Basic

```
Public Function GetColWidth(_col_idx As long) As long
```

#### Description

Function returns a width of the column with the specified index.

#### Version

Available since version 8.2.

### I.1.6.3.3 GetDataType

#### C++

```
HRESULT GetDataType(long _col_idx, IRobotTableDataType* ret);
```

#### C#

```
public IRobotTableDataType GetDataType(long _col_idx);
```

#### Visual Basic

```
Public Function GetDataType(_col_idx As long) As IRobotTableDataType
```

#### Description

Function returns a type of data presented in the column with the specified index.

#### Version

Available since version 8.2.

### I.1.6.3.4 GetRowHeight

#### C++

```
HRESULT GetRowHeight(long _row_idx, long* ret);
```

#### C#

```
public long GetRowHeight(long _row_idx);
```

#### Visual Basic

```
Public Function GetRowHeight(_row_idx As long) As long
```

#### Description

Function returns a height of the row with the specified index.

#### Version

Available since version 8.2.

### I.1.6.3.5 MakeScreenCapture

**C++**

```
HRESULT MakeScreenCapture(IRobotTableScreenCaptureParams* _sc_params);
```

**C#**

```
public void MakeScreenCapture(IRobotTableScreenCaptureParams _sc_params);
```

**Visual Basic**

```
Public Sub MakeScreenCapture(ByRef _sc_params As IRobotTableScreenCaptureParams)
```

**Description**

Function saves a screen capture with the specified parameters.

**Version**

Available since version 8.2.

### I.1.6.3.6 Select

**C++**

```
HRESULT Select(IRobotObjectType _sel_type, BSTR _sel_text);
```

**C#**

```
public void Select(IRobotObjectType _sel_type, String _sel_text);
```

**Visual Basic**

```
Public Sub Select(_sel_type As IRobotObjectType, _sel_text As String)
```

**Description**

The function sets a filter defined by the indicated selection on the table .

### I.1.6.3.7 SelectAllFromStore

**C++**

```
HRESULT SelectAllFromStore();
```

**C#**

```
public void SelectAllFromStore();
```

**Visual Basic**

```
Public Sub SelectAllFromStore()
```

**Description**

The function sets a filter on the table, defined by means of all selections formerly saved in the table by means of the StoreSelection (see page 1366)() function. .

### I.1.6.3.8 SelectFromStore

**C++**

```
HRESULT SelectFromStore(IRobotObjectType _sel_type);
```

**C#**

```
public void SelectFromStore(IRobotObjectType _sel_type);
```

**Visual Basic**

```
Public Sub SelectFromStore(_sel_type As IRobotObjectType)
```

**Description**

The function sets a filter on the table. The filter is defined by means of the selection saved earlier by the StoreSelection (see page 1366)() function.

**I.1.6.3.9 SelectMode****C++**

```
HRESULT SelectMode(IRobotModeSelectionType _type, BSTR _sel_text, IRobotModeCombinationType _comb);
```

**C#**

```
public void SelectMode(IRobotModeSelectionType _type, String _sel_text,
IRobotModeCombinationType _comb);
```

**Visual Basic**

```
Public Sub SelectMode(_type As IRobotModeSelectionType, _sel_text As String, _comb As
IRobotModeCombinationType)
```

**Description**

The function sets a filter on the table, described by the indicated mode selection..

**I.1.6.3.10 SelectModeFromStore****C++**

```
HRESULT SelectModeFromStore();
```

**C#**

```
public void SelectModeFromStore();
```

**Visual Basic**

```
Public Sub SelectModeFromStore()
```

**Description**

The function sets a filter on the table, defined by means of mode selection formerly saved in the table by means of the StoreModeSelection (see page 1365)().

**I.1.6.3.11 SetColWidth****C++**

```
HRESULT SetColWidth(long _col_idx, long _col_width);
```

**C#**

```
public void SetColWidth(long _col_idx, long _col_width);
```

**Visual Basic**

```
Public Sub SetColWidth(_col_idx As long, _col_width As long)
```

**Description**

Function sets a width of the column with the specified index.

**Version**

Available since version 8.2.

### I.1.6.3.12 SetRowHeight

**C++**

```
HRESULT SetRowHeight(long _row_idx, long _row_height);
```

**C#**

```
public void SetRowHeight(long _row_idx, long _row_height);
```

**Visual Basic**

```
Public Sub SetRowHeight(_row_idx As long, _row_height As long)
```

**Description**

Function sets a height of the row with the specified index.

**Version**

Available since version 8.2.

### I.1.6.3.13 StoreModeSelection

**C++**

```
HRESULT StoreModeSelection(IRobotModeSelectionType _type, BSTR _sel_text,
IRobotModeCombinationType _comb);
```

**C#**

```
public void StoreModeSelection(IRobotModeSelectionType _type, String _sel_text,
IRobotModeCombinationType _comb);
```

**Visual Basic**

```
Public Sub StoreModeSelection(_type As IRobotModeSelectionType, _sel_text As String, _comb
As IRobotModeCombinationType)
```

**Description**

The function saves the indicated mode selection to the table. .

### I.1.6.3.14 StoreSelection

**C++**

```
HRESULT StoreSelection(IRobotObjectType _sel_type, BSTR _sel_txt);
```

**C#**

```
public void StoreSelection(IRobotObjectType _sel_type, String _sel_txt);
```

**Visual Basic**

```
Public Sub StoreSelection(_sel_type As IRobotObjectType, _sel_txt As String)
```

**Description**

The function saves the indicated selection in the table without modifying its filter. One may save one selection of each type of objects - structure components - in the table..

## I.1.7 IRobotTableFrame

**Class Hierarchy**

**C++**

```
interface IRobotTableFrame : IDispatch;
```

**C#**

```
public interface IRobotTableFrame;
```

**Visual Basic**

```
Public Interface IRobotTableFrame
```

**Description**

Interface representing the viewer composed of many tables. Each table is a separate tab defined within the same main frame.

**Version**

Available since version 3.

**I.1.7.1 IRobotTableFrame Members**

The following tables list the members exposed by IRobotTableFrame.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1367</a> )	Number of tables defined in the same main frame.
◆	Current ( <a href="#">see page 1367</a> )	Index of active table.
◆	Window ( <a href="#">see page 1368</a> )	

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Get ( <a href="#">see page 1368</a> )	Function returns the table of the specified index.
◆	GetName ( <a href="#">see page 1368</a> )	Function returns the name of the tab on which the table of the indicated index is displayed. .
◆	SetName ( <a href="#">see page 1369</a> )	Function sets the name for the tab which presents the table of the indicated index.

**I.1.7.2 IRobotTableFrame Fields**

The fields of the IRobotTableFrame class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1367</a> )	Number of tables defined in the same main frame.
◆	Current ( <a href="#">see page 1367</a> )	Index of active table.
◆	Window ( <a href="#">see page 1368</a> )	

**I.1.7.2.1 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of tables defined in the same main frame.

**Version**

Available since version 3.

**I.1.7.2.2 Current****C++**

```
HRESULT get_Current(long* );
HRESULT put_Current(long);
```

**C#**

```
public long Current { get; set; }
```

**Visual Basic**

```
Public Current As long
```

**Description**

Index of active table.

**Version**

Available since version 3.

**I.1.7.2.3 Window****C++**

```
HRESULT get_Window(IRobotWindow** );
```

**C#**

```
public IRobotWindow Window { get; }
```

**Visual Basic**

```
Public ReadOnly Window As IRobotWindow
```

**Version**

Available since version 3.

**I.1.7.3 IRobotTableFrame Methods**

The methods of the IRobotTableFrame class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Get (see page 1368)	Function returns the table of the specified index.
💡	GetName (see page 1368)	Function returns the name of the tab on which the table of the indicated index is displayed. .
💡	SetName (see page 1369)	Function sets the name for the tab which presents the table of the indicated index.

**I.1.7.3.1 Get****C++**

```
HRESULT Get(long _idx, IRobotTable** ret);
```

**C#**

```
public IRobotTable Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotTable
```

**Description**

Function returns the table of the specified index.

**Version**

Available since version 3.

**I.1.7.3.2 GetName****C++**

```
HRESULT GetName(long _table_idx, BSTR* ret);
```

**C#**

```
public String GetName(long _table_idx);
```

**Visual Basic**

```
Public Function GetName(_table_idx As long) As String
```

**Description**

Function returns the name of the tab on which the table of the indicated index is displayed. .

**Version**

Available since version 3.

**I.1.7.3.3 SetName****C++**

```
HRESULT SetName(long _table_idx, BSTR _name);
```

**C#**

```
public void SetName(long _table_idx, String _name);
```

**Visual Basic**

```
Public Sub SetName(_table_idx As long, _name As String)
```

**Description**

Function sets the name for the tab which presents the table of the indicated index.

**Version**

Available since version 3.

**I.1.8 IRobotTableScreenCaptureParams****Class Hierarchy****C++**

```
interface IRobotTableScreenCaptureParams : IDispatch;
```

**C#**

```
public interface IRobotTableScreenCaptureParams;
```

**Visual Basic**

```
Public Interface IRobotTableScreenCaptureParams
```

## Description

Parameters of the screen capture for a table.

## Version

Available since version 8.2.

### I.1.8.1 IRobotTableScreenCaptureParams Members

The following tables list the members exposed by IRobotTableScreenCaptureParams.

#### Public Fields

	Name	Description
◆	AddInfoTab ( <a href="#">see page 1370</a> )	
◆	AdjustColumnWidth ( <a href="#">see page 1370</a> )	The flag indicating if width of the table columns should be adjusted automatically to width of the html page.
◆	Comment ( <a href="#">see page 1371</a> )	Comment text to be added to the screen capture.
◆	IncludeDateAndTime ( <a href="#">see page 1371</a> )	The flag indicating if the current date and time should be added automatically to the screen capture.
◆	Name ( <a href="#">see page 1371</a> )	
◆	UpdateType ( <a href="#">see page 1371</a> )	

### I.1.8.2 IRobotTableScreenCaptureParams Fields

The fields of the IRobotTableScreenCaptureParams class are listed here.

#### Public Fields

	Name	Description
◆	AddInfoTab ( <a href="#">see page 1370</a> )	
◆	AdjustColumnWidth ( <a href="#">see page 1370</a> )	The flag indicating if width of the table columns should be adjusted automatically to width of the html page.
◆	Comment ( <a href="#">see page 1371</a> )	Comment text to be added to the screen capture.
◆	IncludeDateAndTime ( <a href="#">see page 1371</a> )	The flag indicating if the current date and time should be added automatically to the screen capture.
◆	Name ( <a href="#">see page 1371</a> )	
◆	UpdateType ( <a href="#">see page 1371</a> )	

#### I.1.8.2.1 AddInfoTab

##### C++

```
HRESULT get_AddInfoTab(VARIANT_BOOL* );
HRESULT put_AddInfoTab(VARIANT_BOOL);
```

##### C#

```
public bool AddInfoTab { get; set; }
```

##### Visual Basic

```
Public AddInfoTab As Boolean
```

## Version

Available since version 8.2.

#### I.1.8.2.2 AdjustColumnWidth

##### C++

```
HRESULT get_AdjustColumnWidth(VARIANT_BOOL* );
HRESULT put_AdjustColumnWidth(VARIANT_BOOL);
```

**C#**

```
public bool AdjustColumnWidth { get; set; }
```

**Visual Basic**

```
Public AdjustColumnWidth As Boolean
```

**Description**

The flag indicating if width of the table columns should be adjusted automatically to width of the html page.

**Version**

Available since version 12.

**I.1.8.2.3 Comment****C++**

```
HRESULT get_Comment(BSTR* );
HRESULT put_Comment(BSTR);
```

**C#**

```
public String Comment { get; set; }
```

**Visual Basic**

```
Public Comment As String
```

**Description**

Comment text to be added to the screen capture.

**Version**

Available since version 12.

**I.1.8.2.4 IncludeDateAndTime****C++**

```
HRESULT get_IncludeDateAndTime(VARIANT_BOOL* );
HRESULT put_IncludeDateAndTime(VARIANT_BOOL);
```

**C#**

```
public bool IncludeDateAndTime { get; set; }
```

**Visual Basic**

```
Public IncludeDateAndTime As Boolean
```

**Description**

The flag indicating if the current date and time should be added automatically to the screen capture.

**Version**

Available since version 12.

**I.1.8.2.5 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Version**

Available since version 8.2.

**I.1.8.2.6 UpdateType****C++**

```
HRESULT get_UpdateType(IRobotTableScreenCaptureUpdateType* );
HRESULT put_UpdateType(IRobotTableScreenCaptureUpdateType);
```

**C#**

```
public IRobotTableScreenCaptureUpdateType UpdateType { get; set; }
```

**Visual Basic**

```
Public UpdateType As IRobotTableScreenCaptureUpdateType
```

**Version**

Available since version 8.2.

**I.1.9 IRobotTableScreenCaptureUpdateType****C++**

```
enum IRobotTableScreenCaptureUpdateType;
```

**C#**

```
public enum IRobotTableScreenCaptureUpdateType;
```

**Visual Basic**

```
Public Enum IRobotTableScreenCaptureUpdateType
```

**Members**

Members	Description
I_TSCUT_UPDATED_UPON_PRINTING = 1	Available since version 8.2.
I_TSCUT_CURRENT_VIEW = 2	Available since version 8.2.
I_TSCUT_COPY_TO_CLIPBOARD = 4	The screen capture will be saved only to the clipboard. Available since version 12.

**Version**

Available since version 8.2.

**I.2 Graphical structure views****Enumerations**

	Name	Description
	IRobotViewType (see page 1388)	Set of available types of graphical views of the structure that can be created by means of the view manager. .
	IRobotViewProjection (see page 1398)	The set of identifiers determining different types of projections to be presented in graphical views has been defined. .
	IRobotViewVisibilityStatusType (see page 1399)	Available types of visibility status for a graphical viewer.
	IRobotViewVisibilityStatusValue (see page 1400)	Accessible values for particular types of visibility status.

	IRobotViewScaleType (see page 1405)	Available types of a scale defined for a graphical viewer.
	IRobotViewDetailedAnalysisTableTab (see page 1409)	Identifiers of the tabs in the table being a part of the view presenting detailed analysis results. Available since version 2.0.
	IRobotViewDisplayAttributes (see page 1411)	Set of attributes whose visibility may be switched on/off for graphical views. .
	IRobotViewHiddenLinesDisplayType (see page 1415)	Possible types of displaying hidden lines.
	IRobotViewDiagramResultType (see page 1420)	Set of result types presented by means of diagrams on bars.
	IRobotViewDiagramDescriptionType (see page 1421)	Available methods of locating descriptions for diagrams.
	IRobotViewDiagramFillingType (see page 1421)	Available methods of diagram filling.
	IRobotViewDiagramSignDifferType (see page 1422)	Available methods of distinguishing positive and negative values on diagrams.
	IRobotViewBarMapResultType (see page 1423)	Set of result types presented by means of maps on bars.
	IRobotViewDetailedAnalysisResultType (see page 1426)	Result types presented by detailed analysis view. .
	IRobotViewFeMapResultType (see page 1430)	Result types presented by means of maps on finite elements. .
	IRobotViewFeMapLocalSystemType (see page 1437)	
	IRobotViewFeMapCrossPresentationType (see page 1437)	Manner of presenting crosses in the map view for surface finite elements. .
	IRobotViewFeMapLayerType (see page 1438)	
	IRobotViewFeMapSmoothingType (see page 1438)	
	IRobotViewDiagramPositionType (see page 1441)	Available diagram positions with respect to the panel plane.
	IRobotViewReinforcementResultType (see page 1441)	Available reinforcement results presented in the graphical view. .
	IRobotViewGlobalAnalysisResultsType (see page 1446)	Types of result display.
	IRobotViewGlobalAnalysisParamsType (see page 1446)	Parameter types.
	IRobotViewScreenCaptureUpdateType (see page 1452)	Available methods of updating the contents of a screen capture for the graphical view.
	IRobotViewDiagramValueType (see page 1454)	Types of values displayed on diagrams.
	IRobotViewScreenCaptureResolution (see page 1455)	

## Interfaces

	Name	Description
	IRobotView (see page 1389)	Interface representing the graphical structure view. .
	IRobotViewDetailedAnalysis (see page 1406)	View presenting results of the detailed analysis. Available since version 2.0.
	IRobotViewDisplayParams (see page 1409)	Parameters of display for graphical viewers.
	IRobotViewDiagramParams (see page 1415)	Interface controlling the manner of displaying diagrams on bars. .
	IRobotViewDiagrams (see page 1422)	A viewer dedicated to result presentation.

	IRobotViewBarMaps (see page 1422)	View presenting maps on bars with a scale. .
	IRobotViewFeMaps (see page 1423)	View presenting maps on finite elements with a scale. .
	IRobotViewBarMapParams (see page 1424)	Parameters describing the manner of displaying maps on bars.
	IRobotViewDetailedAnalysisParams (see page 1427)	Parameters of detailed analysis view.
	IRobotViewFeMapParams (see page 1431)	Parameters of presenting maps on finite elements. .
	IRobotView2 (see page 1438)	Extended graphical view interface. .
	IRobotViewGlobalAnalysisParams (see page 1442)	Interface controlling display of global analysis results. .
	IRobotViewGlobalAnalysisResultsParams (see page 1444)	Interface controlling result display.
	IRobotViewGlobalAnalysis (see page 1447)	View presenting global analysis results. .
	IRobotViewScreenCaptureParams (see page 1449)	Parameters of a screen capture for the graphical view.
	IRobotView3 (see page 1453)	

## I.2.1 Panel cuts

Available since version 3.

### Interfaces

	Name	Description
	IRobotViewPanelCuts (see page 1374)	View presenting cuts through panels.
	IRobotViewPanelCutParams (see page 1374)	Parameters navigating panel cut display.

### I.2.1.1 IRobotViewPanelCuts

#### Class Hierarchy

#### C++

```
interface IRobotViewPanelCuts : IRobotView3;
```

#### C#

```
public interface IRobotViewPanelCuts : IRobotView3;
```

#### Visual Basic

```
Public Interface IRobotViewPanelCuts
```

#### Description

View presenting cuts through panels.

#### Version

Available since version 3.

### I.2.1.2 IRobotViewPanelCutParams

#### Class Hierarchy

#### C++

```
interface IRobotViewPanelCutParams : IDispatch;
```

**C#**

```
public interface IRobotViewPanelCutParams;
```

**Visual Basic**

```
Public Interface IRobotViewPanelCutParams
```

**Description**

Parameters navigating panel cut display.

**Version**

Available since version 3.

**I.2.1.2.1 IRobotViewPanelCutParams Members**

The following tables list the members exposed by IRobotViewPanelCutParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CurrentReinforcementResult (see page 1375)	Currently displayed result type for reinforcement.
❖	CurrentResult (see page 1376)	Currently displayed result type.
❖	Descriptpions (see page 1376)	Method of description display.
❖	Filling (see page 1376)	Method of diagram filling.
❖	IntegralValue (see page 1377)	Activation of a label containing the value of integral for the selected component along the cutting line length.
❖	Layer (see page 1377)	Layer.
❖	LayerArbitraryValue (see page 1377)	Layer (see page 1377) thickness.
❖	Position (see page 1378)	Type of diagram position with respect to the panel plane.
❖	Smoothing (see page 1378)	Type of diagram smoothing.

**I.2.1.2.2 IRobotViewPanelCutParams Fields**

The fields of the IRobotViewPanelCutParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CurrentReinforcementResult (see page 1375)	Currently displayed result type for reinforcement.
❖	CurrentResult (see page 1376)	Currently displayed result type.
❖	Descriptpions (see page 1376)	Method of description display.
❖	Filling (see page 1376)	Method of diagram filling.
❖	IntegralValue (see page 1377)	Activation of a label containing the value of integral for the selected component along the cutting line length.
❖	Layer (see page 1377)	Layer.
❖	LayerArbitraryValue (see page 1377)	Layer (see page 1377) thickness.
❖	Position (see page 1378)	Type of diagram position with respect to the panel plane.
❖	Smoothing (see page 1378)	Type of diagram smoothing.

**I.2.1.2.2.1 CurrentReinforcementResult****C++**

```
HRESULT get_CurrentReinforcementResult(IRobotViewReinforcementResultType* );
```

```
HRESULT put_CurrentReinforcementResult(IRobotViewReinforcementResultType);
```

**C#**

```
public IRobotViewReinforcementResultType CurrentReinforcementResult { get; set; }
```

**Visual Basic**

```
Public CurrentReinforcementResult As IRobotViewReinforcementResultType
```

**Description**

Currently displayed result type for reinforcement.

**Version**

Available since version 3.

### I.2.1.2.2.2 CurrentResult

**C++**

```
HRESULT get_CurrentResult(IRobotViewFeMapResultType*);  
HRESULT put_CurrentResult(IRobotViewFeMapResultType);
```

**C#**

```
public IRobotViewFeMapResultType CurrentResult { get; set; }
```

**Visual Basic**

```
Public CurrentResult As IRobotViewFeMapResultType
```

**Description**

Currently displayed result type.

**Version**

Available since version 3.

### I.2.1.2.2.3 Descriptions

**C++**

```
HRESULT get_Descriptions(IRobotViewDiagramDescriptionType*);  
HRESULT put_Descriptions(IRobotViewDiagramDescriptionType);
```

**C#**

```
public IRobotViewDiagramDescriptionType Descriptions { get; set; }
```

**Visual Basic**

```
Public Descriptions As IRobotViewDiagramDescriptionType
```

**Description**

Method of description display.

**Version**

Available since version 3.

### I.2.1.2.2.4 Filling

**C++**

```
HRESULT get_Filling(IRobotViewDiagramFillingType*);  
HRESULT put_Filling(IRobotViewDiagramFillingType);
```

**C#**

```
public IRobotViewDiagramFillingType Filling { get; set; }
```

**Visual Basic**

```
Public Filling As IRobotViewDiagramFillingType
```

**Description**

Method of diagram filling.

**Version**

Available since version 3.

**I.2.1.2.2.5 IntegralValue****C++**

```
HRESULT get_IntegralValue(VARIANT_BOOL* );
HRESULT put_IntegralValue(VARIANT_BOOL);
```

**C#**

```
public bool IntegralValue { get; set; }
```

**Visual Basic**

```
Public IntegralValue As Boolean
```

**Description**

Activation of a label containing the value of integral for the selected component along the cutting line length.

**Version**

Available since version 3.

**I.2.1.2.2.6 Layer****C++**

```
HRESULT get_Layer(IRobotViewFeMapLayerType* );
HRESULT put_Layer(IRobotViewFeMapLayerType);
```

**C#**

```
public IRobotViewFeMapLayerType Layer { get; set; }
```

**Visual Basic**

```
Public Layer As IRobotViewFeMapLayerType
```

**Description**

Layer.

**Version**

Available since version 3.

**I.2.1.2.2.7 LayerArbitraryValue****C++**

```
HRESULT get_LayerArbitraryValue(double* );
HRESULT put_LayerArbitraryValue(double);
```

**C#**

```
public double LayerArbitraryValue { get; set; }
```

**Visual Basic**

```
Public LayerArbitraryValue As double
```

**Description**

Layer (see page 1377) thickness.

**Version**

Available since version 3.

**I.2.1.2.2.8 Position****C++**

```
HRESULT get_Position(IRobotViewDiagramPositionType* );
HRESULT put_Position(IRobotViewDiagramPositionType);
```

**C#**

```
public IRobotViewDiagramPositionType Position { get; set; }
```

**Visual Basic**

```
Public Position As IRobotViewDiagramPositionType
```

**Description**

Type of diagram position with respect to the panel plane.

**Version**

Available since version 3.

**I.2.1.2.2.9 Smoothing****C++**

```
HRESULT get_Smoothing(IRobotViewFeMapSmoothingType* );
HRESULT put_Smoothing(IRobotViewFeMapSmoothingType);
```

**C#**

```
public IRobotViewFeMapSmoothingType Smoothing { get; set; }
```

**Visual Basic**

```
Public Smoothing As IRobotViewFeMapSmoothingType
```

**Description**

Type of diagram smoothing.

**Version**

Available since version 3.

**I.2.2 Influence lines**

Available since version 3.5.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotViewInfluenceLinesResultType (see page 1386)	Available result types for the influence line view. .
	IRobotViewInfluenceLinesLocalSystemType (see page 1387)	Available types of local coordinate systems. .
	IRobotViewInfluenceLinesLayerType (see page 1388)	Types of layers.

## Interfaces

	Name	Description
↳	IRobotViewInfluenceLines (see page 1379)	Graphical view presenting an influence line.
↳	IRobotViewInfluenceLinesParams (see page 1381)	Parameters of an influence line view. .

### I.2.2.1 IRobotViewInfluenceLines

#### Class Hierarchy

#### C++

```
interface IRobotViewInfluenceLines : IRobotView2;
```

#### C#

```
public interface IRobotViewInfluenceLines : IRobotView2;
```

#### Visual Basic

```
Public Interface IRobotViewInfluenceLines
```

#### Description

Graphical view presenting an influence line.

#### Version

Available since version 3.5.

### I.2.2.1.1 IRobotViewInfluenceLines Members

The following tables list the members exposed by IRobotViewInfluenceLines.

#### Public Fields

	Name	Description
❖	DiagramMagnification (see page 1391)	Diagram enlarging (scale: 1 to 10).
❖	ParamsBarMap (see page 1391)	Parameters of displaying maps an bars Available since version 2.0.
❖	ParamsDiagram (see page 1391)	Parameters of displaying diagrams on the view Available since version 2.0.
❖	ParamsDisplay (see page 1391)	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
❖	ParamsFeMap (see page 1392)	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
❖	ParamsInfluenceLines (see page 1380)	View parameters. Available since version 3.5.
❖	ParamsPanelCut (see page 1440)	Parameters of displaying panel cuts.
❖	Printable (see page 1392)	Object representing the view that may be attached to the printout and printed .
❖	Projection (see page 1392)	Current projection .
❖	Selection (see page 1392)	Selection of structure components presented in the view .
❖	Title (see page 1393)	View title .
❖	Type (see page 1393)	View type .
❖	UserControl (see page 1440)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
❖	Visible (see page 1393)	View display flag .
❖	Window (see page 1440)	

## Public Methods

	Name	Description
≡	CopyToClipboard ( [ see page 1394) )	The function inserts the view into the clipboard.
≡	GetRotationPoint ( [ see page 1394) )	The function gives back a rotation center during a view rotation.
≡	GetScale ( [ see page 1394) )	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
≡	GetSize ( [ see page 1395) )	The function returns the size of the view in mm. .
≡	GetWorkPoint ( [ see page 1395) )	The function gives a work point back (on the XY plane it gives Z level) .
≡	GetZoom ( [ see page 1395) )	The function returns the current zoom - coordinates of view sides in the current coordinate system .
≡	IsLocal ( [ see page 1396) )	The function gives the status of a local coordinate system back.
≡	MakeScreenCapture ( [ see page 1381) )	
≡	Redraw ( [ see page 1396) )	The function draws the view. .
≡	Rotate ( [ see page 1396) )	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
≡	SetGlobal ( [ see page 1396) )	The function sets a global coordinate system.
≡	SetLocal ( [ see page 1397) )	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
≡	SetRotationPoint ( [ see page 1397) )	The funcion sets the rotation center during the view rotation.
≡	SetScale ( [ see page 1397) )	The function assigns a given value to the scale.
≡	SetSize ( [ see page 1397) )	The function sets the view size in mm .
≡	SetWorkPoint ( [ see page 1398) )	The function sets a working point (for the XY plane Z level is set). .
≡	SetZoom ( [ see page 1398) )	The function sets the current zoom - coordinates of view sides in the current coordinate system .

### I.2.2.1.2 IRobotViewInfluenceLines Fields

The fields of the IRobotViewInfluenceLines class are listed here.

#### Public Fields

	Name	Description
◆	ParamsInfluenceLines ( [ see page 1380) )	View parameters. Available since version 3.5.

### I.2.2.1.2.1 ParamsInfluenceLines

#### C#

```
void ParamsInfluenceLines;
```

#### Description

View parameters.

Available since version 3.5.

### I.2.2.1.3 IRobotViewInfluenceLines Methods

The methods of the IRobotViewInfluenceLines class are listed here.

#### Public Methods

	Name	Description
≡	MakeScreenCapture ( [ see page 1381) )	

### I.2.2.1.3.1 MakeScreenCapture

**C++**

```
HRESULT MakeScreenCapture(IRobotViewScreenCaptureParams* _sc_params);
```

**C#**

```
public void MakeScreenCapture(IRobotViewScreenCaptureParams _sc_params);
```

**Visual Basic**

```
Public Sub MakeScreenCapture(ByRef _sc_params As IRobotViewScreenCaptureParams)
```

### I.2.2.2 IRobotViewInfluenceLinesParams

**Class Hierarchy**

**C++**

```
interface IRobotViewInfluenceLinesParams : IDispatch;
```

**C#**

```
public interface IRobotViewInfluenceLinesParams;
```

**Visual Basic**

```
Public Interface IRobotViewInfluenceLinesParams
```

**Description**

Parameters of an influence line view. .

**Version**

Available since version 3.5.

### I.2.2.2.1 IRobotViewInfluenceLinesParams Members

The following tables list the members exposed by IRobotViewInfluenceLinesParams.

**Public Fields**

	Name	Description
◆	Direction ( [ see page 1382) )	Type of the local reference system.
◆	DirectionData ( [ see page 1382) )	Vector defining the local reference system; used in appropriate system types (CARTESIAN_ALONG_VECTOR and POLAR_IN_POINT).
◆	DirectionNode ( [ see page 1383) )	Node number for definition of the POLAR_IN_POINT local reference system .
◆	Element ( [ see page 1383) )	Number of element/bar for which an influence line will be generated .
◆	Layer ( [ see page 1383) )	Layer type.
◆	LayerArbitraryVal ( [ see page 1384) )	Value of the ARBITRARY type layer.
◆	Position ( [ see page 1384) )	Relative coordinate of the point of element/bar for which an influence line will be generated .
◆	RangeFrom ( [ see page 1384) )	Initial number of the moving load position with which generation of an influence line will start .
◆	RangeTo ( [ see page 1385) )	Final number of the moving load position (associated with RangeFrom ( [ see page 1384) )).

**Public Methods**

	Name	Description
◆	IsOn ( [ see page 1385) )	Function checks if a given result type is currently displayed. .

	Set (see page 1385)	Function manages display of results of the specified type..
-----------------------------------------------------------------------------------	---------------------	-------------------------------------------------------------

### I.2.2.2.2 IRobotViewInfluenceLinesParams Fields

The fields of the IRobotViewInfluenceLinesParams class are listed here.

#### Public Fields

	Name	Description
◆	Direction (see page 1382)	Type of the local reference system.
◆	DirectionData (see page 1382)	Vector defining the local reference system; used in appropriate system types (CARTESIAN_ALONG_VECTOR and POLAR_IN_POINT).
◆	DirectionNode (see page 1383)	Node number for definition of the POLAR_IN_POINT local reference system .
◆	Element (see page 1383)	Number of element/bar for which an influence line will be generated .
◆	Layer (see page 1383)	Layer type.
◆	LayerArbitraryVal (see page 1384)	Value of the ARBITRARY type layer.
◆	Position (see page 1384)	Relative coordinate of the point of element/bar for which an influence line will be generated .
◆	RangeFrom (see page 1384)	Initial number of the moving load position with which generation of an influence line will start .
◆	RangeTo (see page 1385)	Final number of the moving load position (associated with RangeFrom (see page 1384)).

#### I.2.2.2.2.1 Direction

##### C++

```
HRESULT get_Direction(IRobotViewInfluenceLinesLocalSystemType* );
HRESULT put_Direction(IRobotViewInfluenceLinesLocalSystemType);
```

##### C#

```
public IRobotViewInfluenceLinesLocalSystemType Direction { get; set; }
```

##### Visual Basic

```
Public Direction As IRobotViewInfluenceLinesLocalSystemType
```

##### Description

Type of the local reference system.

##### Version

Available since version 3.5.

#### I.2.2.2.2.2 DirectionData

##### C++

```
HRESULT get_DirectionData(IRobotGeoPoint3D** );
HRESULT put_DirectionData(IRobotGeoPoint3D* );
```

##### C#

```
public IRobotGeoPoint3D DirectionData { get; set; }
```

##### Visual Basic

```
Public DirectionData As IRobotGeoPoint3D
```

##### Description

Vector defining the local reference system; used in appropriate system types (CARTESIAN\_ALONG\_VECTOR and POLAR\_IN\_POINT).

**Version**

Available since version 3.5.

**I.2.2.2.2.3 DirectionNode****C++**

```
HRESULT get_DirectionNode(long*);  
HRESULT put_DirectionNode(long);
```

**C#**

```
public long DirectionNode { get; set; }
```

**Visual Basic**

```
Public DirectionNode As long
```

**Description**

Node number for definition of the POLAR\_IN\_POINT local reference system .

**Version**

Available since version 3.5.

**I.2.2.2.2.4 Element****C++**

```
HRESULT get_Element(long*);  
HRESULT put_Element(long);
```

**C#**

```
public long Element { get; set; }
```

**Visual Basic**

```
Public Element As long
```

**Description**

Number of element/bar for which an influence line will be generated .

**Version**

Available since version 3.5.

**I.2.2.2.2.5 Layer****C++**

```
HRESULT get_Layer(IRobotViewInfluenceLinesLayerType*);  
HRESULT put_Layer(IRobotViewInfluenceLinesLayerType);
```

**C#**

```
public IRobotViewInfluenceLinesLayerType Layer { get; set; }
```

**Visual Basic**

```
Public Layer As IRobotViewInfluenceLinesLayerType
```

**Description**

Layer type.

**Version**

Available since version 3.5.

### I.2.2.2.6 LayerArbitraryVal

#### C++

```
HRESULT get_LayerArbitraryVal(double*);  
HRESULT put_LayerArbitraryVal(double);
```

#### C#

```
public double LayerArbitraryVal { get; set; }
```

#### Visual Basic

```
Public LayerArbitraryVal As Double
```

#### Description

Value of the ARBITRARY type layer.

#### Version

Available since version 3.5.

### I.2.2.2.7 Position

#### C++

```
HRESULT get_Position(double*);  
HRESULT put_Position(double);
```

#### C#

```
public double Position { get; set; }
```

#### Visual Basic

```
Public Position As Double
```

#### Description

Relative coordinate of the point of element/bar for which an influence line will be generated .

#### Version

Available since version 3.5.

### I.2.2.2.8 RangeFrom

#### C++

```
HRESULT get_RangeFrom(long*);  
HRESULT put_RangeFrom(long);
```

#### C#

```
public long RangeFrom { get; set; }
```

#### Visual Basic

```
Public RangeFrom As Long
```

#### Description

Initial number of the moving load position with which generation of an influence line will start .

#### Version

Available since version 3.5.

### I.2.2.2.9 RangeTo

#### C++

```
HRESULT get_RangeTo(long* );
HRESULT put_RangeTo(long);
```

#### C#

```
public long RangeTo { get; set; }
```

#### Visual Basic

```
Public RangeTo As long
```

#### Description

Final number of the moving load position (associated with RangeFrom (see page 1384)).

#### Version

Available since version 3.5.

### I.2.2.3 IRobotViewInfluenceLinesParams Methods

The methods of the IRobotViewInfluenceLinesParams class are listed here.

#### Public Methods

	Name	Description
💡	IsOn (see page 1385)	Function checks if a given result type is currently displayed. .
💡	Set (see page 1385)	Function manages display of results of the specified type. .

### I.2.2.3.1 IsOn

#### C++

```
HRESULT IsOn(IRobotViewInfluenceLinesResultTypes _type, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsOn(IRobotViewInfluenceLinesResultTypes _type);
```

#### Visual Basic

```
Public Function IsOn(_type As IRobotViewInfluenceLinesResultTypes) As Boolean
```

#### Description

Function checks if a given result type is currently displayed. .

#### Version

Available since version 3.5.

### I.2.2.3.2 Set

#### C++

```
HRESULT Set(IRobotViewInfluenceLinesResultTypes _type, VARIANT_BOOL _val);
```

#### C#

```
public void Set(IRobotViewInfluenceLinesResultTypes _type, bool _val);
```

#### Visual Basic

```
Public Sub Set(_type As IRobotViewInfluenceLinesResultTypes, _val As Boolean)
```

## Description

Function manages display of results of the specified type. .

## Version

Available since version 3.5.

### I.2.2.3 IRobotViewInfluenceLinesResultType

#### C++

```
enum IRobotViewInfluenceLinesResultType;
```

#### C#

```
public enum IRobotViewInfluenceLinesResultType;
```

#### Visual Basic

```
Public Enum IRobotViewInfluenceLinesResultType
```

## Members

Members	Description
I_VILRT_NTM_FX = 1	Available since version 3.5.
I_VILRT_NTM_FY = 2	Available since version 3.5.
I_VILRT_NTM_FZ = 3	Available since version 3.5.
I_VILRT_NTM_MX = 4	Available since version 3.5.
I_VILRT_NTM_MY = 5	Available since version 3.5.
I_VILRT_NTM_MZ = 6	Available since version 3.5.
I_VILRT_NTM_KY = 7	Available since version 3.5.
I_VILRT_NTM_KZ = 8	Available since version 3.5.
I_VILRT_NTM_UX = 9	Available since version 3.5.
I_VILRT_NTM_UZ = 10	Available since version 3.5.
I_VILRT_NODES_UX = 11	Available since version 3.5.
I_VILRT_NODES_UY = 12	Available since version 3.5.
I_VILRT_NODES_UZ = 13	Available since version 3.5.
I_VILRT_NODES_RX = 14	Available since version 3.5.
I_VILRT_NODES_RY = 15	Available since version 3.5.
I_VILRT_NODES_RZ = 16	Available since version 3.5.
I_VILRT_NODES_FX = 17	Available since version 3.5.
I_VILRT_NODES_FY = 18	Available since version 3.5.
I_VILRT_NODES_FZ = 19	Available since version 3.5.
I_VILRT_NODES_MX = 20	Available since version 3.5.
I_VILRT_NODES_MY = 21	Available since version 3.5.
I_VILRT_NODES_MZ = 22	Available since version 3.5.
I_VILRT_DETAILED_STRESS_XX = 23	Available since version 3.5.
I_VILRT_DETAILED_STRESS YY = 24	Available since version 3.5.
I_VILRT_DETAILED_STRESS_XY = 25	Available since version 3.5.
I_VILRT_DETAILED_STRESS_Z = 26	Available since version 3.5.
I_VILRT_DETAILED_MEMBRANE_FORCE_XX = 27	Available since version 3.5.
I_VILRT_DETAILED_MEMBRANE_FORCE_YY = 28	Available since version 3.5.
I_VILRT_DETAILED_MEMBRANE_FORCE_XY = 29	Available since version 3.5.
I_VILRT_DETAILED_MOMENTS_XX = 30	Available since version 3.5.
I_VILRT_DETAILED_MOMENTS_YY = 31	Available since version 3.5.

I_VILRT_DETAILED_MOMENTS_XY = 32	Available since version 3.5.
I_VILRT_DETAILED_SHEAR_STRESS_YY = 34	Available since version 3.5.
I_VILRT_DETAILED_SHEAR_FORCE_XX = 35	Available since version 3.5.
I_VILRT_DETAILED_SHEAR_FORCE_YY = 36	Available since version 3.5.
I_VILRT_DETAILED_DISPLACEMENT_XX = 37	Available since version 3.5.
I_VILRT_DETAILED_DISPLACEMENT_YY = 38	Available since version 3.5.
I_VILRT_DETAILED_DISPLACEMENT_Z = 39	Available since version 3.5.
I_VILRT_DETAILED_ROTATION_XX = 40	Available since version 3.5.
I_VILRT_DETAILED_ROTATION_YY = 41	Available since version 3.5.
I_VILRT_DETAILED_ROTATION_Z = 42	Available since version 3.5.
I_VILRT_DETAILED_SOIL_REACTION_Z = 43	Available since version 3.5.
I_VILRT_EXTREME_STRESS_1 = 44	Available since version 3.5.
I_VILRT_EXTREME_STRESS_2 = 45	Available since version 3.5.
I_VILRT_EXTREME_STRESS_1_2 = 46	Available since version 3.5.
I_VILRT_EXTREME_STRESS_ANGLE = 47	Available since version 3.5.
I_VILRT_EXTREME_MEMBRANE_FORCES_1 = 48	Available since version 3.5.
I_VILRT_EXTREME_MEMBRANE_FORCES_2 = 49	Available since version 3.5.
I_VILRT_EXTREME_MEMBRANE_FORCES_1_2 = 50	Available since version 3.5.
I_VILRT_EXTREME_MEMBRANE_FORCES_ANGLE = 51	Available since version 3.5.
I_VILRT_EXTREME_MOMENTS_1 = 52	Available since version 3.5.
I_VILRT_EXTREME_MOMENTS_2 = 53	Available since version 3.5.
I_VILRT_EXTREME_MOMENTS_1_2 = 54	Available since version 3.5.
I_VILRT_EXTREME_MOMENTS_ANGLE = 55	Available since version 3.5.
I_VILRT_EXTREME_SHEAR_STRESS_1_2 = 56	Available since version 3.5.
I_VILRT_EXTREME_SHEAR_FORCE_1_2 = 57	Available since version 3.5.
I_VILRT_COMPLEX_STRESS = 58	Available since version 3.5.
I_VILRT_COMPLEX_MEMBRANE_FORCE = 59	Available since version 3.5.
I_VILRT_COMPLEX_MOMENT = 60	Available since version 3.5.

**Description**

Available result types for the influence line view. .

**Version**

Available since version 3.5.

**I.2.2.4 IRobotViewInfluenceLinesLocalSystemType****C++**

```
enum IRobotViewInfluenceLinesLocalSystemType;
```

**C#**

```
public enum IRobotViewInfluenceLinesLocalSystemType;
```

**Visual Basic**

```
Public Enum IRobotViewInfluenceLinesLocalSystemType
```

**Members**

Members	Description
I_VILLST_CARTESIAN_ALONG_X = 0	Available since version 3.5.
I_VILLST_CARTESIAN_ALONG_Z = 2	Available since version 3.5.
I_VILLST_CARTESIAN_ALONG_VECTOR = 3	Available since version 3.5.

I_VILLST_POLAR_IN_NODE = 4	Available since version 3.5.
I_VILLST_POLAR_IN_POINT = 5	Available since version 3.5.
I_VILLST_AUTOMATIC = 6	Available since version 3.5.

**Description**

Available types of local coordinate systems. .

**Version**

Available since version 3.5.

**I.2.2.5 IRobotViewInfluenceLinesLayerType****C++**

```
enum IRobotViewInfluenceLinesLayerType;
```

**C#**

```
public enum IRobotViewInfluenceLinesLayerType;
```

**Visual Basic**

```
Public Enum IRobotViewInfluenceLinesLayerType
```

**Members**

Members	Description
I_VILLT_UPPER = 0	Available since version 3.5.
I_VILLT_MIDDLE = 1	Available since version 3.5.
I_VILLT_LOWER = 2	Available since version 3.5.
I_VILLT_ARBITRARY = 3	Available since version 3.5.

**Description**

Types of layers.

**Version**

Available since version 3.5.

**I.2.3 IRobotViewType****C++**

```
enum IRobotViewType;
```

**C#**

```
public enum IRobotViewType;
```

**Visual Basic**

```
Public Enum IRobotViewType
```

**Members**

Members	Description
I_VT_STANDARD = 1	Standard structure view .
I_VT_DETAILED_ANALYSIS = 2	A view presenting detailed analysis results Available since version 1.7.
I_VT_DIAGRAMS = 3	Available since version 2.0.
I_VT_MAPS_ON_BARS = 4	Available since version 2.0.
I_VT_MAPS_ONFINITE_ELEMENTS = 5	Available since version 2.0.

I_VT_PANEL_CUTS = 6	Available since version 3.
I_VT_GLOBAL_ANALYSIS = 7	Available since version 3.
I_VT_INFLUENCE_LINES = 8	Available since version 3.5.

**Description**

Set of available types of graphical views of the structure that can be created by means of the view manager. .

**I.2.4 IRobotView****Class Hierarchy****C++**

```
interface IRobotView : IDispatch;
```

**C#**

```
public interface IRobotView;
```

**Visual Basic**

```
Public Interface IRobotView
```

**Description**

Interface representing the graphical structure view. .

**I.2.4.1 IRobotView Members**

The following tables list the members exposed by IRobotView.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DiagramMagnification (☞ see page 1391)	Diagram enlarging (scale: 1 to 10).
◆	ParamsBarMap (☞ see page 1391)	Parameters of displaying maps an bars Available since version 2.0.
◆	ParamsDiagram (☞ see page 1391)	Parameters of displaying diagrams on the view Available since version 2.0.
◆	ParamsDisplay (☞ see page 1391)	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
◆	ParamsFeMap (☞ see page 1392)	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
◆	Printable (☞ see page 1392)	Object representing the view that may be attached to the printout and printed .
◆	Projection (☞ see page 1392)	Current projection .
◆	Selection (☞ see page 1392)	Selection of structure components presented in the view .
◆	Title (☞ see page 1393)	View title .
◆	Type (☞ see page 1393)	View type .
◆	Visible (☞ see page 1393)	View display flag .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	CopyToClipboard (☞ see page 1394)	The function inserts the view into the clipboard.
◆	GetRotationPoint (☞ see page 1394)	The function gives back a rotation center during a view rotation.
◆	GetScale (☞ see page 1394)	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
◆	GetSize (☞ see page 1395)	The function returns the size of the view in mm. .

	GetWorkPoint ( <a href="#">see page 1395</a> )	The function gives a work point back (on the XY plane it gives Z level) .
	GetZoom ( <a href="#">see page 1395</a> )	The function returns the current zoom - coordinates of view sides in the current coordinate system .
	IsLocal ( <a href="#">see page 1396</a> )	The function gives the status of a local coordinate system back.
	Redraw ( <a href="#">see page 1396</a> )	The function draws the view. .
	Rotate ( <a href="#">see page 1396</a> )	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
	SetGlobal ( <a href="#">see page 1396</a> )	The function sets a global coordinate system.
	SetLocal ( <a href="#">see page 1397</a> )	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
	SetRotationPoint ( <a href="#">see page 1397</a> )	The funcion sets the rotation center during the view rotation.
	SetScale ( <a href="#">see page 1397</a> )	The function assigns a given value to the scale.
	SetSize ( <a href="#">see page 1397</a> )	The function sets the view size in mm .
	SetWorkPoint ( <a href="#">see page 1398</a> )	The function sets a working point (for the XY plane Z level is set). .
	SetZoom ( <a href="#">see page 1398</a> )	The function sets the current zoom - coordinates of view sides in the current coordinate system .

#### I.2.4.2 IRobotView Fields

The fields of the IRobotView class are listed here.

##### Public Fields

	Name	Description
	DiagramMagnification ( <a href="#">see page 1391</a> )	Diagram enlarging (scale: 1 to 10).
	ParamsBarMap ( <a href="#">see page 1391</a> )	Parameters of displaying maps an bars Available since version 2.0.
	ParamsDiagram ( <a href="#">see page 1391</a> )	Parameters of displaying diagrams on the view Available since version 2.0.
	ParamsDisplay ( <a href="#">see page 1391</a> )	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
	ParamsFeMap ( <a href="#">see page 1392</a> )	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
	Printable ( <a href="#">see page 1392</a> )	Object representing the view that may be attached to the printout and printed .
	Projection ( <a href="#">see page 1392</a> )	Current projection .
	Selection ( <a href="#">see page 1392</a> )	Selection of structure components presented in the view .
	Title ( <a href="#">see page 1393</a> )	View title .
	Type ( <a href="#">see page 1393</a> )	View type .
	Visible ( <a href="#">see page 1393</a> )	View display flag .

#### I.2.4.2.1 DiagramMagnification

##### C++

```
HRESULT get_DiagramMagnification(int* );
HRESULT put_DiagramMagnification(int);
```

##### C#

```
public int DiagramMagnification { get; set; }
```

##### Visual Basic

```
Public DiagramMagnification As Integer
```

##### Description

Diagram enlarging (scale: 1 to 10).

### I.2.4.2.2 ParamsBarMap

**C++**

```
HRESULT get_ParamsBarMap(IRobotViewBarMapParams**);
```

**C#**

```
public IRobotViewBarMapParams ParamsBarMap { get; }
```

**Visual Basic**

```
Public ReadOnly ParamsBarMap As IRobotViewBarMapParams
```

**Description**

Parameters of displaying maps an bars Available since version 2.0.

### I.2.4.2.3 ParamsDiagram

**C++**

```
HRESULT get_ParamsDiagram(IRobotViewDiagramParams**);
```

**C#**

```
public IRobotViewDiagramParams ParamsDiagram { get; }
```

**Visual Basic**

```
Public ReadOnly ParamsDiagram As IRobotViewDiagramParams
```

**Description**

Parameters of displaying diagrams on the view Available since version 2.0.

### I.2.4.2.4 ParamsDisplay

**C++**

```
HRESULT get_ParamsDisplay(IRobotViewDisplayParams**);
```

**C#**

```
public IRobotViewDisplayParams ParamsDisplay { get; }
```

**Visual Basic**

```
Public ReadOnly ParamsDisplay As IRobotViewDisplayParams
```

**Description**

Display parameters (a component enabling control of individual attribute display) Available since version 2.0.

### I.2.4.2.5 ParamsFeMap

**C++**

```
HRESULT get_ParamsFeMap(IRobotViewFeMapParams**);
```

**C#**

```
public IRobotViewFeMapParams ParamsFeMap { get; }
```

**Visual Basic**

```
Public ReadOnly ParamsFeMap As IRobotViewFeMapParams
```

**Description**

Display parameters of maps presenting results for surface finite elements Available since version 2.0.

#### I.2.4.2.6 Printable

##### C++

```
HRESULT get_Printable(IRobotPrintable**);
```

##### C#

```
public IRobotPrintable Printable { get; }
```

##### Visual Basic

```
Public ReadOnly Printable As IRobotPrintable
```

##### Description

Object representing the view that may be attached to the printout and printed .

#### I.2.4.2.7 Projection

##### C++

```
HRESULT get_Projection(IRobotViewProjection*);  
HRESULT put_Projection(IRobotViewProjection);
```

##### C#

```
public IRobotViewProjection Projection { get; set; }
```

##### Visual Basic

```
Public Projection As IRobotViewProjection
```

##### Description

Current projection .

#### I.2.4.2.8 Selection

##### C++

```
HRESULT get_Selection(IRobotMultiSelection**);
```

##### C#

```
public IRobotMultiSelection Selection { get; }
```

##### Visual Basic

```
Public ReadOnly Selection As IRobotMultiSelection
```

##### Description

Selection of structure components presented in the view .

#### I.2.4.2.9 Title

##### C++

```
HRESULT get_Title(BSTR*);  
HRESULT put_Title(BSTR);
```

##### C#

```
public String Title { get; set; }
```

##### Visual Basic

```
Public Title As String
```

##### Description

View title .

### I.2.4.2.10 Type

#### C++

```
HRESULT get_Type(IRobotViewType*);
```

#### C#

```
public IRobotViewType Type { get; }
```

#### Visual Basic

```
Public ReadOnly Type As IRobotViewType
```

#### Description

View type .

### I.2.4.2.11 Visible

#### C++

```
HRESULT get_Visible(VARIANT_BOOL*);  
HRESULT put_Visible(VARIANT_BOOL);
```

#### C#

```
public bool Visible { get; set; }
```

#### Visual Basic

```
Public Visible As Boolean
```

#### Description

View display flag .

### I.2.4.3 IRobotView Methods

The methods of the IRobotView class are listed here.

#### Public Methods

	Name	Description
≡	CopyToClipboard ( [ see page 1394] )	The function inserts the view into the clipboard.
≡	GetRotationPoint ( [ see page 1394] )	The function gives back a rotation center during a view rotation.
≡	GetScale ( [ see page 1394] )	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
≡	GetSize ( [ see page 1395] )	The function returns the size of the view in mm. .
≡	GetWorkPoint ( [ see page 1395] )	The function gives a work point back (on the XY plane it gives Z level) .
≡	GetZoom ( [ see page 1395] )	The function returns the current zoom - coordinates of view sides in the current coordinate system .
≡	IsLocal ( [ see page 1396] )	The function gives the status of a local coordinate system back.
≡	Redraw ( [ see page 1396] )	The function draws the view. .
≡	Rotate ( [ see page 1396] )	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
≡	SetGlobal ( [ see page 1396] )	The function sets a global coordinate system.
≡	SetLocal ( [ see page 1397] )	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
≡	SetRotationPoint ( [ see page 1397] )	The funcion sets the rotation center during the view rotation.
≡	SetScale ( [ see page 1397] )	The function assigns a given value to the scale.
≡	SetSize ( [ see page 1397] )	The function sets the view size in mm .

	SetWorkPoint (see page 1398)	The function sets a working point (for the XY plane Z level is set) ..
	SetZoom (see page 1398)	The function sets the current zoom - coordinates of view sides in the current coordinate system .

### I.2.4.3.1 CopyToClipboard

C++

```
HRESULT CopyToClipboard();
```

C#

```
public void CopyToClipboard();
```

Visual Basic

```
Public Sub CopyToClipboard()
```

Description

The function inserts the view into the clipboard.

### I.2.4.3.2 GetRotationPoint

C++

```
HRESULT GetRotationPoint(IRobotGeoPoint3D** ret);
```

C#

```
public IRobotGeoPoint3D GetRotationPoint();
```

Visual Basic

```
Public Function GetRotationPoint() As IRobotGeoPoint3D
```

Description

The function gives back a rotation center during a view rotation.

### I.2.4.3.3 GetScale

C++

```
HRESULT GetScale(int _scale_id, double* ret);
```

C#

```
public double GetScale(int _scale_id);
```

Visual Basic

```
Public Function GetScale(_scale_id As int) As double
```

Description

The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.

### I.2.4.3.4 GetSize

C++

```
HRESULT GetSize(double* _width_in_mm, double* _height_in_mm);
```

C#

```
public void GetSize(double* _width_in_mm, double* _height_in_mm);
```

**Visual Basic**

```
Public Sub GetSize(ByRef _width_in_mm As Double*, ByRef _height_in_mm As Double*)
```

**Description**

The function returns the size of the view in mm. .

**I.2.4.3.5 GetWorkPoint****C++**

```
HRESULT GetWorkPoint(IRobotGeoPoint3D** ret);
```

**C#**

```
public IRobotGeoPoint3D GetWorkPoint();
```

**Visual Basic**

```
Public Function GetWorkPoint() As IRobotGeoPoint3D
```

**Description**

The function gives a work point back (on the XY plane it gives Z level) .

**I.2.4.3.6 GetZoom****C++**

```
HRESULT GetZoom(double* _left, double* _top, double* _right, double* _bottom);
```

**C#**

```
public void GetZoom(double* _left, double* _top, double* _right, double* _bottom);
```

**Visual Basic**

```
Public Sub GetZoom(ByRef _left As Double*, ByRef _top As Double*, ByRef _right As Double*, ByRef _bottom As Double*)
```

**Description**

The function returns the current zoom - coordinates of view sides in the current coordinate system .

**I.2.4.3.7 IsLocal****C++**

```
HRESULT IsLocal(VARIANT_BOOL* ret);
```

**C#**

```
public bool IsLocal();
```

**Visual Basic**

```
Public Function IsLocal() As Boolean
```

**Description**

The function gives the status of a local coordinate system back.

**I.2.4.3.8 Redraw****C++**

```
HRESULT Redraw(VARIANT_BOOL _force_init_zoom);
```

**C#**

```
public void Redraw(bool _force_init_zoom);
```

**Visual Basic**

```
Public Sub Redraw(_force_init_zoom As Boolean)
```

**Description**

The function draws the view. .

**I.2.4.3.9 Rotate****C++**

```
HRESULT Rotate(IRobotGeoCoordinateAxis _axis, double _angle);
```

**C#**

```
public void Rotate(IRobotGeoCoordinateAxis _axis, double _angle);
```

**Visual Basic**

```
Public Sub Rotate(_axis As IRobotGeoCoordinateAxis, _angle As double)
```

**Description**

The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.

**I.2.4.3.10 SetGlobal****C++**

```
HRESULT SetGlobal();
```

**C#**

```
public void SetGlobal();
```

**Visual Basic**

```
Public Sub SetGlobal()
```

**Description**

The function sets a global coordinate system.

**I.2.4.3.11 SetLocal****C++**

```
HRESULT SetLocal(IRobotGeoPoint3D* _p1, IRobotGeoPoint3D* _p2, IRobotGeoPoint3D* _p3,  
VARIANT_BOOL* ret);
```

**C#**

```
public bool SetLocal(IRobotGeoPoint3D _p1, IRobotGeoPoint3D _p2, IRobotGeoPoint3D _p3);
```

**Visual Basic**

```
Public Function SetLocal(ByRef _p1 As IRobotGeoPoint3D, ByRef _p2 As IRobotGeoPoint3D,  
ByRef _p3 As IRobotGeoPoint3D) As Boolean
```

**Description**

The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.

#### I.2.4.3.12 SetRotationPoint

C++

```
HRESULT SetRotationPoint(IRobotGeoPoint3D* _rot_point);
```

C#

```
public void SetRotationPoint(IRobotGeoPoint3D _rot_point);
```

Visual Basic

```
Public Sub SetRotationPoint(ByRef _rot_point As IRobotGeoPoint3D)
```

Description

The function sets the rotation center during the view rotation.

#### I.2.4.3.13 SetScale

C++

```
HRESULT SetScale(int _scale_id, double _val);
```

C#

```
public void SetScale(int _scale_id, double _val);
```

Visual Basic

```
Public Sub SetScale(_scale_id As int, _val As double)
```

Description

The function assigns a given value to the scale.

#### I.2.4.3.14 SetSize

C++

```
HRESULT SetSize(double _width_in_mm, double _height_in_mm);
```

C#

```
public void SetSize(double _width_in_mm, double _height_in_mm);
```

Visual Basic

```
Public Sub SetSize(_width_in_mm As double, _height_in_mm As double)
```

Description

The function sets the view size in mm .

#### I.2.4.3.15 SetWorkPoint

C++

```
HRESULT SetWorkPoint(IRobotGeoPoint3D* _point);
```

C#

```
public void SetWorkPoint(IRobotGeoPoint3D _point);
```

Visual Basic

```
Public Sub SetWorkPoint(ByRef _point As IRobotGeoPoint3D)
```

**Description**

The function sets a working point (for the XY plane Z level is set). .

**I.2.4.3.16 SetZoom****C++**

```
HRESULT SetZoom(double _left, double _top, double _right, double _bottom);
```

**C#**

```
public void SetZoom(double _left, double _top, double _right, double _bottom);
```

**Visual Basic**

```
Public Sub SetZoom(_left As Double, _top As Double, _right As Double, _bottom As Double)
```

**Description**

The function sets the current zoom - coordinates of view sides in the current coordinate system .

**I.2.5 IRobotViewProjection****C++**

```
enum IRobotViewProjection;
```

**C#**

```
public enum IRobotViewProjection;
```

**Visual Basic**

```
Public Enum IRobotViewProjection
```

**Members**

Members	Description
I_VP_XY_3D = 4	Deep projection (XY plane).
I_VP_YZ_3D = 5	Deep projection (YZ plane).
I_VP_XZ_3D = 6	Deep projection (XZ plane).

**Description**

The set of identifiers determining different types of projections to be presented in graphical views has been defined. .

**I.2.6 IRobotViewVisibilityStatusType****C++**

```
enum IRobotViewVisibilityStatusType;
```

**C#**

```
public enum IRobotViewVisibilityStatusType;
```

**Visual Basic**

```
Public Enum IRobotViewVisibilityStatusType
```

**Members**

Members	Description
I_VVST_STRUCTURE_3D = 0	3D structure.
I_VVST_SECTIONS = 1	Sections.

I_VVST_ANIMATION = 2	Animation.
I_VVST_SUPERPOS = 3	Case superposition.
I_VVST_MODE = 4	Modes.
I_VVST_BUCKLING = 5	Buckling.
I_VVST_PLOT = 6	A plotter drawing.
I_VVST_SCALE = 7	Scale.
I_VVST_FILL = 8	A method of area filling.
I_VVST_ACTIVE_CASE = 9	A number of active case.
I_VVST_MOBILE = 10	Settings for moving loads.
I_VVST_ACTIVE_MODE = 11	A number of active mode.
I_VVST_ACTIVE_PHASE = 12	An active structure phase.
I_VVST_ACTIVE_QCMB = 13	Square combination mode.
I_VVST_STRUCTURE = 14	Structure.
I_VVST_NODES = 15	Node numbers.
I_VVST_ELE = 16	Element numbers.
I_VVST_NOT_SELECTED = 17	Objects not selected.
I_VVST_EXPLODE = 18	Explode.
I_VVST_MTC = 19	Materials, compression, tension.
I_VVST_SECTION_NAMES = 20	Section names.
I_VVST_SECTION_SYMB = 21	Section symbols.
I_VVST_SECTION_DRAW = 22	Section drawings.
I_VVST_SECTION_SURF = 23	Section area.
I_VVST_LOCAL = 24	Local systems.
I_VVST_THICKNESS = 25	Thicknesses.
I_VVST_SUPPORTS = 26	Supports.
I_VVST_RELEASES = 27	Releases.
I_VVST_RIGID_LINKS = 28	Rigid links.
I_VVST_LOADS = 29	Loads.
I_VVST_NODE_LOADS = 30	Nodal loads.
I_VVST_BAR_LOADS = 31	Bar loads.
I_VVST_DEFORMATION = 32	Deformed structure.
I_VVST_FORCES = 33	Forces.
I_VVST_EXTREMES = 34	Extreme forces.
I_VVSTREACTIONS = 41	Residual reactions and vectors.
I_VVST_FE = 42	Panels and finite elements.
I_VVST_STRESSES = 43	Stress curvatures.
I_VVST_FE_CUTS = 44	Cuts through finite elements.
I_VVST_FE_DIRECTION = 45	A direction of result calculations in finite elements.
I_VVST_FE_LAY = 46	A layer of stress calculations in finite elements.
I_VVST_REINF_DIR = 47	Main direction of reinforcement.
I_VVST_BAR_MAPS = 48	Maps on bars.
I_VVST_TITLE = 49	Title.
I_VVST_OFFSETS = 50	Offsets.
I_VVST_COMPATIBILITIES = 51	Compatibilities.
I_VVST_ENVELOPES = 52	Envelopes.
I_VVST_ANALYSIS = 53	Analysis.
I_VVST_DIM_LINES = 54	Dimension lines.
I_VVST_REINF_CROSSES = 55	Reinforcement crosses.
I_VVST_STRESSES_GLOBAL = 56	Available since version 1.7.

I_VVST_DESCRIPTIONS = 57	Available since version 2.0.
I_VVST_GRID = 58	Available since version 2.0.
I_VVST_RULER = 59	Available since version 2.0.
_VVST_STRUCTURAL_AXIS = 60	Available since version 2.0.
I_VVST_MAPS = 61	Available since version 2.0.
I_VVST_MAPS_LAYER = 62	Available since version 2.0.
I_VVST_MAPS_DEFORMATION = 63	Available since version 2.0.

**Description**

Available types of visibility status for a graphical viewer.

**I.2.7 IRobotViewVisibilityStatusValue****C++**

```
enum IRobotViewVisibilityStatusValue;
```

**C#**

```
public enum IRobotViewVisibilityStatusValue;
```

**Visual Basic**

```
Public Enum IRobotViewVisibilityStatusValue
```

**Members**

Members	Description
I_VVSV_MTC_MATERIAL = 0x01	Material.
I_VVSV_MTC_COMPRESSION = 0x02	Compression.
I_VVSV_MTC_TENSION = 0x04	Tension.
I_VVSV_MTC_CABLE = 0x08	Cables.
I_VVSV_STRUCTURE_ON = 0x01	Structure geometry.
I_VVSV_STRUCTURE_HDL = 0x04	Hidden lines.
I_VVSV_STRUCTURE_SHD = 0x10	Shading.
I_VVSV_STRUCTURE_HDF = 0x40	Colorful sides, black edges.
I_VVSV_STRUCTURE_HDR = 0x80	Shading with edge correction. .
I_VVSV_SUPPORTS_SYMB = 0x01	Support symbols.
I_VVSV_SUPPORTS_COD = 0x02	Support codes.
I_VVSV_SUPPORTS_LBL = 0x04	Labels for all visible attributes (instead of codes).
I_VVSV_ANALYSIS_DET = 0x01	Detailed analysis.
I_VVSV_ANALYSIS_GLO = 0x02	Global analysis.
I_VVSV_ANALYSIS_INF = 0x08	Analysis of influence lines .
I_VVSV_ENVELOPES_MIN = 0x01	Minimum envelope.
I_VVSV_ENVELOPES_MAX = 0x02	Maximum envelope.
I_VVSV_ENVELOPES_CMP = 0x04	All envelope components.
I_VVSV_ENVELOPES_ONE = 0x08	Single components in complex cases.
I_VVSV_ENVELOPES_HST = 0x10	Time history analysis case.
I_VVSV_ENVELOPES_CUT = 0x20	The analysis in cuts.
I_VVSV_ENVELOPES_SEU = 0x40	The analysis of a single bar.
I_VVSV_SECTION_DRAW_SHAPE = 0x01	Section shapes.
I_VVSV_SECTION_DRAW_COLOR = 0x10	Section colors.
I_VVSV_SECTION_DRAW_TOP_BOTTOM = 0x20	Top and bottom of elements in colors.
I_VVSV_SECTION_DRAW_GROUP = 0x40	Group colors.

I_VVSV_SECTION_SYMB_SYMB = 0x1	Section symbols.
I_VVSV_SECTION_SYMB_ELA = 0x2	Elastic ground for bars.
I_VVSV_FILL_FILL = 0x1	Map filling.
I_VVSV_FILL_AVG = 0x2	Averaging values in finite elements.
I_VVSV_FILL_VAL = 0x4	Values on maps.
I_VVSV_FILL_ISO = 0x8	Isolines.
I_VVSV_FILL_BOR = 0x10	Contour.
I_VVSV_FILL_TAN = 0x20	Cuts on normal plane.
I_VVSV_FILL_DEF = 0x40	Deformation.
I_VVSV_FILL_SMOOTH_NON = 0	No smoothing.
I_VVSV_FILL_SMOOTH_GLO = 0x100	Global smoothing.
I_VVSV_FILL_SMOOTH_LOC = 0x200	Smoothing within panels.
I_VVSV_FILL_SMOOTH_SEL = 0x300	Smoothing on selection.
I_VVSV_RIGID_LINKS_RIG = 0x1	Rigid links.
I_VVSV_RIGID_LINKS_SYM = 0x2	Rigid links - symbols.
I_VVSV_RIGID_LINKS_DET = 0x4	Rigid links - details.
I_VVSV_RIGID_LINKS_SUR = 0x8	Rigid links - areas.
I_VVSV_FE_CON = 0x1	Contours.
I_VVSV_FE_COI = 0x2	Inside part of contours .
I_VVSV_FE_COL = 0x4	Thickness colors.
I_VVSV_FE_COE = 0x8	Contour components.
I_VVSV_FE_MES = 0x10	Mesh.
I_VVSV_FE_EMI = 0x20	Emitters.
I_VVSV_FE_RES = 0x40	Results.
I_VVSV_FE_DSC = 0x80	Description of panels.
I_VVSV_FE_DSO = 0x100	Description of objects.
I_VVSV_FE_EDG = 0x200	Description of edges.
I_VVSV_FE_ADV = 0x400	Detailed description of objects.
I_VVSV_ELE_BAR_NUM = 0x1	Bar numbers.
I_VVSV_ELE_CAL_NUM = 0x2	Calculation element numbers.
I_VVSV_ELE_FE_NUM = 0x4	Finite element numbers.
I_VVSV_ELE_FE_VIS = 0x8	Finite elements.
I_VVSV_MOBILE_ROUTE = 0x1	Route.
I_VVSV_MOBILE_CAR = 0x2	Vehicle.
I_VVSV_MOBILE_ELE = 0x4	Elements.
I_VVSV_MOBILE_LOADS = 0x8	Loads.
I_VVSV_STRUCTURE_HDY = 0x100	Quick shading of faces Available since version 1.7.
I_VVSV_FILL_SMOOTH_CAR = 0x400	Available since version 1.7.
I_VVSV_FORCES_FX = 0x0001	Available since version 1.7.
I_VVSV_FORCES_FY = 0x0002	Available since version 1.7.
I_VVSV_FORCES_FZ = 0x0004	Available since version 1.7.
I_VVSV_FORCES_FXC = 0x0008	Available since version 1.7.
I_VVSV_FORCES_MX = 0x0010	Available since version 1.7.
I_VVSV_FORCES_MY = 0x0020	Available since version 1.7.
I_VVSV_FORCES_MZ = 0x0040	Available since version 1.7.
I_VVSV_FORCES_UX = 0x0100	Available since version 1.7.
I_VVSV_FORCES_UY = 0x0200	Available since version 1.7.
I_VVSV_FORCES_UZ = 0x0400	Available since version 1.7.
I_VVSV_FORCES_DFL = 0x1000	Available since version 1.7.

I_VVSV_STRESSES_NORMALMIN = 0x0001	Available since version 1.7.
I_VVSV_STRESSES_NORMALMAX = 0x0002	Available since version 1.7.
I_VVSV_STRESSES_FLEXMIN_MY = 0x0004	Available since version 1.7.
I_VVSV_STRESSES_FLEXMIN_MZ = 0x0008	Available since version 1.7.
I_VVSV_STRESSES_FLEXMAX_MY = 0x0010	Available since version 1.7.
I_VVSV_STRESSES_FLEXMAX_MZ = 0x0020	Available since version 1.7.
I_VVSV_STRESSES_AXIAL = 0x0040	Available since version 1.7.
I_VVSV_STRESSES_SHEAR_Y = 0x0080	Available since version 1.7.
I_VVSV_STRESSES_SHEAR_Z = 0x0100	Available since version 1.7.
I_VVSV_STRESSES_TORSION = 0x0200	Available since version 1.7.
I_VVSV_DEFORMATION_EXACT = 0x0001	Available since version 1.7.
I_VVSV_DEFORMATION_STD = 0x0002	Available since version 1.7.
I_VVSVREACTIONS_F = 0x01	Available since version 1.7.
I_VVSVREACTIONS_M = 0x02	Available since version 1.7.
I_VVSVREACTIONS_RESID_F = 0x04	Available since version 1.7.
I_VVSVREACTIONS_RESID_M = 0x08	Available since version 1.7.
I_VVSVREACTIONS_VAL = 0x10	Available since version 1.7.
I_VVSVREACTIONS_ROT = 0x20	Available since version 1.7.
I_VVSVREACTIONS_DIV = 0x40	Available since version 1.7.
I_VVSVLOADS_SYMBOL = 0x01	Available since version 1.7.
I_VVSVLOADS_VALUE = 0x02	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_USER_MIN = 0x001	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_USER_MAX = 0x002	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_NORMAL_MIN = 0x004	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_NORMAL_MAX = 0x008	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_TAU_MIN = 0x010	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_TAU_MAX = 0x020	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_MISES_MIN = 0x040	Available since version 1.7.
I_VVSV_STRESSES_GLOBAL_MISES_MAX = 0x080	Available since version 1.7.
I_VVSV_BAR_MAPS_FX = 165	Available since version 1.7.
I_VVSV_BAR_MAPS_FY = 166	Available since version 1.7.
I_VVSV_BAR_MAPS_FZ = 167	Available since version 1.7.
I_VVSV_BAR_MAPS_MX = 168	Available since version 1.7.
I_VVSV_BAR_MAPS_MY = 169	Available since version 1.7.
I_VVSV_BAR_MAPS_MZ = 170	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MAX = 171	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MIN = 172	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_FX_AX = 173	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MAX_MY = 174	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MAX_MZ = 175	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MIN_MY = 176	Available since version 1.7.
I_VVSV_BAR_MAPS_STRESS_S_MIN_MZ = 177	Available since version 1.7.
I_VVSV_BAR_MAPS_SHEAR_STRESS_TY = 179	Available since version 1.7.
I_VVSV_BAR_MAPS_SHEAR_STRESS_TZ = 180	Available since version 1.7.
I_VVSV_BAR_MAPS_SHEAR_STRESS_T = 181	Available since version 1.7.
I_VVSV_BAR_MAPS DESIGN_RATIO = 796	Available since version 1.7.
I_VVSV_BAR_MAPS DESIGN_MEMB_LENGTH = 271	Available since version 1.7.
I_VVSV_BAR_MAPS DESIGN_SLEND_LAY = 831	Available since version 1.7.
I_VVSV_BAR_MAPS DESIGN_SLEND_LA = 832	Available since version 1.7.

I_VVSV_DESCRIPTIONS_NONE = 0	Available since version 2.0.
I_VVSV_DESCRIPTIONS_HORIZONTAL = 0x01	Available since version 2.0.
I_VVSV_DESCRIPTIONS_VERTICAL = 0x02	Available since version 2.0.
I_VVSV_REINFORCEMENT_A1 = 0x0001	Available since version 2.0.
I_VVSV_REINFORCEMENT_A1R = 0x0010	Available since version 2.0.
I_VVSV_REINFORCEMENT_A2 = 0x0002	Available since version 2.0.
I_VVSV_REINFORCEMENT_A2R = 0x0020	Available since version 2.0.
I_VVSV_REINFORCEMENT_PRC = 0x0004	Available since version 2.0.
I_VVSV_REINFORCEMENT_PRCR = 0x0008	Available since version 2.0.
I_VVSV_REINFORCEMENT_DISTRIB = 0x0040	Available since version 2.0.
I_VVSV_REINFORCEMENT_DISTRIBR = 0x0080	Available since version 2.0.
I_VVSV_REINFORCEMENT_A1BARS = 0x0100	Available since version 2.0.
I_VVSV_REINFORCEMENT_A2BARS = 0x0200	Available since version 2.0.
I_VVSV_MAPS_DETAILED_STRESSES_XX = 483	Available since version 2.0.
I_VVSV_MAPS_DETAILED_STRESSES YY = 484	Available since version 2.0.
I_VVSV_MAPS_DETAILED_STRESSES_XY = 485	Available since version 2.0.
I_VVSV_MAPS_DETAILED_STRESSES_Z = 486	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MEMB_FORCES_XX = 492	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MEMB_FORCES_YY = 493	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MEMB_FORCES_XY = 494	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MOMENTS_XX = 501	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MOMENTS_YY = 502	Available since version 2.0.
I_VVSV_MAPS_DETAILED_MOMENTS_XY = 503	Available since version 2.0.
I_VVSV_MAPS_DETAILED_SHEAR_STRESSES_XX = 510	Available since version 2.0.
I_VVSV_MAPS_DETAILED_SHEAR_STRESSES_YY = 511	Available since version 2.0.
I_VVSV_MAPS_DETAILED_SHEAR_FORCES_XX = 519	Available since version 2.0.
I_VVSV_MAPS_DETAILED_SHEAR_FORCES_YY = 520	Available since version 2.0.
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_XX = 537	Available since version 2.0.
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_YY = 538	Available since version 2.0.
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_Z = 531	Available since version 2.0.
I_VVSV_MAPS_DETAILED_ROTATION_XX = 546	Available since version 2.0.
I_VVSV_MAPS_DETAILED_ROTATION_YY = 547	Available since version 2.0.
I_VVSV_MAPS_DETAILED_ROTATION_Z = 549	Available since version 2.0.
I_VVSV_MAPS_DETAILED_SOILREACTIONS = 558	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_STRESSES_1 = 487	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_STRESSES_2 = 488	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_STRESSES_1_2 = 490	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_STRESSES_ANGLE = 489	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_1 = 496	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_2 = 497	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_1_2 = 499	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_ANGLE = 498	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MOMENTS_1 = 505	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MOMENTS_2 = 506	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MOMENTS_ANGLE = 507	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_MOMENTS_1_2 = 508	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_SHEAR_STRESSES = 517	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_SHEAR_FORCES = 526	Available since version 2.0.

I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_X = 1302	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_Y = 1303	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_Z = 1304	Available since version 2.0.
I_VVSV_MAPS_PRINCIPAL_TOTAL_DISP = 1305	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_STRESSES = 491	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_MEMB_FORCEES = 500	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_MOMENTS = 509	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_TOP_MXX = 889	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_TOP_MYY = 891	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_BOTTOM_MXX = 890	Available since version 2.0.
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_BOTTOM_MYY = 892	Available since version 2.0.
I_VVSV_MAPS_CROSS_S = 1404	Available since version 2.0.
I_VVSV_MAPS_CROSS_N = 1405	Available since version 2.0.
I_VVSV_MAPS_CROSS_M = 1406	Available since version 2.0.
I_VVSV_MAPS_LAYER_UPPER = 0	Available since version 2.0.
I_VVSV_MAPS_LAYER_MIDDLE = 1	Available since version 2.0.
I_VVSV_MAPS_LAYER_LOWER = 2	Available since version 2.0.
I_VVSV_MAPS_LAYER_MAXIMUM = 3	Available since version 2.0.
I_VVSV_MAPS_LAYER_MINIMUM = 4	Available since version 2.0.
I_VVSV_MAPS_LAYER_ABS_MINIMUM = 5	Available since version 2.0.
I_VVSV_MAPS_LAYER_ARBITRARY = 6	Available since version 2.0.
I_VVSV_MAPS_ISOLINES = 0x0001	Available since version 2.0.
I_VVSV_MAPS_DESCRIPTION = 0x0002	Available since version 2.0.
I_VVSV_MAPS_SMOOTH_NON = 0x0004	Available since version 2.0.
I_VVSV_MAPS_SMOOTH_GLO = 0x0008	Available since version 2.0.
I_VVSV_MAPS_SMOOTH_LOC = 0x0010	Available since version 2.0.
I_VVSV_MAPS_SMOOTH_SEL = 0x0020	Available since version 2.0.
I_VVSV_MAPS_SMOOTH_CAR = 0x0030	Available since version 2.0.
I_VVSV_FORCES_BAR.REACT_KY = 0x0080	Available since version 3.
I_VVSV_FORCES_BAR.REACT_KZ = 0x0800	Available since version 3.
I_VVSV_FILL_FE = 0x80	Finite element mesh. Available since version 4.1.
I_VVSVREACTIONS_FX = 0x0080	.
I_VVSVREACTIONS_FY = 0x100	.
I_VVSVREACTIONS_FZ = 0x200	.
I_VVSVREACTIONS_MX = 0x400	.
I_VVSVREACTIONS_MY = 0x800	.
I_VVSVREACTIONS_PSEUDO_F = 0x2000	.
I_VVSVREACTIONS_PSEUDO_M = 0x4000	.

I_VVSV_REACTIONS_MZ = 0x1000	.
------------------------------	---

Available since version 15.2.
-------------------------------

## Description

Accessible values for particular types of visibility status.

## I.2.8 IRobotViewScaleType

### C++

```
enum IRobotViewScaleType;
```

### C#

```
public enum IRobotViewScaleType;
```

### Visual Basic

```
Public Enum IRobotViewScaleType
```

### Members

Members	Description
I_VST_FX = 0	A scale for FX force.
I_VST_FY = 1	A scale for FY force .
I_VST_FZ = 2	A scale for FZ force
.	.
I_VST_MX = 3	A scale for MX .
I_VST_MY = 4	A scale for MY
.	.
I_VST_MZ = 5	A scale for MZ
.	.
I_VST_DEFORM = 6	A scale for deformation.
I_VST_SIG = 7	A scale for axial stresses.
I_VST_TAU = 8	A scale for shear stresses .

## Description

Available types of a scale defined for a graphical viewer.

## I.2.9 IRobotViewDetailedAnalysis

### Class Hierarchy

### C++

```
interface IRobotViewDetailedAnalysis : IRobotView;
```

### C#

```
public interface IRobotViewDetailedAnalysis : IRobotView;
```

### Visual Basic

```
Public Interface IRobotViewDetailedAnalysis
```

## Description

View presenting results of the detailed analysis. Available since version 2.0.

### I.2.9.1 IRobotViewDetailedAnalysis Members

The following tables list the members exposed by IRobotViewDetailedAnalysis.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CurrentTableTab ( [ see page 1407)	Identifier of the active tab in the tabular part of the view.
◆	DiagramMagnification ( [ see page 1391)	Diagram enlarging (scale: 1 to 10).
◆	ParamsBarMap ( [ see page 1391)	Parameters of displaying maps an bars Available since version 2.0.
◆	ParamsDetailed ( [ see page 1407)	Parameters of displaying detailed analysis results Available since version 2.0.
◆	ParamsDiagram ( [ see page 1391)	Parameters of displaying diagrams on the view Available since version 2.0.
◆	ParamsDisplay ( [ see page 1391)	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
◆	ParamsFeMap ( [ see page 1392)	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
◆	Printable ( [ see page 1392)	Object representing the view that may be attached to the printout and printed .
◆	Projection ( [ see page 1392)	Current projection .
◆	Selection ( [ see page 1392)	Selection of structure components presented in the view .
◆	Title ( [ see page 1393)	View title .
◆	Type ( [ see page 1393)	View type .
◆	UserControl ( [ see page 1408)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
◆	Visible ( [ see page 1393)	View display flag .
◆	Window ( [ see page 1408)	

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡◆	CopyToClipboard ( [ see page 1394)	The function inserts the view into the clipboard.
≡◆	GetRotationPoint ( [ see page 1394)	The function gives back a rotation center during a view rotation.
≡◆	GetScale ( [ see page 1394)	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
≡◆	GetSize ( [ see page 1395)	The function returns the size of the view in mm. .
≡◆	GetWorkPoint ( [ see page 1395)	The function gives a work point back (on the XY plane it gives Z level) .
≡◆	GetZoom ( [ see page 1395)	The function returns the current zoom - coordinates of view sides in the current coordinate system .
≡◆	IsLocal ( [ see page 1396)	The function gives the status of a local coordinate system back.
≡◆	MakeScreenCapture ( [ see page 1408)	
≡◆	Redraw ( [ see page 1396)	The function draws the view. .
≡◆	Rotate ( [ see page 1396)	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
≡◆	SetGlobal ( [ see page 1396)	The function sets a global coordinate system.
≡◆	SetLocal ( [ see page 1397)	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
≡◆	SetRotationPoint ( [ see page 1397)	The funcion sets the rotation center during the view rotation.
≡◆	SetScale ( [ see page 1397)	The function assigns a given value to the scale.
≡◆	SetSize ( [ see page 1397)	The function sets the view size in mm .
≡◆	SetWorkPoint ( [ see page 1398)	The function sets a working point (for the XY plane Z level is set). .
≡◆	SetZoom ( [ see page 1398)	The function sets the current zoom - coordinates of view sides in the current coordinate system .

## I.2.9.2 IRobotViewDetailedAnalysis Fields

The fields of the IRobotViewDetailedAnalysis class are listed here.

### Public Fields

	Name	Description
◆	CurrentTableTab (see page 1407)	Identifier of the active tab in the tabular part of the view.
◆	ParamsDetailed (see page 1407)	Parameters of displaying detailed analysis results Available since version 2.0.
◆	UserControl (see page 1408)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
◆	Window (see page 1408)	

### I.2.9.2.1 CurrentTableTab

#### C++

```
HRESULT get_CurrentTableTab(IRobotViewDetailedAnalysisTableTab* );
HRESULT put_CurrentTableTab(IRobotViewDetailedAnalysisTableTab);
```

#### C#

```
public IRobotViewDetailedAnalysisTableTab CurrentTableTab { get; set; }
```

#### Visual Basic

```
Public CurrentTableTab As IRobotViewDetailedAnalysisTableTab
```

#### Description

Identifier of the active tab in the tabular part of the view.

### I.2.9.2.2 ParamsDetailed

#### C++

```
HRESULT get_ParamsDetailed(IRobotViewDetailedAnalysisParams** );
```

#### C#

```
public IRobotViewDetailedAnalysisParams ParamsDetailed { get; }
```

#### Visual Basic

```
Public ReadOnly ParamsDetailed As IRobotViewDetailedAnalysisParams
```

#### Description

Parameters of displaying detailed analysis results Available since version 2.0.

### I.2.9.2.3 UserControl

#### C++

```
HRESULT get_UserControl(VARIANT_BOOL* );
HRESULT put_UserControl(VARIANT_BOOL);
```

#### C#

```
public bool UserControl { get; set; }
```

#### Visual Basic

```
Public UserControl As Boolean
```

#### Description

Flag that enables giving/taking back graphical view navigation to/from the interactive program user .

**Version**

Available since version 3.

**I.2.9.2.4 Window****C++**

```
HRESULT get_Window(IRobotWindow**);
```

**C#**

```
public IRobotWindow Window { get; }
```

**Visual Basic**

```
Public ReadOnly Window As IRobotWindow
```

**Version**

Available since version 3.

**I.2.9.3 IRobotViewDetailedAnalysis Methods**

The methods of the IRobotViewDetailedAnalysis class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	MakeScreenCapture (see page 1408)	

**I.2.9.3.1 MakeScreenCapture****C++**

```
HRESULT MakeScreenCapture(IRobotViewScreenCaptureParams* _sc_params);
```

**C#**

```
public void MakeScreenCapture(IRobotViewScreenCaptureParams _sc_params);
```

**Visual Basic**

```
Public Sub MakeScreenCapture(ByRef _sc_params As IRobotViewScreenCaptureParams)
```

**I.2.10 IRobotViewDetailedAnalysisTableTab****C++**

```
enum IRobotViewDetailedAnalysisTableTab;
```

**C#**

```
public enum IRobotViewDetailedAnalysisTableTab;
```

**Visual Basic**

```
Public Enum IRobotViewDetailedAnalysisTableTab
```

**Members**

<b>Members</b>	<b>Description</b>
I_VDATT_LOCAL_EXTREMES = 1	Tab presenting local extreme values.
I_VDATT_GLOBAL_EXTREMES = 2	Tab presenting global extreme values.

**Description**

Identifiers of the tabs in the table being a part of the view presenting detailed analysis results. Available since version 2.0.

## I.2.11 IRobotViewDisplayParams

### Class Hierarchy

#### C++

```
interface IRobotViewDisplayParams : IDispatch;
```

#### C#

```
public interface IRobotViewDisplayParams;
```

#### Visual Basic

```
Public Interface IRobotViewDisplayParams
```

### Description

Parameters of display for graphical viewers.

### I.2.11.1 IRobotViewDisplayParams Members

The following tables list the members exposed by IRobotViewDisplayParams.

#### Public Fields

	Name	Description
◆	HiddenLines (see page 1410)	Type of displaying hidden lines Available since version 2.0.
◆	SymbolSize (see page 1410)	Symbol size (from 1 to 10) Available since version 2.0.

#### Public Methods

	Name	Description
♫	IsOn (see page 1410)	Function returns a value different from zero (True) if a selected attribute is switched on. Available since version 2.0.
♫	Set (see page 1411)	Function switches on/off a selected visibility attribute. Available since version 2.0.

### I.2.11.2 IRobotViewDisplayParams Fields

The fields of the IRobotViewDisplayParams class are listed here.

#### Public Fields

	Name	Description
◆	HiddenLines (see page 1410)	Type of displaying hidden lines Available since version 2.0.
◆	SymbolSize (see page 1410)	Symbol size (from 1 to 10) Available since version 2.0.

### I.2.11.2.1 HiddenLines

#### C++

```
HRESULT get_HiddenLines(IRobotViewHiddenLinesDisplayType* );
HRESULT put_HiddenLines(IRobotViewHiddenLinesDisplayType);
```

#### C#

```
public IRobotViewHiddenLinesDisplayType HiddenLines { get; set; }
```

#### Visual Basic

```
Public HiddenLines As IRobotViewHiddenLinesDisplayType
```

### Description

Type of displaying hidden lines Available since version 2.0.

### I.2.11.2.2 SymbolSize

#### C++

```
HRESULT get_SymbolSize(short* );
HRESULT put_SymbolSize(short);
```

#### C#

```
public short SymbolSize { get; set; }
```

#### Visual Basic

```
Public SymbolSize As short
```

#### Description

Symbol size (from 1 to 10) Available since version 2.0.

### I.2.11.3 IRobotViewDisplayParams Methods

The methods of the IRobotViewDisplayParams class are listed here.

#### Public Methods

	Name	Description
💡	IsOn (see page 1410)	Function returns a value different from zero (True) if a selected attribute is switched on. Available since version 2.0.
💡	Set (see page 1411)	Function switches on/off a selected visibility attribute. Available since version 2.0.

### I.2.11.3.1 IsOn

#### C++

```
HRESULT IsOn(IRobotViewDisplayAttributes _v_attrib, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsOn(IRobotViewDisplayAttributes _v_attrib);
```

#### Visual Basic

```
Public Function IsOn(_v_attrib As IRobotViewDisplayAttributes) As Boolean
```

#### Description

Function returns a value different from zero (True) if a selected attribute is switched on. Available since version 2.0.

### I.2.11.3.2 Set

#### C++

```
HRESULT Set(IRobotViewDisplayAttributes _v_attrib, VARIANT_BOOL _on_off);
```

#### C#

```
public void Set(IRobotViewDisplayAttributes _v_attrib, bool _on_off);
```

#### Visual Basic

```
Public Sub Set(_v_attrib As IRobotViewDisplayAttributes, _on_off As Boolean)
```

#### Description

Function switches on/off a selected visibility attribute. Available since version 2.0.

## I.2.12 IRobotViewDisplayAttributes

**C++**

```
enum IRobotViewDisplayAttributes;
```

**C#**

```
public enum IRobotViewDisplayAttributes;
```

**Visual Basic**

```
Public Enum IRobotViewDisplayAttributes
```

**Members**

Members	Description
I_VDA_STRUCTURE_STRUCTURE = 1	Available since version 2.0.
I_VDA_STRUCTURE_NODE_NUMBERS = 2	Available since version 2.0.
I_VDA_STRUCTURE_BAR_NUMBERS = 3	Available since version 2.0.
I_VDA_STRUCTURE_SUPPORT_SYMBOLS = 4	Available since version 2.0.
I_VDA_STRUCTURE_SUPPORT_CODES = 5	Available since version 2.0.
I_VDA_STRUCTURE_ONLY_FOR_SELECTED_OBJECTS = 6	Available since version 2.0.
I_VDA_STRUCTURE_GROUP_COLORS = 8	Available since version 2.0.
I_VDA_STRUCTURE_ATTRIBUTE_LABELS = 9	Available since version 2.0.
I_VDA_SECTIONS_NAME = 11	Available since version 2.0.
I_VDA_SECTIONS_COLORS = 12	Available since version 2.0.
I_VDA_SECTIONS_SYMBOLS = 13	Available since version 2.0.
I_VDA_SECTIONS_SHAPE = 14	Available since version 2.0.
I_VDA_SECTIONS_SURFACE = 15	Available since version 2.0.
I_VDA_SECTIONS_MATERIAL = 16	Available since version 2.0.
I_VDA_LOADS_VALUES = 18	Available since version 2.0.
I_VDA_LOADS_MOVING_ROUTE = 19	Available since version 2.0.
I_VDA_LOADS_MOVING_VEHICLE = 20	Available since version 2.0.
I_VDA_LOADS_MOVING_ELEMENTS = 21	Available since version 2.0.
I_VDA_ADVANCED_RELEASE_SYMBOLS = 23	Available since version 2.0.
I_VDA_ADVANCED_RELEASE_CODES = 24	Available since version 2.0.
I_VDA_ADVANCED_OFFSETS = 25	Available since version 2.0.
I_VDA_ADVANCED_COMPATIBLE_NODES = 26	Available since version 2.0.
I_VDA_ADVANCED_RIGID_LINKS = 27	Available since version 2.0.
I_VDA_ADVANCED_RIGID_LINK_SURFACE = 28	Available since version 2.0.
I_VDA_ADVANCED_ELASTIC_FOUNDATION = 29	Available since version 2.0.
I_VDA_ADVANCED_CABLES = 30	Available since version 2.0.
I_VDA_ADVANCED_TENSION_COMPRESSION = 31	Available since version 2.0.
I_VDA_ADVANCED_STEEL_CONNECTIONS = 32	Available since version 2.0.
I_VDA_FE_PANEL_CONTOURS = 33	Available since version 2.0.
I_VDA_FE_PANEL_INTERIOR = 34	Available since version 2.0.
I_VDA_FE_THICKNESS = 35	Available since version 2.0.
I_VDA_FE_PANEL_NUMBERS = 36	Available since version 2.0.
I_VDA_FE_PANEL_COMPLEX_DESC = 37	Available since version 2.0.
I_VDA_FE_COLOR_LEGEND = 38	Available since version 2.0.
I_VDA_FE_CONTOUR_COMPONENTS = 39	Available since version 2.0.
I_VDA_FE_MESH_PREVIEW = 40	Available since version 2.0.

I_VDA_FE_EDGE_NUMBERS = 41	Available since version 2.0.
I_VDA_FEFINITE_ELEMENTS = 42	Available since version 2.0.
I_VDA_FEFINITE_ELEMENT_NUMBERS = 43	Available since version 2.0.
I_VDA_FEEMITTERS = 44	Available since version 2.0.
I_VDA_OTHER_STRUCTURAL_AXIS = 45	Available since version 2.0.
I_VDA_OTHER_DIMENSION_LINES = 46	Available since version 2.0.
I_VDA_OTHER_GRID = 47	Available since version 2.0.
I_VDA_OTHER_RULER = 48	Available since version 2.0.
I_VDA_OTHER_HIDE_NODES = 49	Available since version 2.0.
I_VDA_OTHER_HIDE_INACTIVE = 51	Available since version 2.0.
I_VDA_OTHER_CALC_ELEM_NUMBERS = 52	Available since version 2.0.
I_VDA_OTHER_CALC_POINT_NUMBERS = 53	Available since version 2.0.
I_VDA_OTHER_OBJECTS_OUT_OF_PLANE = 54	Available since version 2.0.
I_VDA_SECTIONS_MEMBER_TYPE_NAME = 55	Bar type - name. Available since version 3.5.
I_VDA_SECTIONS_MEMBER_TYPE_LEGEND = 56	Bar type - color legend. Available since version 3.5.
I_VDA_ADVANCED_GEOIMPERFECTIONS = 57	Geometrical imperfections. Available since version 3.5.
I_VDA_ADVANCED_GEOIMPERFECTIONS_NAME = 58	Geometrical imperfections - name. Available since version 3.5.
I_VDA_ADVANCED_NONLINEAR_HINGES = 59	Non-linear hinges. Available since version 3.5.
I_VDA_STRUCTURE_SUPPORT_SHAPES = 60	Support shapes. Available since version 9.7.
I_VDA_STRUCTURE_EXPLODE_BARS = 61	Available since version 9.7.
I_VDA_STRUCTURE_EXPLODE_FE = 62	Available since version 9.7.
I_VDA_STRUCTURE_LOCAL_SYSTEM_BARS = 63	Available since version 9.7.
I_VDA_STRUCTURE_LOCAL_SYSTEM_PANELS = 64	Available since version 9.7.
I_VDA_STRUCTURE_LOCAL_SYSTEM_FE = 65	Available since version 9.7.
I_VDA_STRUCTURE_BAR_NAMES = 66	Available since version 9.7.
I_VDA_SECTIONS_CODE_GROUPS = 67	Available since version 9.7.
I_VDA_LOADS_SYMBOLS_CONCENTRATED = 68	Symbols of concentrated loads. Available since version 9.7.
I_VDA_LOADS_SYMBOLS_LINEAR = 69	Symbols of linear loads. Available since version 9.7.
I_VDA_LOADS_SYMBOLS_PLANAR = 70	Symbols of planar loads. Available since version 9.7.
I_VDA_LOADS_SYMBOLS_UNIFORM_SIZE = 71	Uniform size of load symbols. Available since version 9.7.
I_VDA_LOADS_AUTOMATICALLY = 72	Automatically-generated forces. Available since version 9.7.
I_VDA_LOADS_DISTRIBUTION_REGIONS = 73	Regions of load distribution. Available since version 9.7.
I_VDA_STRUCTURE_SUPPORT_DETAILED_SYMBOLS = 74	Detailed support symbols. Available since version 9.7.
I_VDA_STRUCTURE_SUPPORT_DIRECTION = 75	Symbol of support direction. Available since version 9.7.
I_VDA_ADVANCED_STORIES_COLORS = 76	Color designation of stories. Available since version 9.7.

I_VDA_ADVANCED_STORIES_RESULTS = 77	Results for stories. Available since version 9.7.
I_VDA_ADVANCED_RC_BEAM = 78	RC beams. Available since version 9.7.
I_VDA_ADVANCED_RC_COLUMN = 79	RC columns. Available since version 9.7.
I_VDA_ADVANCED_RC_FOUNDATION = 80	RC foundations. Available since version 9.7.
I_VDA_ADVANCED_RC_CONTINOUS_FOOTING = 81	RC continuous footings. Available since version 9.7.
I_VDA_ADVANCED_RC_WALL = 82	RC walls. Available since version 9.7.
I_VDA_ADVANCED_RC_DEEP_BEAM = 83	RC deep beams. Available since version 9.7.
I_VDA_ADVANCED_STEEL_CONNECTION_NUMBERS = 84	Numbers of steel connections. Available since version 9.7.
I_VDA_ADVANCED_STEEL_CONNECTION_NAMES = 85	Names of steel connections. Available since version 9.7.
I_VDA_ADVANCED_STEEL_CONNECTION_TYPES = 86	Types of steel connections. Available since version 9.7.
I_VDA_FE_CLADDING_INTERIOR = 87	Filling of claddings. Available since version 9.7.
I_VDA_FE_FE_INTERIOR = 88	Filling of FEs. Available since version 9.7.
I_VDA_FE_PANEL_THICKNESSES = 89	Panel thickness. Available since version 9.7.
I_VDA_FE_PANEL_REINFORCEMENT = 90	Panel reinforcement. Available since version 9.7.
I_VDA_FE_PANEL_NAMES = 91	Panel names. Available since version 9.7.
I_VDA_FE_LOAD_DIRECTION = 93	Direction of load distribution. Available since version 9.7.
I_VDA_FE_CHARACTERISTIC_POINTS = 94	Characteristic points. Available since version 9.7.
I_VDA_FE_AUXILIARY_OBJECTS = 95	Auxiliary objects. Available since version 9.7.
I_VDA_FE_CLADDING_COLORS = 96	Colors of claddings. Available since version 9.7.
I_VDA_OTHER_STRUCTURAL_AXIS_DESCRIPTION = 97	Labels of structure axes. Available since version 9.7.
I_VDA_VIEWOGL_HIDE_INVISIBLE_LINES = 98	Hide invisible lines. Available since version 9.7.
I_VDA_VIEWOGL_COLOR_FACES = 99	Color faces. Available since version 9.7.
I_VDA_VIEWOGL_LIGHT = 100	Light. Available since version 9.7.
I_VDA_VIEWOGL_LIGHT_ON_MAPS = 101	Light on maps. Available since version 9.7.
I_VDA_VIEWOGL_TRANSLUCENT = 102	Translucent faces. Available since version 9.7.
I_VDA_VIEWOGL_BLACK_EDGES = 103	Black edges. Available since version 9.7.

I_VDA_VIEWOGL_ANTYALIASING = 104	Antialiasing. Available since version 9.7.
I_VDA_VIEWOGL_HIDE_XY_CUTTING_PLANES = 105	Hide XY cutting plane. Available since version 9.7.
I_VDA_VIEWOGL_HIDE_XZ_CUTTING_PLANES = 106	Hide XZ cutting plane. Available since version 9.7.
I_VDA_VIEWOGL_HIDE_YZ_CUTTING_PLANES = 107	Hide YZ cutting plane. Available since version 9.7.
I_VDA_VIEWOGL_DRAW_OUT_OF_SCREEN = 108	Draw objects positioned out of screen. Available since version 9.7.
I_VDA_VIEWOGL_REDRAW = 109	Full redrawing after modification. Available since version 9.7.
I_VDA_VIEWOGL_DETAILS = 110	Draw details during structure viewing. Available since version 9.7.
I_VDA_VIEW_HIDDEN_LINES = 111	Hidden lines. Available since version 9.7.
I_VDA_VIEW_COLOR = 112	Colored sides. Available since version 9.7.
I_VDA_VIEW_SHADING = 113	Shading. Available since version 9.7.
I_VDA_VIEW_SHADING_EDGES = 114	Shading/edges. Available since version 9.7.
I_VDA_VIEW_QUICK_SHADING = 115	Quick shading of faces. Available since version 9.7.
I_VDA_VIEW_NONE = 116	Available since version 9.7.
I_VDA_LOADS_PRESSURE_MAP = 117	Available since version 14.5.

**Description**

Set of attributes whose visibility may be switched on/off for graphical views. .

**I.2.13 IRobotViewHiddenLinesDisplayType****C++**

```
enum IRobotViewHiddenLinesDisplayType;
```

**C#**

```
public enum IRobotViewHiddenLinesDisplayType;
```

**Visual Basic**

```
Public Enum IRobotViewHiddenLinesDisplayType
```

**Members**

Members	Description
I_VHLDT_HIDDEN_LINES = 0x04	Available since version 2.0.
I_VHLDT_SHADING = 0x10	Available since version 2.0.
I_VHLDT_COLOR_SIDES_EDGES = 0x40	Available since version 2.0.
I_VHLDT_SHADING_EDGES = 0x80	Available since version 2.0.
I_VHLDT_QUICK_SHADING_OF_FACES = 0x100	Available since version 2.0.
I_VHLDT_NONE = 0	Available since version 2.0.

**Description**

Possible types of displaying hidden lines.

## I.2.14 IRobotViewDiagramParams

### Class Hierarchy

C++

```
interface IRobotViewDiagramParams : IDispatch;
```

C#

```
public interface IRobotViewDiagramParams;
```

### Visual Basic

```
Public Interface IRobotViewDiagramParams
```

### Description

Interface controlling the manner of displaying diagrams on bars. .

### I.2.14.1 IRobotViewDiagramParams Members

The following tables list the members exposed by IRobotViewDiagramParams.

#### Public Fields

	Name	Description
◆	Descriptions (↗ see page 1416)	The manner of locating the diagram description Available since version 2.0.
◆	Filling (↗ see page 1416)	Type of diagram filling Available since version 2.0.
◆	PositiveNegative (↗ see page 1417)	Manner of differentiating the values of different signs on the diagrams Available since version 2.0.
◆	ReactionsInLocalSystem (↗ see page 1417)	Available since version 2.0.
◆	Values (↗ see page 1417)	

#### Public Methods

	Name	Description
♫	GetColor (↗ see page 1418)	Function returns color which is used while drawing a diagram of a specified result type. Available since version 2.0.
♫	GetScale (↗ see page 1418)	Function returns scale value for a specified result diagram for 1 cm. Available since version 2.0.
♫	IsOn (↗ see page 1418)	Function returns value different from zero (True) if a specified result type is switched on (i.e. it is displayed in the view). Available since version 2.0.
♫	Set (↗ see page 1419)	Function switches on/off diagram display for a specified result type. Available since version 2.0.
♫	SetColor (↗ see page 1419)	Function sets color which is to be used while drawing a diagram of a specified result type. Available since version 2.0.
♫	SetScale (↗ see page 1419)	Function sets a diagram scale for 1 cm for a specified result type. If the same scale is valid for several associated results it is set for all of them. If scale definition for a specified result type makes no sense, then scale value is ignored. Available since version 2.0.

### I.2.14.2 IRobotViewDiagramParams Fields

The fields of the IRobotViewDiagramParams class are listed here.

#### Public Fields

	Name	Description
◆	Descriptions (↗ see page 1416)	The manner of locating the diagram description Available since version 2.0.

◆	Filling ( <a href="#">see page 1416</a> )	Type of diagram filling Available since version 2.0.
◆	PositiveNegative ( <a href="#">see page 1417</a> )	Manner of differentiating the values of different signs on the diagrams Available since version 2.0.
◆	ReactionsInLocalSystem ( <a href="#">see page 1417</a> )	Available since version 2.0.
◆	Values ( <a href="#">see page 1417</a> )	

#### I.2.14.2.1 Descriptions

##### C++

```
HRESULT get_Descriptions(IRobotViewDiagramDescriptionType* );
HRESULT put_Descriptions(IRobotViewDiagramDescriptionType);
```

##### C#

```
public IRobotViewDiagramDescriptionType Descriptions { get; set; }
```

##### Visual Basic

```
Public Descriptions As IRobotViewDiagramDescriptionType
```

##### Description

The manner of locating the diagram description Available since version 2.0.

#### I.2.14.2.2 Filling

##### C++

```
HRESULT get_Filling(IRobotViewDiagramFillingType* );
HRESULT put_Filling(IRobotViewDiagramFillingType);
```

##### C#

```
public IRobotViewDiagramFillingType Filling { get; set; }
```

##### Visual Basic

```
Public Filling As IRobotViewDiagramFillingType
```

##### Description

Type of diagram filling Available since version 2.0.

#### I.2.14.2.3 PositiveNegative

##### C++

```
HRESULT get_PositiveNegative(IRobotViewDiagramSignDifferType* );
HRESULT put_PositiveNegative(IRobotViewDiagramSignDifferType);
```

##### C#

```
public IRobotViewDiagramSignDifferType PositiveNegative { get; set; }
```

##### Visual Basic

```
Public PositiveNegative As IRobotViewDiagramSignDifferType
```

##### Description

Manner of differentiating the values of different signs on the diagrams Available since version 2.0.

#### I.2.14.2.4 ReactionsInLocalSystem

##### C++

```
HRESULT get_ReactionsInLocalSystem(VARIANT_BOOL* );
HRESULT put_ReactionsInLocalSystem(VARIANT_BOOL);
```

**C#**

```
public bool ReactionsInLocalSystem { get; set; }
```

**Visual Basic**

```
Public ReactionsInLocalSystem As Boolean
```

**Description**

Available since version 2.0.

**I.2.14.2.5 Values****C++**

```
HRESULT get_Values(IRobotViewDiagramValueType* );
HRESULT put_Values(IRobotViewDiagramValueType);
```

**C#**

```
public IRobotViewDiagramValueType Values { get; set; }
```

**Visual Basic**

```
Public Values As IRobotViewDiagramValueType
```

**Version**

Available since version 8.1.

**I.2.14.3 IRobotViewDiagramParams Methods**

The methods of the IRobotViewDiagramParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetColor (see page 1418)	Function returns color which is used while drawing a diagram of a specified result type. Available since version 2.0.
💡	GetScale (see page 1418)	Function returns scale value for a specified result diagram for 1 cm. Available since version 2.0.
💡	IsOn (see page 1418)	Function returns value different from zero (True) if a specified result type is switched on (i.e. it is displayed in the view). Available since version 2.0.
💡	Set (see page 1419)	Function switches on/off diagram display for a specified result type. Available since version 2.0.
💡	SetColor (see page 1419)	Function sets color which is to be used while drawing a diagram of a specified result type. Available since version 2.0.
💡	SetScale (see page 1419)	Function sets a diagram scale for 1 cm for a specified result type. If the same scale is valid for several associated results it is set for all of them. If scale definition for a specified result type makes no sense, then scale value is ignored. Available since version 2.0.

**I.2.14.3.1 GetColor****C++**

```
HRESULT GetColor(IRobotViewDiagramResultType _res_type, long* ret);
```

**C#**

```
public long GetColor(IRobotViewDiagramResultType _res_type);
```

**Visual Basic**

```
Public Function GetColor(_res_type As IRobotViewDiagramResultType) As long
```

## Description

Function returns color which is used while drawing a diagram of a specified result type. Available since version 2.0.

### I.2.14.3.2 GetScale

#### C++

```
HRESULT GetScale(IRobotViewDiagramResultType _res_type, double* ret);
```

#### C#

```
public double GetScale(IRobotViewDiagramResultType _res_type);
```

#### Visual Basic

```
Public Function GetScale(_res_type As IRobotViewDiagramResultType) As Double
```

## Description

Function returns scale value for a specified result diagram for 1 cm. Available since version 2.0.

### I.2.14.3.3 IsOn

#### C++

```
HRESULT IsOn(IRobotViewDiagramResultType _res_type, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsOn(IRobotViewDiagramResultType _res_type);
```

#### Visual Basic

```
Public Function IsOn(_res_type As IRobotViewDiagramResultType) As Boolean
```

## Description

Function returns value different from zero (True) if a specified result type is switched on (i.e. it is displayed in the view). Available since version 2.0.

### I.2.14.3.4 Set

#### C++

```
HRESULT Set(IRobotViewDiagramResultType _res_type, VARIANT_BOOL _on_off);
```

#### C#

```
public void Set(IRobotViewDiagramResultType _res_type, bool _on_off);
```

#### Visual Basic

```
Public Sub Set(_res_type As IRobotViewDiagramResultType, _on_off As Boolean)
```

## Description

Function switches on/off diagram display for a specified result type. Available since version 2.0.

### I.2.14.3.5 SetColor

#### C++

```
HRESULT SetColor(IRobotViewDiagramResultType _res_type, long _color);
```

#### C#

```
public void SetColor(IRobotViewDiagramResultType _res_type, long _color);
```

**Visual Basic**

```
Public Sub SetColor(_res_type As IRobotViewDiagramResultType, _color As long)
```

**Description**

Function sets color which is to be used while drawing a diagram of a specified result type. Available since version 2.0.

**I.2.14.3.6 SetScale****C++**

```
HRESULT SetScale(IRobotViewDiagramResultType _res_type, double _scale_for_1_cm);
```

**C#**

```
public void SetScale(IRobotViewDiagramResultType _res_type, double _scale_for_1_cm);
```

**Visual Basic**

```
Public Sub SetScale(_res_type As IRobotViewDiagramResultType, _scale_for_1_cm As double)
```

**Description**

Function sets a diagram scale for 1 cm for a specified result type. If the same scale is valid for several associated results it is set for all of them. If scale definition for a specified result type makes no sense, then scale value is ignored. Available since version 2.0.

**I.2.15 IRobotViewDiagramResultType****C++**

```
enum IRobotViewDiagramResultType;
```

**C#**

```
public enum IRobotViewDiagramResultType;
```

**Visual Basic**

```
Public Enum IRobotViewDiagramResultType
```

**Members**

Members	Description
I_VDRT_NTM_FX = 1	Available since version 2.0.
I_VDRT_NTM_FY = 2	Available since version 2.0.
I_VDRT_NTM_FZ = 3	Available since version 2.0.
I_VDRT_NTM_MX = 4	Available since version 2.0.
I_VDRT_NTM_MY = 5	Available since version 2.0.
I_VDRT_NTM_MZ = 6	Available since version 2.0.
I_VDRT_DEFORMATION_DEFORMATION = 7	Available since version 2.0.
I_VDRT_DEFORMATION_EXACT = 8	Available since version 2.0.
I_VDRT_STRESS_S_MAX = 9	Available since version 2.0.
I_VDRT_STRESS_S_MIN = 10	Available since version 2.0.
I_VDRT_STRESS_BENDING_S_MAX_MY = 11	Available since version 2.0.
I_VDRT_STRESS_BENDING_S_MAX_MZ = 12	Available since version 2.0.
I_VDRT_STRESS_BENDING_S_MIN_MY = 13	Available since version 2.0.
I_VDRT_STRESS_BENDING_S_MIN_MZ = 14	Available since version 2.0.
I_VDRT_STRESS_AXIAL_FX_AX = 15	Available since version 2.0.
I_VDRT_STRESS_SHEAR_TY = 16	Available since version 2.0.
I_VDRT_STRESS_SHEAR_TZ = 17	Available since version 2.0.

I_VDRT_STRESS_TORSION = 18	Available since version 2.0.
I_VDRTREACTION_FORCES = 19	Available since version 2.0.
I_VDRTREACTION_MOMENTS = 20	Available since version 2.0.
I_VDRTREACTION_RESIDUAL_FORCES = 21	Available since version 2.0.
I_VDRTREACTION_RESIDUAL_MOMENTS = 22	Available since version 2.0.
I_VDRTREINFORCE_TOP = 23	Available since version 2.0.
I_VDRTREINFORCE_TOP_REAL = 24	Available since version 2.0.
I_VDRTREINFORCE_BOTTOM = 25	Available since version 2.0.
I_VDRTREINFORCE_BOTTOM_REAL = 26	Available since version 2.0.
I_VDRTREINFORCE_RATIO = 27	Available since version 2.0.
I_VDRTREINFORCE_RATIO_REAL = 28	Available since version 2.0.
I_VDRTREINFORCE_STIRRUP_SPACING = 29	Available since version 2.0.
I_VDRTREINFORCE_STIRRUP_SPACING_REAL = 30	Available since version 2.0.
I_VDRTREINFORCE_NUMBER_OF_UPPER_BARS = 31	Available since version 2.0.
I_VDRTREINFORCE_NUMBER_OF_LOWER_BARS = 32	Available since version 2.0.
I_VDRTNTM_KY = 33	Coefficient of elastic ground reaction in Y direction. Available since version 3.
I_VDRTNTM_KZ = 34	Coefficient of elastic ground reaction in Z direction. Available since version 3.
I_VDRTREACTION_FX = 35	.
I_VDRTREACTION_FY = 36	.
I_VDRTREACTION_FZ = 37	.
I_VDRTREACTION_MX = 38	.
I_VDRTREACTION_MY = 39	.
I_VDRTREACTION_MZ = 40	.
I_VDRTREACTION_PSEUDOSTATIC_FORCES = 42	.
I_VDRTREACTION_LINEAR_SUPPORTS = 43	.
I_VDRTREACTION_DESC = 44	.
I_VDRTREACTION_DESC_AVERAGE = 45	.
I_VDRTREACTION_DESC_INTEGRAL = 46	.
I_VDRTREACTION_PSEUDOSTATIC_MOMENTS = 47	.

## Description

Set of result types presented by means of diagrams on bars.

### I.2.16 IRobotViewDiagramDescriptionType

#### C++

```
enum IRobotViewDiagramDescriptionType;
```

**C#**

```
public enum IRobotViewDiagramDescriptionType;
```

**Visual Basic**

```
Public Enum IRobotViewDiagramDescriptionType
```

**Members**

Members	Description
I_VDDT_NONE = 0	Available since version 2.0.
I_VDDT_LABELS = 1	Available since version 8.1.
I_VDDT_TEXT = 2	Available since version 8.1.

**Description**

Available methods of locating descriptions for diagrams.

**I.2.17 IRobotViewDiagramFillingType****C++**

```
enum IRobotViewDiagramFillingType;
```

**C#**

```
public enum IRobotViewDiagramFillingType;
```

**Visual Basic**

```
Public Enum IRobotViewDiagramFillingType
```

**Members**

Members	Description
I_VDFT_FENCE = 0	Available since version 2.0.
I_VDFT_FILLED = 1	Available since version 2.0.

**Description**

Available methods of diagram filling.

**I.2.18 IRobotViewDiagramSignDifferType****C++**

```
enum IRobotViewDiagramSignDifferType;
```

**C#**

```
public enum IRobotViewDiagramSignDifferType;
```

**Visual Basic**

```
Public Enum IRobotViewDiagramSignDifferType
```

**Members**

Members	Description
I_VDSDT_UNDIFFERENTIATED = 0	Available since version 2.0.
I_VDSDT_DIFFERENTIATED = 1	Available since version 2.0.

**Description**

Available methods of distinguishing positive and negative values on diagrams.

## I.2.19 IRobotViewDiagrams

### Class Hierarchy

C++

```
interface IRobotViewDiagrams : IRobotView3;
```

C#

```
public interface IRobotViewDiagrams : IRobotView3;
```

### Visual Basic

```
Public Interface IRobotViewDiagrams
```

### Description

A viewer dedicated to result presentation.

## I.2.20 IRobotViewBarMaps

### Class Hierarchy

C++

```
interface IRobotViewBarMaps : IRobotView3;
```

C#

```
public interface IRobotViewBarMaps : IRobotView3;
```

### Visual Basic

```
Public Interface IRobotViewBarMaps
```

### Description

View presenting maps on bars with a scale. .

## I.2.21 IRobotViewFeMaps

### Class Hierarchy

C++

```
interface IRobotViewFeMaps : IRobotView3;
```

C#

```
public interface IRobotViewFeMaps : IRobotView3;
```

### Visual Basic

```
Public Interface IRobotViewFeMaps
```

### Description

View presenting maps on finite elements with a scale. .

## I.2.22 IRobotViewBarMapResultType

C++

```
enum IRobotViewBarMapResultType;
```

C#

```
public enum IRobotViewBarMapResultType;
```

## Visual Basic

```
Public Enum IRobotViewBarMapResultType
```

## Members

Members	Description
I_VBMRT_NTM_FX = 165	Available since version 2.0.
I_VBMRT_NTM_FY = 166	Available since version 2.0.
I_VBMRT_NTM_FZ = 167	Available since version 2.0.
I_VBMRT_NTM_MX = 168	Available since version 2.0.
I_VBMRT_NTM_MY = 169	Available since version 2.0.
I_VBMRT_NTM_MZ = 170	Available since version 2.0.
I_VBMRT_STRESS_S_MAX = 171	Available since version 2.0.
I_VBMRT_STRESS_S_MIN = 172	Available since version 2.0.
I_VBMRT_STRESS_S_MAX_MY = 174	Available since version 2.0.
I_VBMRT_STRESS_S_MIN_MY = 176	Available since version 2.0.
I_VBMRT_STRESS_S_MAX_MZ = 175	Available since version 2.0.
I_VBMRT_STRESS_S_MIN_MZ = 177	Available since version 2.0.
I_VBMRT_STRESS_FX_AX = 173	Available since version 2.0.
I_VBMRT_STRESS_SHEAR_TY = 179	Available since version 2.0.
I_VBMRT_STRESS_SHEAR_TZ = 180	Available since version 2.0.
I_VBMRT_STRESS_TORSION_T = 182	Available since version 2.0.
I_VBMRT DESIGN_RATIO = 796	Available since version 2.0.
I_VBMRT DESIGN_MEMBER_LENGTH = 271	Available since version 2.0.
I_VBMRT DESIGN_SLEND_LAY = 831	Available since version 2.0.
I_VBMRT DESIGN_SLEND_LA = 832	Available since version 2.0.
I_VBMRT NOTHING = -1	Available since version 2.0.

## Description

Set of result types presented by means of maps on bars.

## I.2.23 IRobotViewBarMapParams

### Class Hierarchy

#### C++

```
interface IRobotViewBarMapParams : IDispatch;
```

#### C#

```
public interface IRobotViewBarMapParams;
```

## Visual Basic

```
Public Interface IRobotViewBarMapParams
```

## Description

Parameters describing the manner of displaying maps on bars.

### I.2.23.1 IRobotViewBarMapParams Members

The following tables list the members exposed by IRobotViewBarMapParams.

## Public Fields

	Name	Description
◆	CurrentResult (see page 1425)	Result type for which a map on bars is currently displayed.
◆	Descriptions (see page 1425)	Manner of displaying map description Available since version 2.0.
◆	MapThicknessCoeff (see page 1425)	Thickness coefficient of a map drawing line in relation to the thickness of a bar drawing line Available since version 2.0.
◆	StructureDeformation (see page 1425)	Flag switching on/off structure deformation drawing Available since version 2.0.

### I.2.23.2 IRobotViewBarMapParams Fields

The fields of the IRobotViewBarMapParams class are listed here.

## Public Fields

	Name	Description
◆	CurrentResult (see page 1425)	Result type for which a map on bars is currently displayed.
◆	Descriptions (see page 1425)	Manner of displaying map description Available since version 2.0.
◆	MapThicknessCoeff (see page 1425)	Thickness coefficient of a map drawing line in relation to the thickness of a bar drawing line Available since version 2.0.
◆	StructureDeformation (see page 1425)	Flag switching on/off structure deformation drawing Available since version 2.0.

#### I.2.23.2.1 CurrentResult

##### C++

```
HRESULT get_CurrentResult(IRobotViewBarMapResultType* );
HRESULT put_CurrentResult(IRobotViewBarMapResultType);
```

##### C#

```
public IRobotViewBarMapResultType CurrentResult { get; set; }
```

##### Visual Basic

```
Public CurrentResult As IRobotViewBarMapResultType
```

##### Description

Result type for which a map on bars is currently displayed.

#### I.2.23.2.2 Descriptions

##### C++

```
HRESULT get_Descriptions(IRobotViewDiagramDescriptionType* );
HRESULT put_Descriptions(IRobotViewDiagramDescriptionType);
```

##### C#

```
public IRobotViewDiagramDescriptionType Descriptions { get; set; }
```

##### Visual Basic

```
Public Descriptions As IRobotViewDiagramDescriptionType
```

##### Description

Manner of displaying map description Available since version 2.0.

#### I.2.23.2.3 MapThicknessCoeff

##### C++

```
HRESULT get_MapThicknessCoeff(double* );
```

```
HRESULT put_MapThicknessCoeff(double);
```

**C#**

```
public double MapThicknessCoeff { get; set; }
```

**Visual Basic**

```
Public MapThicknessCoeff As Double
```

**Description**

Thickness coefficient of a map drawing line in relation to the thickness of a bar drawing line Available since version 2.0.

**I.2.23.2.4 StructureDeformation****C++**

```
HRESULT get_StructureDeformation(VARIANT_BOOL* );
HRESULT put_StructureDeformation(VARIANT_BOOL);
```

**C#**

```
public bool StructureDeformation { get; set; }
```

**Visual Basic**

```
Public StructureDeformation As Boolean
```

**Description**

Flag switching on/off structure deformation drawing Available since version 2.0.

**I.2.24 IRobotViewDetailedAnalysisResultType****C++**

```
enum IRobotViewDetailedAnalysisResultType;
```

**C#**

```
public enum IRobotViewDetailedAnalysisResultType;
```

**Visual Basic**

```
Public Enum IRobotViewDetailedAnalysisResultType
```

**Members**

Members	Description
I_VDART_NTM_FX = 1	Available since version 2.0.
I_VDART_NTM_FY = 2	Available since version 2.0.
I_VDART_NTM_FZ = 3	Available since version 2.0.
I_VDART_NTM_MX = 4	Available since version 2.0.
I_VDART_NTM_MY = 5	Available since version 2.0.
I_VDART_NTM_MZ = 6	Available since version 2.0.
I_VDART_NTM_UX = 7	Available since version 2.0.
I_VDART_NTM_UY = 8	Available since version 2.0.
I_VDART_NTM_UZ = 9	Available since version 2.0.
I_VDART_STRESS_S_MAX = 10	Available since version 2.0.
I_VDART_STRESS_S_MIN = 11	Available since version 2.0.
I_VDART_STRESS_BENDING_S_MAX_MY = 12	Available since version 2.0.
I_VDART_STRESS_BENDING_S_MIN_MY = 13	Available since version 2.0.
I_VDART_STRESS_BENDING_S_MAX_MZ = 14	Available since version 2.0.
I_VDART_STRESS_BENDING_S_MIN_MZ = 15	Available since version 2.0.

I_VDART_STRESS_AXIAL_FX_AX = 16	Available since version 2.0.
I_VDART_STRESS_SHEAR_TY = 17	Available since version 2.0.
I_VDART_STRESS_SHEAR_TZ = 18	Available since version 2.0.
I_VDART_STRESS_TORSION_T = 19	Available since version 2.0.
I_VDART_REINFORCE_TOP = 20	Available since version 2.0.
I_VDART_REINFORCE_TOP_REAL = 21	Available since version 2.0.
I_VDART_REINFORCE_BOTTOM = 22	Available since version 2.0.
I_VDART_REINFORCE_BOTTOM_REAL = 23	Available since version 2.0.
I_VDART_REINFORCE_RATIO = 24	Available since version 2.0.
I_VDART_REINFORCE_RATIO_REAL = 25	Available since version 2.0.
I_VDART_REINFORCE_STIRRUP_SPACING = 26	Available since version 2.0.
I_VDART_REINFORCE_STIRRUP_SPACING_REAL = 27	Available since version 2.0.
I_VDART_REINFORCE_NUMBER_OF_UPPER_BARS = 28	Available since version 2.0.
I_VDART_REINFORCE_NUMBER_OF_LOWER_BARS = 29	Available since version 2.0.

**Description**

Result types presented by detailed analysis view.

**I.2.25 IRobotViewDetailedAnalysisParams****Class Hierarchy****C++**

```
interface IRobotViewDetailedAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotViewDetailedAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotViewDetailedAnalysisParams
```

**Description**

Parameters of detailed analysis view.

**I.2.25.1 IRobotViewDetailedAnalysisParams Members**

The following tables list the members exposed by IRobotViewDetailedAnalysisParams.

**Public Fields**

	Name	Description
❖	Descriptions (☞ see page 1428)	Manner of displaying diagram description Available since version 2.0.
❖	Filling (☞ see page 1428)	Type of diagram filling Available since version 2.0.
❖	PositiveNegative (☞ see page 1428)	Manner of presenting values of different signs on the diagram Available since version 2.0.
❖	ReinforceShowTheoreticAndRealVals (☞ see page 1428)	Available since version 2.0.

**Public Methods**

	Name	Description
❖	GetColor (☞ see page 1429)	Function returns color assigned to a specified result type. Available since version 2.0.
❖	IsOn (☞ see page 1429)	Function returns value different from zero (True) if a selected result type is displayed by the detailed analysis view. Available since version 2.0.

	Set ( <a href="#">see page 1429</a> )	Function switches on/off display of a defined result type. Available since version 2.0.
	SetColor ( <a href="#">see page 1430</a> )	Function sets color which is to be used while drawing a diagram for a specified result type. Available since version 2.0.

### I.2.25.2 IRobotViewDetailedAnalysisParams Fields

The fields of the IRobotViewDetailedAnalysisParams class are listed here.

#### Public Fields

	Name	Description
	Descriptions ( <a href="#">see page 1428</a> )	Manner of displaying diagram description Available since version 2.0.
	Filling ( <a href="#">see page 1428</a> )	Type of diagram filling Available since version 2.0.
	PositiveNegative ( <a href="#">see page 1428</a> )	Manner of presenting values of different signs on the diagram Available since version 2.0.
	ReinforceShowTheoreticAndRealVals ( <a href="#">see page 1428</a> )	Available since version 2.0.

#### I.2.25.2.1 Descriptions

##### C++

```
HRESULT get_Descriptions(IRobotViewDiagramDescriptionType* );
HRESULT put_Descriptions(IRobotViewDiagramDescriptionType);
```

##### C#

```
public IRobotViewDiagramDescriptionType Descriptions { get; set; }
```

##### Visual Basic

```
Public Descriptions As IRobotViewDiagramDescriptionType
```

##### Description

Manner of displaying diagram description Available since version 2.0.

#### I.2.25.2.2 Filling

##### C++

```
HRESULT get_Filling(IRobotViewDiagramFillingType* );
HRESULT put_Filling(IRobotViewDiagramFillingType);
```

##### C#

```
public IRobotViewDiagramFillingType Filling { get; set; }
```

##### Visual Basic

```
Public Filling As IRobotViewDiagramFillingType
```

##### Description

Type of diagram filling Available since version 2.0.

#### I.2.25.2.3 PositiveNegative

##### C++

```
HRESULT get_PositiveNegative(IRobotViewDiagramSignDifferType* );
HRESULT put_PositiveNegative(IRobotViewDiagramSignDifferType);
```

##### C#

```
public IRobotViewDiagramSignDifferType PositiveNegative { get; set; }
```

**Visual Basic**

```
Public PositiveNegative As IRobotViewDiagramSignDifferType
```

**Description**

Manner of presenting values of different signs on the diagram Available since version 2.0.

**I.2.25.2.4 ReinforceShowTheoreticAndRealVals****C++**

```
HRESULT get_ReinforceShowTheoreticAndRealVals(VARIANT_BOOL* );
HRESULT put_ReinforceShowTheoreticAndRealVals(VARIANT_BOOL);
```

**C#**

```
public bool ReinforceShowTheoreticAndRealVals { get; set; }
```

**Visual Basic**

```
Public ReinforceShowTheoreticAndRealVals As Boolean
```

**Description**

Available since version 2.0.

**I.2.25.3 IRobotViewDetailedAnalysisParams Methods**

The methods of the IRobotViewDetailedAnalysisParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	GetColor ( <a href="#">see page 1429</a> )	Function returns color assigned to a specified result type. Available since version 2.0.
≡	IsOn ( <a href="#">see page 1429</a> )	Function returns value different from zero (True) if a selected result type is displayed by the detailed analysis view. Available since version 2.0.
≡	Set ( <a href="#">see page 1429</a> )	Function switches on/off display of a defined result type. Available since version 2.0.
≡	SetColor ( <a href="#">see page 1430</a> )	Function sets color which is to be used while drawing a diagram for a specified result type. Available since version 2.0.

**I.2.25.3.1 GetColor****C++**

```
HRESULT GetColor(IRobotViewDetailedAnalysisResultType _res_type, long* ret);
```

**C#**

```
public long GetColor(IRobotViewDetailedAnalysisResultType _res_type);
```

**Visual Basic**

```
Public Function GetColor(_res_type As IRobotViewDetailedAnalysisResultType) As long
```

**Description**

Function returns color assigned to a specified result type. Available since version 2.0.

**I.2.25.3.2 IsOn****C++**

```
HRESULT IsOn(IRobotViewDetailedAnalysisResultType _res_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsOn(IRobotViewDetailedAnalysisResultType _res_type);
```

**Visual Basic**

```
Public Function IsOn(_res_type As IRobotViewDetailedAnalysisResultType) As Boolean
```

**Description**

Function returns value different from zero (True) if a selected result type is displayed by the detailed analysis view. Available since version 2.0.

**I.2.25.3.3 Set****C++**

```
HRESULT Set(IRobotViewDetailedAnalysisResultType _res_type, VARIANT_BOOL _on_off);
```

**C#**

```
public void Set(IRobotViewDetailedAnalysisResultType _res_type, bool _on_off);
```

**Visual Basic**

```
Public Sub Set(_res_type As IRobotViewDetailedAnalysisResultType, _on_off As Boolean)
```

**Description**

Function switches on/off display of a defined result type. Available since version 2.0.

**I.2.25.3.4 SetColor****C++**

```
HRESULT SetColor(IRobotViewDetailedAnalysisResultType _res_type, long _color);
```

**C#**

```
public void SetColor(IRobotViewDetailedAnalysisResultType _res_type, long _color);
```

**Visual Basic**

```
Public Sub SetColor(_res_type As IRobotViewDetailedAnalysisResultType, _color As long)
```

**Description**

Function sets color which is to be used while drawing a diagram for a specified result type. Available since version 2.0.

**I.2.26 IRobotViewFeMapResultType****C++**

```
enum IRobotViewFeMapResultType;
```

**C#**

```
public enum IRobotViewFeMapResultType;
```

**Visual Basic**

```
Public Enum IRobotViewFeMapResultType
```

**Members**

Members	Description
I_VFMRT_DETAILED_STRESS_XX = 483	Available since version 2.0.
I_VFMRT_DETAILED_STRESS_YY = 484	Available since version 2.0.

I_VFMRT_DETAILED_STRESS_XY = 485	Available since version 2.0.
I_VFMRT_DETAILED_STRESS_Z = 486	Available since version 2.0.
I_VFMRT_DETAILED_MEMBRANE_FORCE_XX = 492	Available since version 2.0.
I_VFMRT_DETAILED_MEMBRANE_FORCE_YY = 493	Available since version 2.0.
I_VFMRT_DETAILED_MEMBRANE_FORCE_XY = 494	Available since version 2.0.
I_VFMRT_DETAILED_MOMENT_XX = 501	Available since version 2.0.
I_VFMRT_DETAILED_MOMENT_YY = 502	Available since version 2.0.
I_VFMRT_DETAILED_MOMENT_XY = 503	Available since version 2.0.
I_VFMRT_DETAILED_SHEAR_STRESS_XX = 510	Available since version 2.0.
I_VFMRT_DETAILED_SHEAR_STRESS_YY = 511	Available since version 2.0.
I_VFMRT_DETAILED_SHEAR_FORCE_XX = 519	Available since version 2.0.
I_VFMRT_DETAILED_SHEAR_FORCE_YY = 520	Available since version 2.0.
I_VFMRT_DETAILED_DISPLACEMENT_XX = 537	Available since version 2.0.
I_VFMRT_DETAILED_DISPLACEMENT_YY = 538	Available since version 2.0.
I_VFMRT_DETAILED_ROTATION_XX = 546	Available since version 2.0.
I_VFMRT_DETAILED_ROTATION_YY = 547	Available since version 2.0.
I_VFMRT_DETAILED_ROTATION_Z = 549	Available since version 2.0.
I_VFMRT_DETAILED_SOILREACTION_Z = 558	Available since version 2.0.
I_VFMRT_PRINCIPAL_STRESS_1 = 487	Available since version 2.0.
I_VFMRT_PRINCIPAL_STRESS_2 = 488	Available since version 2.0.
I_VFMRT_PRINCIPAL_STRESS_1_2 = 490	Available since version 2.0.
I_VFMRT_PRINCIPAL_STRESS_ANGLE = 489	Available since version 2.0.
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_1 = 496	Available since version 2.0.
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_2 = 497	Available since version 2.0.
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_1_2 = 499	Available since version 2.0.
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_ANGLE = 498	Available since version 2.0.
I_VFMRT_PRINCIPAL_MOMENT_1 = 505	Available since version 2.0.
I_VFMRT_PRINCIPAL_MOMENT_2 = 506	Available since version 2.0.
I_VFMRT_PRINCIPAL_MOMENT_1_2 = 508	Available since version 2.0.
I_VFMRT_PRINCIPAL_MOMENT_ANGLE = 507	Available since version 2.0.
I_VFMRT_PRINCIPAL_SHEAR_STRESS_1_2 = 517	Available since version 2.0.
I_VFMRT_PRINCIPAL_SHEAR_FORCE_1_2 = 526	Available since version 2.0.
I_VFMRT_GLOBAL_DISPLACEMENT_X = 1302	Available since version 2.0.
I_VFMRT_GLOBAL_DISPLACEMENT_Y = 1303	Available since version 2.0.
I_VFMRT_GLOBAL_DISPLACEMENT_Z = 1304	Available since version 2.0.
I_VFMRT_TOTAL_DISPLACEMENTS = 1305	Available since version 2.0.
I_VFMRT_COMPLEX_STRESSES = 491	Available since version 2.0.
I_VFMRT_COMPLEX_MEMBRANE_FORCES = 500	Available since version 2.0.
I_VFMRT_COMPLEX_MOMENTS = 509	Available since version 2.0.
I_VFMRT_COMPLEX_REINFORCE_TOP_MXX = 889	Available since version 2.0.
I_VFMRT_COMPLEX_REINFORCE_TOP_MYY = 891	Available since version 2.0.
I_VFMRT_COMPLEX_REINFORCE_BOTTOM_MXX = 890	Available since version 2.0.
I_VFMRT_COMPLEX_REINFORCE_BOTTOM_MYY = 892	Available since version 2.0.
I_VFMRT_DETAILED_DISPLACEMENT_Z = 531	.
	Available since version 15.2.

**Description**

Result types presented by means of maps on finite elements. .

## I.2.27 IRobotViewFeMapParams

### Class Hierarchy

#### C++

```
interface IRobotViewFeMapParams : IDispatch;
```

#### C#

```
public interface IRobotViewFeMapParams;
```

### Visual Basic

```
Public Interface IRobotViewFeMapParams
```

### Description

Parameters of presenting maps on finite elements. .

### I.2.27.1 IRobotViewFeMapParams Members

The following tables list the members exposed by IRobotViewFeMapParams.

#### Public Fields

	Name	Description
◆	CrossPresentation ( <a href="#">see page 1433</a> )	Available since version 2.0.
◆	CurrentResult ( <a href="#">see page 1433</a> )	Currently displayed result type Available since version 2.0.
◆	DeformationActive ( <a href="#">see page 1433</a> )	Available since version 2.0.
◆	DeformationConstScale ( <a href="#">see page 1434</a> )	Available since version 2.0.
◆	Direction ( <a href="#">see page 1434</a> )	Available since version 2.0.
◆	DirectionData ( <a href="#">see page 1434</a> )	Additional data describing direction; depending on the selected method of determining direction, it contains coordinates of the point or vector defining the direction (for the polar system defined in a node, the DirectionNode ( <a href="#">see page 1434</a> ) variable denoting a node number should be applied) Available since version 2.0.
◆	DirectionNode ( <a href="#">see page 1434</a> )	Number of the node determining direction in case when direction is defined in the polar system by indicating a structure node Available since version 2.0.
◆	Isolines ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	Layer ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	LayerArbitraryVal ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	MapScale ( <a href="#">see page 1436</a> )	Available since version 2.0.
◆	Smoothing ( <a href="#">see page 1436</a> )	Available since version 2.0.
◆	WithDescription ( <a href="#">see page 1436</a> )	Available since version 2.0.

#### Public Methods

	Name	Description
◆	SetDirectionData ( <a href="#">see page 1436</a> )	

### I.2.27.2 IRobotViewFeMapParams Fields

The fields of the IRobotViewFeMapParams class are listed here.

## Public Fields

	Name	Description
◆	CrossPresentation ( <a href="#">see page 1433</a> )	Available since version 2.0.
◆	CurrentResult ( <a href="#">see page 1433</a> )	Currently displayed result type Available since version 2.0.
◆	DeformationActive ( <a href="#">see page 1433</a> )	Available since version 2.0.
◆	DeformationConstScale ( <a href="#">see page 1434</a> )	Available since version 2.0.
◆	Direction ( <a href="#">see page 1434</a> )	Available since version 2.0.
◆	DirectionData ( <a href="#">see page 1434</a> )	Additional data describing direction; depending on the selected method of determining direction, it contains coordinates of the point or vector defining the direction (for the polar system defined in a node, the DirectionNode ( <a href="#">see page 1434</a> ) variable denoting a node number should be applied) Available since version 2.0.
◆	DirectionNode ( <a href="#">see page 1434</a> )	Number of the node determining direction in case when direction is defined in the polar system by indicating a structure node Available since version 2.0.
◆	Isolines ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	Layer ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	LayerArbitraryVal ( <a href="#">see page 1435</a> )	Available since version 2.0.
◆	MapScale ( <a href="#">see page 1436</a> )	Available since version 2.0.
◆	Smoothing ( <a href="#">see page 1436</a> )	Available since version 2.0.
◆	WithDescription ( <a href="#">see page 1436</a> )	Available since version 2.0.

### I.2.27.2.1 CrossPresentation

#### C++

```
HRESULT get_CrossPresentation(IRobotViewFeMapCrossPresentationType* );
HRESULT put_CrossPresentation(IRobotViewFeMapCrossPresentationType);
```

#### C#

```
public IRobotViewFeMapCrossPresentationType CrossPresentation { get; set; }
```

#### Visual Basic

```
Public CrossPresentation As IRobotViewFeMapCrossPresentationType
```

#### Description

Available since version 2.0.

### I.2.27.2.2 CurrentResult

#### C++

```
HRESULT get_CurrentResult(IRobotViewFeMapResultType* );
HRESULT put_CurrentResult(IRobotViewFeMapResultType);
```

#### C#

```
public IRobotViewFeMapResultType CurrentResult { get; set; }
```

#### Visual Basic

```
Public CurrentResult As IRobotViewFeMapResultType
```

#### Description

Currently displayed result type Available since version 2.0.

### I.2.27.2.3 DeformationActive

#### C++

```
HRESULT get_DeformationActive(VARIANT_BOOL* );
HRESULT put_DeformationActive(VARIANT_BOOL);
```

#### C#

```
public bool DeformationActive { get; set; }
```

#### Visual Basic

```
Public DeformationActive As Boolean
```

#### Description

Available since version 2.0.

### I.2.27.2.4 DeformationConstScale

#### C++

```
HRESULT get_DeformationConstScale(VARIANT_BOOL* );
HRESULT put_DeformationConstScale(VARIANT_BOOL);
```

#### C#

```
public bool DeformationConstScale { get; set; }
```

#### Visual Basic

```
Public DeformationConstScale As Boolean
```

#### Description

Available since version 2.0.

### I.2.27.2.5 Direction

#### C++

```
HRESULT get_Direction(IRobotViewFeMapLocalSystemType* );
HRESULT put_Direction(IRobotViewFeMapLocalSystemType);
```

#### C#

```
public IRobotViewFeMapLocalSystemType Direction { get; set; }
```

#### Visual Basic

```
Public Direction As IRobotViewFeMapLocalSystemType
```

#### Description

Available since version 2.0.

### I.2.27.2.6 DirectionData

#### C++

```
HRESULT get_DirectionData(IRobotGeoPoint3D** );
```

#### C#

```
public IRobotGeoPoint3D DirectionData { get; }
```

#### Visual Basic

```
Public ReadOnly DirectionData As IRobotGeoPoint3D
```

## Description

Additional data describing direction; depending on the selected method of determining direction, it contains coordinates of the point or vector defining the direction (for the polar system defined in a node, the DirectionNode (see page 1434) variable denoting a node number should be applied) Available since version 2.0.

### I.2.27.2.7 DirectionNode

#### C++

```
HRESULT get_DirectionNode(long* );
HRESULT put_DirectionNode(long);
```

#### C#

```
public long DirectionNode { get; set; }
```

#### Visual Basic

```
Public DirectionNode As long
```

#### Description

Number of the node determining direction in case when direction is defined in the polar system by indicating a structure node Available since version 2.0.

### I.2.27.2.8 Isolines

#### C++

```
HRESULT get_Isolines(VARIANT_BOOL* );
HRESULT put_Isolines(VARIANT_BOOL);
```

#### C#

```
public bool Isolines { get; set; }
```

#### Visual Basic

```
Public Isolines As Boolean
```

#### Description

Available since version 2.0.

### I.2.27.2.9 Layer

#### C++

```
HRESULT get_Layer(IRobotViewFeMapLayerType* );
HRESULT put_Layer(IRobotViewFeMapLayerType);
```

#### C#

```
public IRobotViewFeMapLayerType Layer { get; set; }
```

#### Visual Basic

```
Public Layer As IRobotViewFeMapLayerType
```

#### Description

Available since version 2.0.

### I.2.27.2.10 LayerArbitraryVal

#### C++

```
HRESULT get_LayerArbitraryVal(double* );
HRESULT put_LayerArbitraryVal(double);
```

**C#**

```
public double LayerArbitraryVal { get; set; }
```

**Visual Basic**

```
Public LayerArbitraryVal As Double
```

**Description**

Available since version 2.0.

### I.2.27.2.11 MapScale

**C++**

```
HRESULT get_MapScale(double*);  
HRESULT put_MapScale(double);
```

**C#**

```
public double MapScale { get; set; }
```

**Visual Basic**

```
Public MapScale As Double
```

**Description**

Available since version 2.0.

### I.2.27.2.12 Smoothing

**C++**

```
HRESULT get_Smoothing(IRobotViewFeMapSmoothingType*);  
HRESULT put_Smoothing(IRobotViewFeMapSmoothingType);
```

**C#**

```
public IRobotViewFeMapSmoothingType Smoothing { get; set; }
```

**Visual Basic**

```
Public Smoothing As IRobotViewFeMapSmoothingType
```

**Description**

Available since version 2.0.

### I.2.27.2.13 WithDescription

**C++**

```
HRESULT get_WithDescription(VARIANT_BOOL*);  
HRESULT put_WithDescription(VARIANT_BOOL);
```

**C#**

```
public bool WithDescription { get; set; }
```

**Visual Basic**

```
Public WithDescription As Boolean
```

**Description**

Available since version 2.0.

### I.2.27.3 IRobotViewFeMapParams Methods

The methods of the IRobotViewFeMapParams class are listed here.

## Public Methods

	Name	Description
	SetDirectionData (see page 1436)	

### I.2.27.3.1 SetDirectionData

#### C++

```
HRESULT SetDirectionData(IRobotGeoPoint3D* _dir_data);
```

#### C#

```
public void SetDirectionData(IRobotGeoPoint3D _dir_data);
```

#### Visual Basic

```
Public Sub SetDirectionData(ByRef _dir_data As IRobotGeoPoint3D)
```

#### Version

Available since version 3.5.

## I.2.28 IRobotViewFeMapLocalSystemType

#### C++

```
enum IRobotViewFeMapLocalSystemType;
```

#### C#

```
public enum IRobotViewFeMapLocalSystemType;
```

#### Visual Basic

```
Public Enum IRobotViewFeMapLocalSystemType
```

#### Members

Members	Description
I_VFMLST_AUTOMATIC = 0	Available since version 2.0.
I_VFMLST_CARTESIAN_ALONG_X = 1	Available since version 2.0.
I_VFMLST_CARTESIAN_ALONG_Y = 2	Available since version 2.0.
I_VFMLST_CARTESIAN_ALONG_Z = 3	Available since version 2.0.
I_VFMLST_CARTESIAN_ALONG_VECTOR = 4	Available since version 2.0.
I_VFMLST_POLAR_IN_NODE = 5	Available since version 2.0.
I_VFMLST_POLAR_IN_POINT = 6	Available since version 2.0.

## I.2.29 IRobotViewFeMapCrossPresentationType

#### C++

```
enum IRobotViewFeMapCrossPresentationType;
```

#### C#

```
public enum IRobotViewFeMapCrossPresentationType;
```

#### Visual Basic

```
Public Enum IRobotViewFeMapCrossPresentationType
```

## Members

Members	Description
I_VFCPT_NONE = 0	Available since version 2.0.
I_VFCPT_STRESSES = 1404	Available since version 2.0.
I_VFCPT_NORMAL_FORCES = 1405	Available since version 2.0.
I_VFCPT_MOMENTS = 1406	Available since version 2.0.

## Description

Manner of presenting crosses in the map view for surface finite elements. .

## I.2.30 IRobotViewFeMapLayerType

### C++

```
enum IRobotViewFeMapLayerType;
```

### C#

```
public enum IRobotViewFeMapLayerType;
```

### Visual Basic

```
Public Enum IRobotViewFeMapLayerType
```

## Members

Members	Description
I_VFMLT_UPPER = 0	Available since version 2.0.
I_VFMLT_MIDDLE = 1	Available since version 2.0.
I_VFMLT_LOWER = 2	Available since version 2.0.
I_VFMLT_MAXIMUM = 3	Available since version 2.0.
I_VFMLT_MINIMUM = 4	Available since version 2.0.
I_VFMLT_ABSOLUTE_MAXIMUM = 5	Available since version 2.0.
I_VFMLT_ARBITRARY = 6	Available since version 2.0.

## I.2.31 IRobotViewFeMapSmoothingType

### C++

```
enum IRobotViewFeMapSmoothingType;
```

### C#

```
public enum IRobotViewFeMapSmoothingType;
```

### Visual Basic

```
Public Enum IRobotViewFeMapSmoothingType
```

## Members

Members	Description
I_VFMST_NO_SMOOTHING = 0	Available since version 2.0.
I_VFMST_GLOBAL_SMOOTHING = 1	Available since version 2.0.
I_VFMST_SMOOTHING_WITH_PANEL = 2	Available since version 2.0.
I_VFMST_SMOOTHING_ACCORDING_TO_SELECTION = 3	Available since version 2.0.

## I.2.32 IRobotView2

### Class Hierarchy

#### C++

```
interface IRobotView2 : IRobotView;
```

#### C#

```
public interface IRobotView2 : IRobotView;
```

### Visual Basic

```
Public Interface IRobotView2
```

### Description

Extended graphical view interface. .

### Version

Available since version 3.

#### I.2.32.1 IRobotView2 Members

The following tables list the members exposed by IRobotView2.

##### Public Fields

	Name	Description
◆	DiagramMagnification (↗ see page 1391)	Diagram enlarging (scale: 1 to 10).
◆	ParamsBarMap (↗ see page 1391)	Parameters of displaying maps an bars Available since version 2.0.
◆	ParamsDiagram (↗ see page 1391)	Parameters of displaying diagrams on the view Available since version 2.0.
◆	ParamsDisplay (↗ see page 1391)	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
◆	ParamsFeMap (↗ see page 1392)	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
◆	ParamsPanelCut (↗ see page 1440)	Parameters of displaying panel cuts.
◆	Printable (↗ see page 1392)	Object representing the view that may be attached to the printout and printed .
◆	Projection (↗ see page 1392)	Current projection .
◆	Selection (↗ see page 1392)	Selection of structure components presented in the view .
◆	Title (↗ see page 1393)	View title .
◆	Type (↗ see page 1393)	View type .
◆	UserControl (↗ see page 1440)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
◆	Visible (↗ see page 1393)	View display flag .
◆	Window (↗ see page 1440)	

##### Public Methods

	Name	Description
♫	CopyToClipboard (↗ see page 1394)	The function inserts the view into the clipboard.
♫	GetRotationPoint (↗ see page 1394)	The function gives back a rotation center during a view rotation.
♫	GetScale (↗ see page 1394)	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
♫	GetSize (↗ see page 1395)	The function returns the size of the view in mm. .

	GetWorkPoint ( <a href="#">see page 1395</a> )	The function gives a work point back (on the XY plane it gives Z level) .
	GetZoom ( <a href="#">see page 1395</a> )	The function returns the current zoom - coordinates of view sides in the current coordinate system .
	IsLocal ( <a href="#">see page 1396</a> )	The function gives the status of a local coordinate system back.
	Redraw ( <a href="#">see page 1396</a> )	The function draws the view. .
	Rotate ( <a href="#">see page 1396</a> )	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
	SetGlobal ( <a href="#">see page 1396</a> )	The function sets a global coordinate system.
	SetLocal ( <a href="#">see page 1397</a> )	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
	SetRotationPoint ( <a href="#">see page 1397</a> )	The funcion sets the rotation center during the view rotation.
	SetScale ( <a href="#">see page 1397</a> )	The function assigns a given value to the scale.
	SetSize ( <a href="#">see page 1397</a> )	The function sets the view size in mm .
	SetWorkPoint ( <a href="#">see page 1398</a> )	The function sets a working point (for the XY plane Z level is set). .
	SetZoom ( <a href="#">see page 1398</a> )	The function sets the current zoom - coordinates of view sides in the current coordinate system .

### I.2.32.2 IRobotView2 Fields

The fields of the IRobotView2 class are listed here.

#### Public Fields

	Name	Description
	ParamsPanelCut ( <a href="#">see page 1440</a> )	Parameters of displaying panel cuts.
	UserControl ( <a href="#">see page 1440</a> )	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
	Window ( <a href="#">see page 1440</a> )	

#### I.2.32.2.1 ParamsPanelCut

##### C++

```
HRESULT get_ParamsPanelCut(IRobotViewPanelCutParams**);
```

##### C#

```
public IRobotViewPanelCutParams ParamsPanelCut { get; }
```

##### Visual Basic

```
Public ReadOnly ParamsPanelCut As IRobotViewPanelCutParams
```

##### Description

Parameters of displaying panel cuts.

##### Version

Available since version 3.

#### I.2.32.2.2 UserControl

##### C++

```
HRESULT get_UserControl(VARIANT_BOOL*);  
HRESULT put_UserControl(VARIANT_BOOL);
```

##### C#

```
public bool UserControl { get; set; }
```

**Visual Basic**

```
Public UserControl As Boolean
```

**Description**

Flag that enables giving/taking back graphical view navigation to/from the interactive program user .

**Version**

Available since version 3.

**I.2.32.2.3 Window****C++**

```
HRESULT get_Window(IRobotWindow**);
```

**C#**

```
public IRobotWindow Window { get; }
```

**Visual Basic**

```
Public ReadOnly Window As IRobotWindow
```

**Version**

Available since version 3.

**I.2.33 IRobotViewDiagramPositionType****C++**

```
enum IRobotViewDiagramPositionType;
```

**C#**

```
public enum IRobotViewDiagramPositionType;
```

**Visual Basic**

```
Public Enum IRobotViewDiagramPositionType
```

**Members**

Members	Description
I_VDPT_NORMAL = 1	Diagram perpendicular to the panel plane. Available since version 3.
I_VDPT_IN_PLANE = 2	Diagram in the panel plane. Available since version 3.

**Description**

Available diagram positions with respect to the panel plane.

**Version**

Available since version 3.

**I.2.34 IRobotViewReinforcementResultType****C++**

```
enum IRobotViewReinforcementResultType;
```

**C#**

```
public enum IRobotViewReinforcementResultType;
```

**Visual Basic**

```
Public Enum IRobotViewReinforcementResultType
```

**Members**

Members	Description
I_VRRT_AX_TOP = 577	Top reinforcement area (X direction). Available since version 3.
I_VRRT_AX_BOTTOM = 579	Bottom reinforcement area (X direction). Available since version 3.
I_VRRT_AY_TOP = 578	Top reinforcement area (Y direction). Available since version 3.
I_VRRT_AY_BOTTOM = 580	Bottom reinforcement area (Y direction). Available since version 3.
I_VRRT_E_AX_TOP = 699	Top reinforcement spacing (X direction). Available since version 3.
I_VRRT_E_AX_BOTTOM = 701	Bottom reinforcement spacing (X direction). Available since version 3.
I_VRRT_E_AY_TOP = 700	Top reinforcement spacing (Y direction). Available since version 3.
I_VRRT_E_AY_BOTTOM = 702	Bottom reinforcement spacing (Y direction). Available since version 3.
I_VRRT_A_MIN = 1119	Minimal reinforcement area for one layer and one reinforcement direction. Available since version 3.
I_VRRT_AX = 1116	Cracking (X axis direction). Available since version 3.
I_VRRT_AY = 1117	Cracking (Y axis direction). Available since version 3.
I_VRRT_F = 1118	Deflection value. Available since version 3.

**Description**

Available reinforcement results presented in the graphical view. .

**Version**

Available since version 3.

**I.2.35 IRobotViewGlobalAnalysisParams****Class Hierarchy****C++**

```
interface IRobotViewGlobalAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotViewGlobalAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotViewGlobalAnalysisParams
```

**Description**

Interface controlling display of global analysis results. .

**Version**

Available since version 3.

**I.2.35.1 IRobotViewGlobalAnalysisParams Members**

The following tables list the members exposed by IRobotViewGlobalAnalysisParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Results ( <a href="#">see page 1443</a> )	Method of result display .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	IsOn ( <a href="#">see page 1443</a> )	Function checks status of the indicated parameter..
◆	Set ( <a href="#">see page 1444</a> )	Function sets status for the indicated parameter..

**I.2.35.2 IRobotViewGlobalAnalysisParams Fields**

The fields of the IRobotViewGlobalAnalysisParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Results ( <a href="#">see page 1443</a> )	Method of result display .

**I.2.35.2.1 Results****C++**

```
HRESULT get_Results(IRobotViewGlobalAnalysisResultsParams**);
```

**C#**

```
public IRobotViewGlobalAnalysisResultsParams Results { get; }
```

**Visual Basic**

```
Public ReadOnly Results As IRobotViewGlobalAnalysisResultsParams
```

**Description**

Method of result display .

**Version**

Available since version 3.

**I.2.35.3 IRobotViewGlobalAnalysisParams Methods**

The methods of the IRobotViewGlobalAnalysisParams class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	IsOn ( <a href="#">see page 1443</a> )	Function checks status of the indicated parameter..
◆	Set ( <a href="#">see page 1444</a> )	Function sets status for the indicated parameter..

**I.2.35.3.1 IsOn****C++**

```
HRESULT IsOn(IRobotViewGlobalAnalysisParamsType _type, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsOn(IRobotViewGlobalAnalysisParamsType _type);
```

**Visual Basic**

```
Public Function IsOn(_type As IRobotViewGlobalAnalysisParamsType) As Boolean
```

**Description**

Function checks status of the indicated parameter. .

**Version**

Available since version 3.

**I.2.35.3.2 Set****C++**

```
HRESULT Set(IRobotViewGlobalAnalysisParamsType _type, VARIANT_BOOL _status);
```

**C#**

```
public void Set(IRobotViewGlobalAnalysisParamsType _type, bool _status);
```

**Visual Basic**

```
Public Sub Set(_type As IRobotViewGlobalAnalysisParamsType, _status As Boolean)
```

**Description**

Function sets status for the indicated parameter. .

**Version**

Available since version 3.

**I.2.36 IRobotViewGlobalAnalysisResultsParams****Class Hierarchy****C++**

```
interface IRobotViewGlobalAnalysisResultsParams : IDispatch;
```

**C#**

```
public interface IRobotViewGlobalAnalysisResultsParams;
```

**Visual Basic**

```
Public Interface IRobotViewGlobalAnalysisResultsParams
```

**Description**

Interface controlling result display.

**Version**

Available since version 3.

**I.2.36.1 IRobotViewGlobalAnalysisResultsParams Members**

The following tables list the members exposed by IRobotViewGlobalAnalysisResultsParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	NPointsValue (see page 1445)	Number of points along the bar length.

❖	RelativeValue ( <a href="#">see page 1445</a> )	Relative point coordinate.
❖	Type ( <a href="#">see page 1445</a> )	Method of result presentation.

### I.2.36.2 IRobotViewGlobalAnalysisResultsParams Fields

The fields of the IRobotViewGlobalAnalysisResultsParams class are listed here.

#### Public Fields

	Name	Description
❖	NPointsValue ( <a href="#">see page 1445</a> )	Number of points along the bar length.
❖	RelativeValue ( <a href="#">see page 1445</a> )	Relative point coordinate.
❖	Type ( <a href="#">see page 1445</a> )	Method of result presentation.

#### I.2.36.2.1 NPointsValue

##### C++

```
HRESULT get_NPointsValue(long*);
HRESULT put_NPointsValue(long);
```

##### C#

```
public long NPointsValue { get; set; }
```

##### Visual Basic

```
Public NPointsValue As long
```

##### Description

Number of points along the bar length.

##### Version

Available since version 3.

#### I.2.36.2.2 RelativeValue

##### C++

```
HRESULT get_RelativeValue(double*);
HRESULT put_RelativeValue(double);
```

##### C#

```
public double RelativeValue { get; set; }
```

##### Visual Basic

```
Public RelativeValue As double
```

##### Description

Relative point coordinate.

##### Version

Available since version 3.

#### I.2.36.2.3 Type

##### C++

```
HRESULT get_Type(IRobotViewGlobalAnalysisResultsType*);
HRESULT put_Type(IRobotViewGlobalAnalysisResultsType);
```

##### C#

```
public IRobotViewGlobalAnalysisResultsType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRobotViewGlobalAnalysisResultsType
```

**Description**

Method of result presentation.

**Version**

Available since version 3.

**I.2.37 IRobotViewGlobalAnalysisResultsType****C++**

```
enum IRobotViewGlobalAnalysisResultsType;
```

**C#**

```
public enum IRobotViewGlobalAnalysisResultsType;
```

**Visual Basic**

```
Public Enum IRobotViewGlobalAnalysisResultsType
```

**Members**

Members	Description
I_VGART_N_POINTS = 1	Available since version 3.
I_VGART_RELATIVE = 2	Available since version 3.

**Description**

Types of result display.

**Version**

Available since version 3.

**I.2.38 IRobotViewGlobalAnalysisParamsType****C++**

```
enum IRobotViewGlobalAnalysisParamsType;
```

**C#**

```
public enum IRobotViewGlobalAnalysisParamsType;
```

**Visual Basic**

```
Public Enum IRobotViewGlobalAnalysisParamsType
```

**Members**

Members	Description
I_VGAPT_FORCE_FX = 1	Available since version 3.
I_VGAPT_FORCE_FY = 2	Available since version 3.
I_VGAPT_FORCE_FZ = 3	Available since version 3.
I_VGAPT_FORCE_MX = 4	Available since version 3.
I_VGAPT_FORCE_MY = 5	Available since version 3.
I_VGAPT_FORCE_MZ = 6	Available since version 3.
I_VGAPT_FORCE_KX = 7	Available since version 3.
I_VGAPT_FORCE_KY = 8	Available since version 3.
I_VGAPT_STRESS_NORMAL = 9	Available since version 3.

I_VGAPT_STRESS_EXTREME = 10	Available since version 3.
I_VGAPT_STRESS_BENDING_SMY = 11	Available since version 3.
I_VGAPT_STRESS_BENDING_SMZ = 12	Available since version 3.
I_VGAPT_STRESS_AXIAL_SXFX = 13	Available since version 3.
I_VGAPT_STRESS_SHEAR = 14	Available since version 3.
I_VGAPT_STRESS_TORSION = 15	Available since version 3.
I_VGAPT_DESIGN_RATIO = 16	Available since version 3.
I_VGAPT_DESIGN_MEMBER_LENGTH = 17	Available since version 3.
I_VGAPT_DESIGN_SLENDERNESS_LAY = 18	Available since version 3.
I_VGAPT_DESIGN_SLENDERNESS_LAZ = 19	Available since version 3.

**Description**

Parameter types.

**Version**

Available since version 3.

**I.2.39 IRobotViewGlobalAnalysis****Class Hierarchy****C++**

```
interface IRobotViewGlobalAnalysis : IRobotView2;
```

**C#**

```
public interface IRobotViewGlobalAnalysis : IRobotView2;
```

**Visual Basic**

```
Public Interface IRobotViewGlobalAnalysis
```

**Description**

View presenting global analysis results..

**Version**

Available since version 3.

**I.2.39.1 IRobotViewGlobalAnalysis Members**

The following tables list the members exposed by IRobotViewGlobalAnalysis.

**Public Fields**

	Name	Description
❖	DiagramMagnification ( <a href="#">see page 1391</a> )	Diagram enlarging (scale: 1 to 10).
❖	ParamsBarMap ( <a href="#">see page 1391</a> )	Parameters of displaying maps an bars Available since version 2.0.
❖	ParamsDiagram ( <a href="#">see page 1391</a> )	Parameters of displaying diagrams on the view Available since version 2.0.
❖	ParamsDisplay ( <a href="#">see page 1391</a> )	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
❖	ParamsFeMap ( <a href="#">see page 1392</a> )	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
❖	ParamsGlobalAnalysis ( <a href="#">see page 1448</a> )	Display parameters for global analysis results.
❖	ParamsPanelCut ( <a href="#">see page 1440</a> )	Parameters of displaying panel cuts.

◆	Printable ( [ see page 1392)	Object representing the view that may be attached to the printout and printed .
◆	Projection ( [ see page 1392)	Current projection .
◆	Selection ( [ see page 1392)	Selection of structure components presented in the view .
◆	Title ( [ see page 1393)	View title .
◆	Type ( [ see page 1393)	View type .
◆	UserControl ( [ see page 1440)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
◆	Visible ( [ see page 1393)	View display flag .
◆	Window ( [ see page 1440)	

**Public Methods**

	Name	Description
◆	CopyToClipboard ( [ see page 1394)	The function inserts the view into the clipboard.
◆	GetRotationPoint ( [ see page 1394)	The function gives back a rotation center during a view rotation.
◆	GetScale ( [ see page 1394)	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
◆	GetSize ( [ see page 1395)	The function returns the size of the view in mm. .
◆	GetWorkPoint ( [ see page 1395)	The function gives a work point back (on the XY plane it gives Z level) .
◆	GetZoom ( [ see page 1395)	The function returns the current zoom - coordinates of view sides in the current coordinate system .
◆	IsLocal ( [ see page 1396)	The function gives the status of a local coordinate system back.
◆	MakeScreenCapture ( [ see page 1449)	
◆	Redraw ( [ see page 1396)	The function draws the view. .
◆	Rotate ( [ see page 1396)	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
◆	SetGlobal ( [ see page 1396)	The function sets a global coordinate system.
◆	SetLocal ( [ see page 1397)	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
◆	SetRotationPoint ( [ see page 1397)	The funcion sets the rotation center during the view rotation.
◆	SetScale ( [ see page 1397)	The function assigns a given value to the scale.
◆	SetSize ( [ see page 1397)	The function sets the view size in mm .
◆	SetWorkPoint ( [ see page 1398)	The function sets a working point (for the XY plane Z level is set). .
◆	SetZoom ( [ see page 1398)	The function sets the current zoom - coordinates of view sides in the current coordinate system .

**I.2.39.2 IRobotViewGlobalAnalysis Fields**

The fields of the IRobotViewGlobalAnalysis class are listed here.

**Public Fields**

	Name	Description
◆	ParamsGlobalAnalysis ( [ see page 1448)	Display parameters for global analysis results.

**I.2.39.2.1 ParamsGlobalAnalysis****C++**

```
HRESULT get_ParamsGlobalAnalysis(IRobotViewGlobalAnalysisParams**);
```

**C#**

```
public IRobotViewGlobalAnalysisParams ParamsGlobalAnalysis { get; }
```

**Visual Basic**

```
Public ReadOnly ParamsGlobalAnalysis As IRobotViewGlobalAnalysisParams
```

**Description**

Display parameters for global analysis results.

**Version**

Available since version 3.

**I.2.39.3 IRobotViewGlobalAnalysis Methods**

The methods of the IRobotViewGlobalAnalysis class are listed here.

**Public Methods**

	Name	Description
=	MakeScreenCapture ( [ see page 1449 )	

**I.2.39.3.1 MakeScreenCapture****C++**

```
HRESULT MakeScreenCapture(IRobotViewScreenCaptureParams* _sc_params);
```

**C#**

```
public void MakeScreenCapture(IRobotViewScreenCaptureParams _sc_params);
```

**Visual Basic**

```
Public Sub MakeScreenCapture(ByRef _sc_params As IRobotViewScreenCaptureParams)
```

**I.2.40 IRobotViewScreenCaptureParams****Class Hierarchy****C++**

```
interface IRobotViewScreenCaptureParams : IDispatch;
```

**C#**

```
public interface IRobotViewScreenCaptureParams;
```

**Visual Basic**

```
Public Interface IRobotViewScreenCaptureParams
```

**Description**

Parameters of a screen capture for the graphical view.

**I.2.40.1 IRobotViewScreenCaptureParams Members**

The following tables list the members exposed by IRobotViewScreenCaptureParams.

**Public Fields**

	Name	Description
=	Comment ( [ see page 1450 )	Comment text to be added to the screen capture.

❖	IncludeDateAndTime (see page 1450)	The flag indicating if the current date and time should be added automatically to the screen capture.
❖	Name (see page 1451)	Screen capture name.
❖	Orientation (see page 1451)	Screen capture orientation (landscape or portait).
❖	Resolution (see page 1451)	
❖	ScaleAutomatic (see page 1451)	True value denotes selection of the automatic scale, while False value indicates selection of the user scale defined by ScaleValue (see page 1452) value.
❖	ScaleValue (see page 1452)	User scale value - distance expressed in meters (m) which corresponds to one centimeter (1 cm) in the graphical view.
❖	UpdateType (see page 1452)	

#### I.2.40.2 IRobotViewScreenCaptureParams Fields

The fields of the IRobotViewScreenCaptureParams class are listed here.

##### Public Fields

	Name	Description
❖	Comment (see page 1450)	Comment text to be added to the screen capture.
❖	IncludeDateAndTime (see page 1450)	The flag indicating if the current date and time should be added automatically to the screen capture.
❖	Name (see page 1451)	Screen capture name.
❖	Orientation (see page 1451)	Screen capture orientation (landscape or portait).
❖	Resolution (see page 1451)	
❖	ScaleAutomatic (see page 1451)	True value denotes selection of the automatic scale, while False value indicates selection of the user scale defined by ScaleValue (see page 1452) value.
❖	ScaleValue (see page 1452)	User scale value - distance expressed in meters (m) which corresponds to one centimeter (1 cm) in the graphical view.
❖	UpdateType (see page 1452)	

#### I.2.40.2.1 Comment

##### C++

```
HRESULT get_Comment(BSTR* );
HRESULT put_Comment(BSTR);
```

##### C#

```
public String Comment { get; set; }
```

##### Visual Basic

```
Public Comment As String
```

##### Description

Comment text to be added to the screen capture.

##### Version

Available since version 12.

#### I.2.40.2.2 IncludeDateAndTime

##### C++

```
HRESULT get_IncludeDateAndTime(VARIANT_BOOL* );
HRESULT put_IncludeDateAndTime(VARIANT_BOOL);
```

**C#**

```
public bool IncludeDateAndTime { get; set; }
```

**Visual Basic**

```
Public IncludeDateAndTime As Boolean
```

**Description**

The flag indicating if the current date and time should be added automatically to the screen capture.

**Version**

Available since version 12.

#### I.2.40.2.3 Name

**C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Screen capture name.

#### I.2.40.2.4 Orientation

**C++**

```
HRESULT get_Orientation(IRobotPageSetupOrientation* );
HRESULT put_Orientation(IRobotPageSetupOrientation);
```

**C#**

```
public IRobotPageSetupOrientation Orientation { get; set; }
```

**Visual Basic**

```
Public Orientation As IRobotPageSetupOrientation
```

**Description**

Screen capture orientation (landscape or portait).

#### I.2.40.2.5 Resolution

**C++**

```
HRESULT get_Resolution(IRobotViewScreenCaptureResolution* );
HRESULT put_Resolution(IRobotViewScreenCaptureResolution);
```

**C#**

```
public IRobotViewScreenCaptureResolution Resolution { get; set; }
```

**Visual Basic**

```
Public Resolution As IRobotViewScreenCaptureResolution
```

**Version**

Available since version 12.

### I.2.40.2.6 ScaleAutomatic

**C++**

```
HRESULT get_ScaleAutomatic(VARIANT_BOOL* );
HRESULT put_ScaleAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool ScaleAutomatic { get; set; }
```

**Visual Basic**

```
Public ScaleAutomatic As Boolean
```

**Description**

True value denotes selection of the automatic scale, while False value indicates selection of the user scale defined by ScaleValue (see page 1452) value.

### I.2.40.2.7 ScaleValue

**C++**

```
HRESULT get_ScaleValue(double* );
HRESULT put_ScaleValue(double);
```

**C#**

```
public double ScaleValue { get; set; }
```

**Visual Basic**

```
Public ScaleValue As Double
```

**Description**

User scale value - distance expressed in meters (m) which corresponds to one centimeter (1 cm) in the graphical view.

### I.2.40.2.8 UpdateType

**C++**

```
HRESULT get_UpdateType(IRobotViewScreenCaptureUpdateType* );
HRESULT put_UpdateType(IRobotViewScreenCaptureUpdateType);
```

**C#**

```
public IRobotViewScreenCaptureUpdateType UpdateType { get; set; }
```

**Visual Basic**

```
Public UpdateType As IRobotViewScreenCaptureUpdateType
```

## I.2.41 IRobotViewScreenCaptureUpdateType

**C++**

```
enum IRobotViewScreenCaptureUpdateType;
```

**C#**

```
public enum IRobotViewScreenCaptureUpdateType;
```

**Visual Basic**

```
Public Enum IRobotViewScreenCaptureUpdateType
```

## Members

Members	Description
I_SCUT_UPDATED_UPON_PRINTING = 1	The view contents will be updated automatically while printing.
I_SCUT_CURRENT_VIEW = 2	The view will be printed in the form it had while making a screen capture - the view contents will not be updated.
I_SCUT_UPDATED_WHOLE_STRUCTURE = 3	The view contents will be updated while printing in such a way so that the view presents the whole current structure.
I_SCUT_COPY_TO_CLIPBOARD = 4	The screen capture will be saved only to the clipboard. Available since version 12.

## Description

Available methods of updating the contents of a screen capture for the graphical view.

## I.2.42 IRobotView3

### Class Hierarchy

#### C++

```
interface IRobotView3 : IRobotView2;
```

#### C#

```
public interface IRobotView3 : IRobotView2;
```

### Visual Basic

```
Public Interface IRobotView3
```

## I.2.42.1 IRobotView3 Members

The following tables list the members exposed by IRobotView3.

### Public Fields

	Name	Description
◆	DiagramMagnification ( [ see page 1391)	Diagram enlarging (scale: 1 to 10).
◆	ParamsBarMap ( [ see page 1391)	Parameters of displaying maps an bars Available since version 2.0.
◆	ParamsDiagram ( [ see page 1391)	Parameters of displaying diagrams on the view Available since version 2.0.
◆	ParamsDisplay ( [ see page 1391)	Display parameters (a component enabling control of individual attribute display) Available since version 2.0.
◆	ParamsFeMap ( [ see page 1392)	Display parameters of maps presenting results for surface finite elements Available since version 2.0.
◆	ParamsPanelCut ( [ see page 1440)	Parameters of displaying panel cuts.
◆	Printable ( [ see page 1392)	Object representing the view that may be attached to the printout and printed .
◆	Projection ( [ see page 1392)	Current projection .
◆	Selection ( [ see page 1392)	Selection of structure components presented in the view .
◆	Title ( [ see page 1393)	View title .
◆	Type ( [ see page 1393)	View type .
◆	UserControl ( [ see page 1440)	Flag that enables giving/taking back graphical view navigation to/from the interactive program user .
◆	Visible ( [ see page 1393)	View display flag .
◆	Window ( [ see page 1440)	

## Public Methods

	Name	Description
💡	CopyToClipboard ( [ see page 1394] )	The function inserts the view into the clipboard.
💡	GetRotationPoint ( [ see page 1394] )	The function gives back a rotation center during a view rotation.
💡	GetScale ( [ see page 1394] )	The function gives a value of the given scale back. A set of available scales describes RobotViewScaleType type.
💡	GetSize ( [ see page 1395] )	The function returns the size of the view in mm. .
💡	GetWorkPoint ( [ see page 1395] )	The function gives a work point back (on the XY plane it gives Z level) .
💡	GetZoom ( [ see page 1395] )	The function returns the current zoom - coordinates of view sides in the current coordinate system .
💡	IsLocal ( [ see page 1396] )	The function gives the status of a local coordinate system back.
💡	MakeScreenCapture ( [ see page 1454] )	Function makes a screen capture of a graphical view.
💡	Redraw ( [ see page 1396] )	The function draws the view. .
💡	Rotate ( [ see page 1396] )	The function rotates structure view with respect to the indicated global coordinate axis, with the rotation point recognized.
💡	SetGlobal ( [ see page 1396] )	The function sets a global coordinate system.
💡	SetLocal ( [ see page 1397] )	The function sets a local coordinate system. If the given points do not describe a local plane in the unambiguous way the function gives a zero value (False) back.
💡	SetRotationPoint ( [ see page 1397] )	The funcion sets the rotation center during the view rotation.
💡	SetScale ( [ see page 1397] )	The function assigns a given value to the scale.
💡	SetSize ( [ see page 1397] )	The function sets the view size in mm .
💡	SetWorkPoint ( [ see page 1398] )	The function sets a working point (for the XY plane Z level is set). .
💡	SetZoom ( [ see page 1398] )	The function sets the current zoom - coordinates of view sides in the current coordinate system .

### I.2.42.2 IRobotView3 Methods

The methods of the IRobotView3 class are listed here.

## Public Methods

	Name	Description
💡	MakeScreenCapture ( [ see page 1454] )	Function makes a screen capture of a graphical view.

### I.2.42.2.1 MakeScreenCapture

#### C++

```
HRESULT MakeScreenCapture( IRobotViewScreenCaptureParams* _sc_params );
```

#### C#

```
public void MakeScreenCapture( IRobotViewScreenCaptureParams _sc_params );
```

#### Visual Basic

```
Public Sub MakeScreenCapture(ByRef _sc_params As IRobotViewScreenCaptureParams)
```

#### Description

Function makes a screen capture of a graphical view.

## I.2.43 IRobotViewDiagramValueType

### C++

```
enum IRobotViewDiagramValueType;
```

### C#

```
public enum IRobotViewDiagramValueType;
```

### Visual Basic

```
Public Enum IRobotViewDiagramValueType
```

### Members

Members	Description
I_VDVT_ALL = 0	Available since version 8.1.
I_VDVT_LOCAL_EXTREMES = 1	Available since version 8.1.
I_VDVT_GLOBAL_EXTREMES = 2	Available since version 8.1.

### Description

Types of values displayed on diagrams.

### Version

Available since version 8.1.

## I.2.44 IRobotViewScreenCaptureResolution

### C++

```
enum IRobotViewScreenCaptureResolution;
```

### C#

```
public enum IRobotViewScreenCaptureResolution;
```

### Visual Basic

```
Public Enum IRobotViewScreenCaptureResolution
```

### Members

Members	Description
I_VSCR_DEFAULT = 0	Available since version 12.
I_VSCR_2048 = 1	Available since version 12.
I_VSCR_3072 = 2	Available since version 12.
I_VSCR_4096 = 3	Available since version 12.

### Version

Available since version 12.

## I.3 Views for RTF format files

### Interfaces

	Name	Description
IRobotRtfView (see page 1455)		Files in RTF format are frequently used to present data or results. Robot enriched the format with the possibility of introducing into an RTF file the variables defined in the program. A variable is written to file in the form of its name, marked with the "@" sign at both ends. The view RobotRtfView automatically extends variables to give them their current value.

## I.3.1 IRobotRtfView

### Class Hierarchy

#### C++

```
interface IRobotRtfView : IDispatch;
```

#### C#

```
public interface IRobotRtfView;
```

### Visual Basic

```
Public Interface IRobotRtfView
```

### Description

Files in RTF format are frequently used to present data or results. Robot enriched the format with the possibility of introducing into an RTF file the variables defined in the program. A variable is written to file in the form of its name, marked with the "@" sign at both ends. The view RobotRtfView automatically extends variables to give them their current value.

### I.3.1.1 IRobotRtfView Members

The following tables list the members exposed by IRobotRtfView.

#### Public Fields

	Name	Description
◆	Printable (see page 1456)	Object representing a view, that may be added to the print-out and printed.

#### Public Methods

	Name	Description
◆	AppendFromFile (see page 1457)	The function pastes the contents of the indicated view at the end of the view. .
◆	Evaluate (see page 1457)	The function extends all the variables in the text and changes their names to their current values. There is no corresponding undo operation. .
◆	LoadFromFile (see page 1457)	The function reads the indicated file in RTF format. .
◆	SaveToFile (see page 1458)	The function saves the content of the view to the indicated file. If the "as_template" parameter is not equal zero (set to True), the view contents will be written as a template (variable names and not their values will be saved to file).

### I.3.1.2 IRobotRtfView Fields

The fields of the IRobotRtfView class are listed here.

#### Public Fields

	Name	Description
◆	Printable (see page 1456)	Object representing a view, that may be added to the print-out and printed.

### I.3.1.2.1 Printable

#### C++

```
HRESULT get_Printable(IRobotPrintable**);
```

#### C#

```
public IRobotPrintable Printable { get; }
```

## Visual Basic

```
Public ReadOnly Printable As IRobotPrintable
```

### Description

Object representing a view, that may be added to the print-out and printed.

## I.3.1.3 IRobotRtfView Methods

The methods of the IRobotRtfView class are listed here.

### Public Methods

	Name	Description
💡	AppendFromFile (see page 1457)	The function pastes the contents of the indicated view at the end of the view..
💡	Evaluate (see page 1457)	The function extends all the variables in the text and changes their names to their current values. There is no corresponding undo operation. .
💡	LoadFromFile (see page 1457)	The function reads the indicated file in RTF format..
💡	SaveToFile (see page 1458)	The function saves the content of the view to the indicated file. If the "as_template" parameter is not equal zero (set to True), the view contents will be written as a template (variable names and not their values will be saved to file).

### I.3.1.3.1 AppendFromFile

#### C++

```
HRESULT AppendFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

#### C#

```
public bool AppendFromFile(String _file_path);
```

## Visual Basic

```
Public Function AppendFromFile(_file_path As String) As Boolean
```

### Description

The function pastes the contents of the indicated view at the end of the view..

### I.3.1.3.2 Evaluate

#### C++

```
HRESULT Evaluate();
```

#### C#

```
public void Evaluate();
```

## Visual Basic

```
Public Sub Evaluate()
```

### Description

The function extends all the variables in the text and changes their names to their current values. There is no corresponding undo operation..

### I.3.1.3.3 LoadFromFile

**C++**

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromFile(String _file_path);
```

**Visual Basic**

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

**Description**

The function reads the indicated file in RTF format. .

### I.3.1.3.4 SaveToFile

**C++**

```
HRESULT SaveToFile(BSTR _file_path, VARIANT_BOOL _as_template = false, VARIANT_BOOL* ret);
```

**C#**

```
public bool SaveToFile(String _file_path, bool _as_template = false);
```

**Visual Basic**

```
Public Function SaveToFile(_file_path As String, Optional _as_template As Boolean = false) As Boolean
```

**Description**

The function saves the content of the view to the indicated file. If the "as\_template" parameter is not equal zero (set to True), the view contents will be written as a template (variable names and not their values will be saved to file).

## I.4 Screen captures

The Robot program allows saving the current state (settings and/or contents) of a selected view into the archive to be used later at the stage of creating documentation. It is performed by making a screen capture of a selected view. The screen captures are identified by their names. .

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotScreenCaptureMngr (see page 1458)	An object manages screen captures defined in the current project. .

### I.4.1 IRobotScreenCaptureMngr

**Class Hierarchy**

**C++**

```
interface IRobotScreenCaptureMngr : IDispatch;
```

**C#**

```
public interface IRobotScreenCaptureMngr;
```

**Visual Basic**

```
Public Interface IRobotScreenCaptureMngr
```

## Description

An object manages screen captures defined in the current project. .

### I.4.1.1 IRobotScreenCaptureMngr Members

The following tables list the members exposed by IRobotScreenCaptureMngr.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1459</a> )	Number of defined screen captures Available since version 1.7.

#### Public Methods

	Name	Description
◆	Find ( <a href="#">see page 1459</a> )	Function returns index of a screen capture with a specified name. If screen capture with this name does not exist, then zero value is returned. Available since version 1.7.
◆	Get ( <a href="#">see page 1460</a> )	Function returns name of a screen capture with a specified index. Screen captures are indexed from 1 to Count ( <a href="#">see page 1459</a> ). Available since version 1.7.
◆	Remove ( <a href="#">see page 1460</a> )	Function deletes the archive containing a screen capture with a specified name from the project. Available since version 1.7.

### I.4.1.2 IRobotScreenCaptureMngr Fields

The fields of the IRobotScreenCaptureMngr class are listed here.

#### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1459</a> )	Number of defined screen captures Available since version 1.7.

#### I.4.1.2.1 Count

##### C++

```
HRESULT get_Count(long*);
```

##### C#

```
public long Count { get; }
```

##### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of defined screen captures Available since version 1.7.

### I.4.1.3 IRobotScreenCaptureMngr Methods

The methods of the IRobotScreenCaptureMngr class are listed here.

#### Public Methods

	Name	Description
◆	Find ( <a href="#">see page 1459</a> )	Function returns index of a screen capture with a specified name. If screen capture with this name does not exist, then zero value is returned. Available since version 1.7.
◆	Get ( <a href="#">see page 1460</a> )	Function returns name of a screen capture with a specified index. Screen captures are indexed from 1 to Count ( <a href="#">see page 1459</a> ). Available since version 1.7.

	Remove ( <a href="#">see page 1460</a> )	Function deletes the archive containing a screen capture with a specified name from the project. Available since version 1.7.
-----------------------------------------------------------------------------------	------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------

#### I.4.1.3.1 Find

**C++**

```
HRESULT Find(BSTR _sc_name, long* ret);
```

**C#**

```
public long Find(String _sc_name);
```

**Visual Basic**

```
Public Function Find(_sc_name As String) As long
```

**Description**

Function returns index of a screen capture with a specified name. If screen capture with this name does not exist, then zero value is returned. Available since version 1.7.

#### I.4.1.3.2 Get

**C++**

```
HRESULT Get(long _idx, BSTR* ret);
```

**C#**

```
public String Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As String
```

**Description**

Function returns name of a screen capture with a specified index. Screen captures are indexed from 1 to Count ( [see page 1459](#) ). Available since version 1.7.

#### I.4.1.3.3 Remove

**C++**

```
HRESULT Remove(BSTR _sc_name);
```

**C#**

```
public void Remove(String _sc_name);
```

**Visual Basic**

```
Public Sub Remove(_sc_name As String)
```

**Description**

Function deletes the archive containing a screen capture with a specified name from the project. Available since version 1.7.

### I.5 Views for HTML format files

Available since version 7.5.

**Interfaces**

	Name	Description
	IRobotHtmlView ( <a href="#">see page 1460</a> )	

## I.5.1 IRobotHtmlView

### Class Hierarchy

#### C++

```
interface IRobotHtmlView : IDispatch;
```

#### C#

```
public interface IRobotHtmlView;
```

### Visual Basic

```
Public Interface IRobotHtmlView
```

### Version

Available since version 7.5.

### I.5.1.1 IRobotHtmlView Members

The following tables list the members exposed by IRobotHtmlView.

#### Public Fields

	Name	Description
◆	Printable (see page 1461)	Object representing a view that may be attached to the printout and printed.

#### Public Methods

	Name	Description
☞	LoadFromFile (see page 1462)	Function loads an HTML format file.
☞	SaveToFile (see page 1462)	Function saves the view contents to the specified HTML format file.

### I.5.1.2 IRobotHtmlView Fields

The fields of the IRobotHtmlView class are listed here.

#### Public Fields

	Name	Description
◆	Printable (see page 1461)	Object representing a view that may be attached to the printout and printed.

### I.5.1.2.1 Printable

#### C++

```
HRESULT get_Printable(IRobotPrintable**);
```

#### C#

```
public IRobotPrintable Printable { get; }
```

### Visual Basic

```
Public ReadOnly Printable As IRobotPrintable
```

### Description

Object representing a view that may be attached to the printout and printed.

### Version

Available since version 7.5.

### I.5.1.3 IRobotHtmlView Methods

The methods of the IRobotHtmlView class are listed here.

#### Public Methods

	Name	Description
💡	LoadFromFile (see page 1462)	Function loads an HTML format file.
💡	SaveToFile (see page 1462)	Function saves the view contents to the specified HTML format file.

#### I.5.1.3.1 LoadFromFile

##### C++

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

##### C#

```
public bool LoadFromFile(String _file_path);
```

##### Visual Basic

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

##### Description

Function loads an HTML format file.

##### Version

Available since version 7.5.

#### I.5.1.3.2 SaveToFile

##### C++

```
HRESULT SaveToFile(BSTR _file_path, VARIANT_BOOL* ret);
```

##### C#

```
public bool SaveToFile(String _file_path);
```

##### Visual Basic

```
Public Function SaveToFile(_file_path As String) As Boolean
```

##### Description

Function saves the view contents to the specified HTML format file.

##### Version

Available since version 7.5.

## I.6 IRobotViewMngr

#### Class Hierarchy

##### C++

```
interface IRobotViewMngr : IDispatch;
```

##### C#

```
public interface IRobotViewMngr;
```

## Visual Basic

```
Public Interface IRobotViewMngr
```

### Description

View manager controls all the views and layouts available in Robot. .

## I.6.1 IRobotViewMngr Members

The following tables list the members exposed by IRobotViewMngr.

### Public Fields

	Name	Description
❖	CurrentLayout ( <a href="#">see page 1464</a> )	Identifier of active layout.
❖	NewViewAsTab ( <a href="#">see page 1464</a> )	Flag indicating if a new view is created as a tab.
❖	RecycleViews ( <a href="#">see page 1464</a> )	
❖	TableCount ( <a href="#">see page 1464</a> )	Number of opened tables.
❖	ViewCount ( <a href="#">see page 1465</a> )	Number of opened graphical views.

### Public Methods

	Name	Description
❖	CreateFromSc ( <a href="#">see page 1466</a> )	Function creates and returns a view on the basis of a screen capture with the specified name. Type of the created view depends on the screen capture contents. If structure view manager is unable to make any of the defined view interfaces available, then, IRobotPrintable ( <a href="#">see page 1470</a> ) interface is returned. If a screen capture with the specified name does not exist or if during its reading from the archive an error occurs, then, empty reference will be returned (zero from the level of C language, Nothing from the level of Visual Basic). Available since version 1.7.
❖	CreateRtfView ( <a href="#">see page 1466</a> )	The function creates a view for files in RTF format.
❖	CreateTable ( <a href="#">see page 1466</a> )	The function creates and returns a table of the indicated type that will present the appropriate type of data. .
❖	CreateView ( <a href="#">see page 1466</a> )	The function creates and returns the graphical view of the indicated type. .
❖	GetTable ( <a href="#">see page 1467</a> )	Function returns the interface for the table of the specified index. .
❖	GetView ( <a href="#">see page 1467</a> )	Function returns the interface for the graphical view of the indicated index. .
❖	Refresh ( <a href="#">see page 1467</a> )	The function results in refreshing all the displayed views. .
❖	ShowDialog ( <a href="#">see page 1468</a> )	Function displays a dialog of the given identifier. For modal dialogs the function returns a code transferred originally as a result of the dialog operation. For non-modal dialogs the function returns a True/False status informing whether the dialog displaying failed or not. .

## I.6.2 IRobotViewMngr Fields

The fields of the IRobotViewMngr class are listed here.

### Public Fields

	Name	Description
❖	CurrentLayout ( <a href="#">see page 1464</a> )	Identifier of active layout.
❖	NewViewAsTab ( <a href="#">see page 1464</a> )	Flag indicating if a new view is created as a tab.
❖	RecycleViews ( <a href="#">see page 1464</a> )	
❖	TableCount ( <a href="#">see page 1464</a> )	Number of opened tables.
❖	ViewCount ( <a href="#">see page 1465</a> )	Number of opened graphical views.

### I.6.2.1 CurrentLayout

#### C++

```
HRESULT get_CurrentLayout(IRobotLayoutId* );
HRESULT put_CurrentLayout(IRobotLayoutId);
```

#### C#

```
public IRobotLayoutId CurrentLayout { get; set; }
```

#### Visual Basic

```
Public CurrentLayout As IRobotLayoutId
```

#### Description

Identifier of active layout.

#### Version

Available since version 3.

### I.6.2.2 NewViewAsTab

#### C++

```
HRESULT get_NewViewAsTab(VARIANT_BOOL* );
HRESULT put_NewViewAsTab(VARIANT_BOOL);
```

#### C#

```
public bool NewViewAsTab { get; set; }
```

#### Visual Basic

```
Public NewViewAsTab As Boolean
```

#### Description

Flag indicating if a new view is created as a tab.

#### Version

Available since version 9.7.

### I.6.2.3 RecycleViews

#### C++

```
HRESULT get_RecycleViews(VARIANT_BOOL* );
HRESULT put_RecycleViews(VARIANT_BOOL);
```

#### C#

```
public bool RecycleViews { get; set; }
```

#### Visual Basic

```
Public RecycleViews As Boolean
```

### I.6.2.4 TableCount

#### C++

```
HRESULT get_TableCount(long* );
```

#### C#

```
public long TableCount { get; }
```

#### Visual Basic

```
Public ReadOnly TableCount As long
```

**Description**

Number of opened tables.

**Version**

Available since version 3.

**I.6.2.5 ViewCount****C++**

```
HRESULT get_ViewCount(long*);
```

**C#**

```
public long ViewCount { get; }
```

**Visual Basic**

```
Public ReadOnly ViewCount As long
```

**Description**

Number of opened graphical views.

**Version**

Available since version 3.

**I.6.3 IRobotViewMngr Methods**

The methods of the IRobotViewMngr class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	CreateFromSc (see page 1466)	Function creates and returns a view on the basis of a screen capture with the specified name. Type of the created view depends on the screen capture contents. If structure view manager is unable to make any of the defined view interfaces available, then, IRobotPrintable (see page 1470) interface is returned. If a screen capture with the specified name does not exist or if during its reading from the archive an error occurs, then, empty reference will be returned (zero from the level of C language, Nothing from the level of Visual Basic). Available since version 1.7.
✳	CreateRtfView (see page 1466)	The function creates a view for files in RTF format.
✳	CreateTable (see page 1466)	The function creates and returns a table of the indicated type that will present the appropriate type of data. .
✳	CreateView (see page 1466)	The function creates and returns the graphical view of the indicated type. .
✳	GetTable (see page 1467)	Function returns the interface for the table of the specified index. .
✳	GetView (see page 1467)	Function returns the interface for the graphical view of the indicated index. .
✳	Refresh (see page 1467)	The function results in refreshing all the displayed views. .
✳	ShowDialog (see page 1468)	Function displays a dialog of the given identifier. For modal dialogs the function returns a code transferred originally as a result of the dialog operation. For non-modal dialogs the function returns a True/False status informing whether the dialog displaying failed or not. .

**I.6.3.1 CreateFromSc****C++**

```
HRESULT CreateFromSc(BSTR _sc_name, IDispatch* ret);
```

**C#**

```
public IDispatch CreateFromSc(String _sc_name);
```

**Visual Basic**

```
Public Function CreateFromSc(_sc_name As String) As IDispatch
```

**Description**

Function creates and returns a view on the basis of a screen capture with the specified name. Type of the created view depends on the screen capture contents. If structure view manager is unable to make any of the defined view interfaces available, then, IRobotPrintable (see page 1470) interface is returned. If a screen capture with the specified name does not exist or if during its reading from the archive an error occurs, then, empty reference will be returned (zero from the level of C language, Nothing from the level of Visual Basic). Available since version 1.7.

**I.6.3.2 CreateRtfView****C++**

```
HRESULT CreateRtfView(IRobotRtfView** ret);
```

**C#**

```
public IRobotRtfView CreateRtfView();
```

**Visual Basic**

```
Public Function CreateRtfView() As IRobotRtfView
```

**Description**

The function creates a view for files in RTF format.

**I.6.3.3 CreateTable****C++**

```
HRESULT CreateTable(IRobotTableType _tab_type, IRobotTableDataType _data_type,
IRobotTable** ret);
```

**C#**

```
public IRobotTable CreateTable(IRobotTableType _tab_type, IRobotTableDataType _data_type);
```

**Visual Basic**

```
Public Function CreateTable(_tab_type As IRobotTableType, _data_type As
IRobotTableDataType) As IRobotTable
```

**Description**

The function creates and returns a table of the indicated type that will present the appropriate type of data. .

**I.6.3.4 CreateView****C++**

```
HRESULT CreateView(IRobotViewType _view_type, IRobotView** ret);
```

**C#**

```
public IRobotView CreateView(IRobotViewType _view_type);
```

**Visual Basic**

```
Public Function CreateView(_view_type As IRobotViewType) As IRobotView
```

**Description**

The function creates and returns the graphical view of the indicated type. .

**I.6.3.5 GetTable****C++**

```
HRESULT GetTable(long _table_idx, IRobotTableFrame** ret);
```

**C#**

```
public IRobotTableFrame GetTable(long _table_idx);
```

**Visual Basic**

```
Public Function GetTable(_table_idx As long) As IRobotTableFrame
```

**Description**

Function returns the interface for the table of the specified index. .

**Version**

Available since version 3.

**I.6.3.6 GetView****C++**

```
HRESULT GetView(long _view_idx, IRobotView2** ret);
```

**C#**

```
public IRobotView2 GetView(long _view_idx);
```

**Visual Basic**

```
Public Function GetView(_view_idx As long) As IRobotView2
```

**Description**

Function returns the interaface for the graphical view of the indicated index. .

**Version**

Available since version 3.

**I.6.3.7 Refresh****C++**

```
HRESULT Refresh();
```

**C#**

```
public void Refresh();
```

**Visual Basic**

```
Public Sub Refresh()
```

**Description**

The function results in refreshing all the displayed views. .

### I.6.3.8 ShowDialog

#### C++

```
HRESULT ShowDialog(IRobotDialogId _dlg_id, long _parent_wnd = 0, VARIANT_BOOL _modal = True, VARIANT* ret);
```

#### C#

```
public VARIANT ShowDialog(IRobotDialogId _dlg_id, long _parent_wnd = 0, bool _modal = True);
```

#### Visual Basic

```
Public Function ShowDialog(_dlg_id As IRobotDialogId, Optional _parent_wnd As long = 0, Optional _modal As Boolean = True) As VARIANT
```

#### Description

Function displays a dialog of the given identifier. For modal dialogs the function returns a code transferred originally as a result of the dialog operation. For non-modal dialogs the function returns a True/False status informing whether the dialog displaying failed or not. .

#### Version

Available since version 11.

## I.7 IRobotLayoutId

#### C++

```
enum IRobotLayoutId;
```

#### C#

```
public enum IRobotLayoutId;
```

#### Visual Basic

```
Public Enum IRobotLayoutId
```

#### Members

Members	Description
I_LI_MODEL_GEOMETRY = 1	Available since version 3.
I_LI_MODEL_NODES = 2	Available since version 3.
I_LI_MODEL_BARS = 3	Available since version 3.
I_LI_MODEL_SUPPORTS = 4	Available since version 3.
I_LI_MODEL_PROPERTIES = 5	Available since version 3.
I_LI_MODEL_LOADS = 6	Available since version 3.
I_LI_RC_COLUMN_DEFINITION = 11	Available since version 3.
I_LI_RC_BEAM_DEFINITION = 21	Available since version 3.
I_LI_RC_BEAM_WALL_DEFINITION = 31	Available since version 3.
I_LI_FOOTING_DEFINITION = 41	Available since version 3.
I_LI DESIGN CONNECTIONS = 51	Available since version 3.
I_LI DESIGN STEEL_ALUMINUM = 52	Available since version 3.
I_LI DESIGN WOOD = 53	Available since version 3.
I_LI DESIGN RC_MEMBERS = 54	Available since version 3.
I_LI TOOL SECTION DEFINITION = 61	Available since version 3.
I_LI TOOL TEXT FILE = 62	Available since version 3.
I_LI TOOL FINAL DRAWING = 63	Available since version 3.
I_LI FOUNDATION DEFINITION = 71	Available since version 3.

I_LI_RESULTS_DIAGRAMS = 81	Available since version 3.
I_LI_RESULTS_DETAILED = 82	Available since version 3.
I_LI_RESULTS_STRESS_BARS = 83	Available since version 3.
I_LI_RESULTS_STRESS_STRUCTURE = 84	Available since version 3.
I_LI_RESULTS_MAPS = 85	Available since version 3.
I_LI_RC_SLAB_REQUIRED_REINF = 91	Available since version 3.
I_LI_RC_SLAB_PROVIDED_REINF = 92	Available since version 3.
I_LI_RC_SLAB_REINFORCEMENT = 93	Available since version 3.

**Description**

Set of identifiers for the layouts defined in the Robot program. .

**Version**

Available since version 3.

## I.8 IRobotDialogId

**C++**

```
enum IRobotDialogId;
```

**C#**

```
public enum IRobotDialogId;
```

**Visual Basic**

```
Public Enum IRobotDialogId
```

**Members**

Members	Description
I_DI_BAR_SECTIONS = 1	Section dialog. Available since version 11.
I_DI_SUPPORTS = 2	Support dialog. Available since version 11.
I_DI_PANEL_THICKNESS = 3	Panel thickness dialog. Available since version 11.

**Description**

Identifiers of available dialogs.

**Version**

Available since version 11.

## II Printouts

**Enumerations**

	Name	Description
	IRobotVariablePredefinedId ( <a href="#">see page 1483</a> )	Set of identifiers for standard (previously prepared for the user) variables. . .
	IRobotOutputFileFormat ( <a href="#">see page 1495</a> )	Available file formats to which a printout may be directed. .
	IRobotPageSetupOrientation ( <a href="#">see page 1507</a> )	Page setup.

	IRobotPageSetupFrameType (see page 1507)	Types of frames used for decoration of individual page elements. .
	IRobotPageSetupTocLocation (see page 1509)	
	IRobotReportItemType (see page 1510)	Available types of printout components.
	IRobotExternalPreviewFormat (see page 1510)	External format of a print preview.

## Interfaces

	Name	Description
	IRobotPrintable (see page 1470)	Interface of any data object that is to be printed by the printing module of Robot. .
	IRobotPrintEngine (see page 1473)	Interface representing the Robot printing module. .
	IRobotVariableMngr (see page 1480)	Manager of variables (parameters) used in printout templates.
	IRobotPageSetup (see page 1484)	Interface allows reading and modifying current page settings and enables template management. The Robot program allows definition of many templates storing page settings, which are differentiated by name. At the given moment one of the templates is set as current (default). .
	IRobotReportTemplateMngr (see page 1495)	There is a possibility to save a previously composed printout as a template, which may be used many times afterwards in the same or different project. Printout templates are recognized by names. All templates available in the current project are managed by the template manager. .
	IRobotReportStdElementRtf (see page 1497)	Standard (predefined) printout element based on an RTF format template file.
	IRobotReportItemList (see page 1500)	List of printout components.
	IRobotReportItem (see page 1501)	Interface that describes a printout element.
	IRobotPageSetupTableOfContents (see page 1508)	Interface that describes the table of contents.
	IRobotVariableMngrExtension (see page 1511)	Interface of external extension for the manager of variables (parameters) used in printouts.

## II.1 IRobotPrintable

### Class Hierarchy

#### C++

```
interface IRobotPrintable : IDispatch;
```

#### C#

```
public interface IRobotPrintable;
```

### Visual Basic

```
Public Interface IRobotPrintable
```

### Description

Interface of any data object that is to be printed by the printing module of Robot. .

## II.1.1 IRobotPrintable Members

The following tables list the members exposed by IRobotPrintable.

## Public Fields

	Name	Description
◆	Comment ( [ see page 1471) )	Text of the comment to be added to the object on the printout.
◆	IncludeDateTime ( [ see page 1471) )	The flag indicating if the current date and time is added to the object on the printout.
◆	StartFromNewPage ( [ see page 1472) )	Falg indicating whether an object should be printed from a new page .
◆	Title ( [ see page 1472) )	Title.

## Public Methods

	Name	Description
◆	SaveToFile ( [ see page 1472) )	Function exports object contents to the indicated file with the specified format. Available since version 1.7.

## II.1.2 IRobotPrintable Fields

The fields of the IRobotPrintable class are listed here.

### Public Fields

	Name	Description
◆	Comment ( [ see page 1471) )	Text of the comment to be added to the object on the printout.
◆	IncludeDateTime ( [ see page 1471) )	The flag indicating if the current date and time is added to the object on the printout.
◆	StartFromNewPage ( [ see page 1472) )	Falg indicating whether an object should be printed from a new page .
◆	Title ( [ see page 1472) )	Title.

### II.1.2.1 Comment

#### C++

```
HRESULT get_Comment(BSTR* );
HRESULT put_Comment(BSTR);
```

#### C#

```
public String Comment { get; set; }
```

#### Visual Basic

```
Public Comment As String
```

#### Description

Text of the comment to be added to the object on the printout.

#### Version

Available since version 12.

## II.1.2.2 IncludeDateTime

#### C++

```
HRESULT get_IncludeDateTime(VARIANT_BOOL* );
HRESULT put_IncludeDateTime(VARIANT_BOOL);
```

#### C#

```
public bool IncludeDateTime { get; set; }
```

#### Visual Basic

```
Public IncludeDateTime As Boolean
```

**Description**

The flag indicating if the current date and time is added to the object on the printout.

**Version**

Available since version 12.

**II.1.2.3 StartFromNewPage****C#**

```
void StartFromNewPage;
```

**Description**

Falg indicating whether an object should be printed from a new page .

**II.1.2.4 Title****C++**

```
HRESULT get_Title(BSTR* );
HRESULT put_Title(BSTR);
```

**C#**

```
public String Title { get; set; }
```

**Visual Basic**

```
Public Title As String
```

**Description**

Title.

**II.1.3 IRobotPrintable Methods**

The methods of the IRobotPrintable class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	SaveToFile (see page 1472)	Function exports object contents to the indicated file with the specified format. Available since version 1.7.

**II.1.3.1 SaveToFile****C++**

```
HRESULT SaveToFile(BSTR _file_path, IRobotOutputFormat _file_format, VARIANT_BOOL* ret);
```

**C#**

```
public bool SaveToFile(String _file_path, IRobotOutputFormat _file_format);
```

**Visual Basic**

```
Public Function SaveToFile(_file_path As String, _file_format As IRobotOutputFormat) As Boolean
```

**Description**

Function exports object contents to the indicated file with the specified format. Available since version 1.7.

**II.2 IRobotPrintEngine****Class Hierarchy**

**C++**

```
interface IRobotPrintEngine : IDispatch;
```

**C#**

```
public interface IRobotPrintEngine;
```

**Visual Basic**

```
Public Interface IRobotPrintEngine
```

**Description**

Interface representing the Robot printing module. .

**II.2.1 IRobotPrintEngine Members**

The following tables list the members exposed by IRobotPrintEngine.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	IsWhilePreview ( <a href="#">see page 1474</a> )	Flag indicating that the application is currently working in the "print preview" mode.
❖	OrganizerItems ( <a href="#">see page 1474</a> )	List of printout components defined in the Printout Composition dialog box.
❖	PageSetup ( <a href="#">see page 1475</a> )	Page setup.
❖	ReportTemplates ( <a href="#">see page 1475</a> )	List of available printout templates Available since version 1.7.
❖	ScreenCaptures ( <a href="#">see page 1475</a> )	List of available screen captures Available since version 1.7.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	AddScToReport ( <a href="#">see page 1476</a> )	Function adds screen capture with a specified name at the end of the current printout. If a screen capture with a specified name is not found, zero value is returned (False). Available since version 1.7.
❖	AddTemplateToReport ( <a href="#">see page 1476</a> )	Function adds contents of a printout template with a specified name at the end of the printout. If a template with this name is not found, then function returns zero value (False). Available since version 1.7.
❖	AddToReport ( <a href="#">see page 1477</a> )	The function adds the indicated object at the end of the currently composed printout. .
❖	ClosePreview ( <a href="#">see page 1477</a> )	Function forces closing a print preview.
❖	CreateReportFromOrganizer ( <a href="#">see page 1477</a> )	Function creates a printout composed of elements currently defined in the Printout Composition dialog box.
❖	ExternalPreviewReport ( <a href="#">see page 1478</a> )	Function launches an external program and exports the current printout to it.
❖	PreviewReport ( <a href="#">see page 1478</a> )	The function runs the print preview of the current printout composition .
❖	PrintReport ( <a href="#">see page 1478</a> )	The function results in sending the current printout composition to the printer. .
❖	RemoveFromReport ( <a href="#">see page 1478</a> )	The function deletes the indicated object from the current printout. .
❖	RemoveScFromReport ( <a href="#">see page 1479</a> )	Function deletes screen capture with a specified name from the current printout. First screen capture with this name occurring on the printout component list is deleted. Available since version 1.7.
❖	ResetReport ( <a href="#">see page 1479</a> )	The function deletes all objects from the current printout. .
❖	SaveReportToFile ( <a href="#">see page 1479</a> )	Function exports printout to the specified file with the defined format. Available since version 1.7.
❖	SaveReportToOrganizer ( <a href="#">see page 1479</a> )	Function saves the current printout as a current list of printout elements presented in the Printout Composition dialog box.

	SaveReportToTemplate (see page 1480)	Function saves printout to the template with a specified name. If a template with this name already exists its contents will be replaced. Available since version 1.7.
-----------------------------------------------------------------------------------	--------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## II.2.2 IRobotPrintEngine Fields

The fields of the IRobotPrintEngine class are listed here.

### Public Fields

	Name	Description
◆	IsWhilePreview (see page 1474)	Flag indicating that the application is currently working in the "print preview" mode.
◆	OrganizerItems (see page 1474)	List of printout components defined in the Printout Composition dialog box.
◆	PageSetup (see page 1475)	Page setup.
◆	ReportTemplates (see page 1475)	List of available printout templates Available since version 1.7.
◆	ScreenCaptures (see page 1475)	List of available screen captures Available since version 1.7.

### II.2.2.1 IsWhilePreview

#### C++

```
HRESULT get_IsWhilePreview(VARIANT_BOOL*);
```

#### C#

```
public bool IsWhilePreview { get; }
```

#### Visual Basic

```
Public ReadOnly IsWhilePreview As Boolean
```

#### Description

Flag indicating that the application is currently working in the "print preview" mode.

### II.2.2.2 OrganizerItems

#### C++

```
HRESULT get_OrganizerItems(IRobotReportItemList**);
```

#### C#

```
public IRobotReportItemList OrganizerItems { get; }
```

#### Visual Basic

```
Public ReadOnly OrganizerItems As IRobotReportItemList
```

#### Description

List of printout components defined in the Printout Composition dialog box.

#### Version

Available since version 5.5.

### II.2.2.3 PageSetup

#### C++

```
HRESULT get_PageSetup(IRobotPageSetup**);
```

#### C#

```
public IRobotPageSetup PageSetup { get; }
```

**Visual Basic**

```
Public ReadOnly PageSetup As IRobotPageSetup
```

**Description**

Page setup.

**II.2.2.4 ReportTemplates****C++**

```
HRESULT get_ReportTemplates(IRobotReportTemplateMngr**);
```

**C#**

```
public IRobotReportTemplateMngr ReportTemplates { get; }
```

**Visual Basic**

```
Public ReadOnly ReportTemplates As IRobotReportTemplateMngr
```

**Description**

List of available printout templates Available since version 1.7.

**II.2.2.5 ScreenCaptures****C++**

```
HRESULT get_ScreenCaptures(IRobotScreenCaptureMngr**);
```

**C#**

```
public IRobotScreenCaptureMngr ScreenCaptures { get; }
```

**Visual Basic**

```
Public ReadOnly ScreenCaptures As IRobotScreenCaptureMngr
```

**Description**

List of available screen captures Available since version 1.7.

**II.2.3 IRobotPrintEngine Methods**

The methods of the IRobotPrintEngine class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	AddScToReport (see page 1476)	Function adds screen capture with a specified name at the end of the current printout. If a screen capture with a specified name is not found, zero value is returned (False). Available since version 1.7.
✳	AddTemplateToReport (see page 1476)	Function adds contents of a printout template with a specified name at the end of the printout. If a template with this name is not found, then function returns zero value (False). Available since version 1.7.
✳	AddToReport (see page 1477)	The function adds the indicated object at the end of the currently composed printout. .
✳	ClosePreview (see page 1477)	Function forces closing a print preview.
✳	CreateReportFromOrganizer (see page 1477)	Function creates a printout composed of elements currently defined in the Printout Composition dialog box.
✳	ExternalPreviewReport (see page 1478)	Function launches an external program and exports the current printout to it.
✳	PreviewReport (see page 1478)	The function runs the print preview of the current printout composition .
✳	PrintReport (see page 1478)	The function results in sending the current printout composition to the printer. .

	RemoveFromReport ( <a href="#">see page 1478</a> )	The function deletes the indicated object from the current printout. .
	RemoveScFromReport ( <a href="#">see page 1479</a> )	Function deletes screen capture with a specified name from the current printout. First screen capture with this name occurring on the printout component list is deleted. Available since version 1.7.
	ResetReport ( <a href="#">see page 1479</a> )	The function deletes all objects from the current printout. .
	SaveReportToFile ( <a href="#">see page 1479</a> )	Function exports printout to the specified file with the defined format. Available since version 1.7.
	SaveReportToOrganizer ( <a href="#">see page 1479</a> )	Function saves the current printout as a current list of printout elements presented in the Printout Composition dialog box.
	SaveReportToTemplate ( <a href="#">see page 1480</a> )	Function saves printout to the template with a specified name. If a template with this name already exists its contents will be replaced. Available since version 1.7.

### II.2.3.1 AddScToReport

#### C++

```
HRESULT AddScToReport(BSTR _sc_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool AddScToReport(String _sc_name);
```

#### Visual Basic

```
Public Function AddScToReport(_sc_name As String) As Boolean
```

#### Description

Function adds screen capture with a specified name at the end of the current printout. If a screen capture with a specified name is not found, zero value is returned (False). Available since version 1.7.

### II.2.3.2 AddTemplateToReport

#### C++

```
HRESULT AddTemplateToReport(BSTR _tmpl_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool AddTemplateToReport(String _tmpl_name);
```

#### Visual Basic

```
Public Function AddTemplateToReport(_tmpl_name As String) As Boolean
```

#### Description

Function adds contents of a printout template with a specified name at the end of the printout. If a template with this name is not found, then function returns zero value (False). Available since version 1.7.

### II.2.3.3 AddToReport

#### C++

```
HRESULT AddToReport(IRobotPrintable* _obj);
```

#### C#

```
public void AddToReport(IRobotPrintable _obj);
```

#### Visual Basic

```
Public Sub AddToReport(ByRef _obj As IRobotPrintable)
```

**Description**

The function adds the indicated object at the end of the currently composed printout. .

**II.2.3.4 ClosePreview****C++**

```
HRESULT ClosePreview();
```

**C#**

```
public void ClosePreview();
```

**Visual Basic**

```
Public Sub ClosePreview()
```

**Description**

Function forces closing a print preview.

**II.2.3.5 CreateReportFromOrganizer****C++**

```
HRESULT CreateReportFromOrganizer();
```

**C#**

```
public void CreateReportFromOrganizer();
```

**Visual Basic**

```
Public Sub CreateReportFromOrganizer()
```

**Description**

Function creates a printout composed of elements currently defined in the Printout Composition dialog box.

**Version**

Available since version 6.4.

**II.2.3.6 ExternalPreviewReport****C++**

```
HRESULT ExternalPreviewReport(IRobotExternalPreviewFormat _format);
```

**C#**

```
public void ExternalPreviewReport(IRobotExternalPreviewFormat _format);
```

**Visual Basic**

```
Public Sub ExternalPreviewReport(_format As IRobotExternalPreviewFormat)
```

**Description**

Function launches an external program and exports the current printout to it.

**Version**

Available since version 6.4.

### II.2.3.7 PreviewReport

C++

```
HRESULT PreviewReport();
```

C#

```
public void PreviewReport();
```

Visual Basic

```
Public Sub PreviewReport()
```

Description

The function runs the print preview of the current printout composition .

### II.2.3.8 PrintReport

C++

```
HRESULT PrintReport();
```

C#

```
public void PrintReport();
```

Visual Basic

```
Public Sub PrintReport()
```

Description

The function results in sending the current printout composition to the printer. .

### II.2.3.9 RemoveFromReport

C++

```
HRESULT RemoveFromReport(IRobotPrintable* _obj);
```

C#

```
public void RemoveFromReport(IRobotPrintable _obj);
```

Visual Basic

```
Public Sub RemoveFromReport(ByRef _obj As IRobotPrintable)
```

Description

The function deletes the indicated object from the current printout. .

### II.2.3.10 RemoveScFromReport

C++

```
HRESULT RemoveScFromReport(BSTR _sc_name);
```

C#

```
public void RemoveScFromReport(String _sc_name);
```

Visual Basic

```
Public Sub RemoveScFromReport(_sc_name As String)
```

## Description

Function deletes screen capture with a specified name from the current printout. First screen capture with this name occurring on the printout component list is deleted. Available since version 1.7.

### II.2.3.11 ResetReport

#### C++

```
HRESULT ResetReport();
```

#### C#

```
public void ResetReport();
```

#### Visual Basic

```
Public Sub ResetReport()
```

## Description

The function deletes all objects from the current printout. .

### II.2.3.12 SaveReportToFile

#### C++

```
HRESULT SaveReportToFile(BSTR _file_path, IRobotOutputFormat _file_format,
VARIANT_BOOL* ret);
```

#### C#

```
public bool SaveReportToFile(String _file_path, IRobotOutputFormat _file_format);
```

#### Visual Basic

```
Public Function SaveReportToFile(_file_path As String, _file_format As
IRobotOutputFormat) As Boolean
```

## Description

Function exports printout to the specified file with the defined format. Available since version 1.7.

### II.2.3.13 SaveReportToOrganizer

#### C++

```
HRESULT SaveReportToOrganizer();
```

#### C#

```
public void SaveReportToOrganizer();
```

#### Visual Basic

```
Public Sub SaveReportToOrganizer()
```

## Description

Function saves the current printout as a current list of printout elements presented in the Printout Composition dialog box.

## Version

Available since version 6.4.

### II.2.3.14 SaveReportToTemplate

**C++**

```
HRESULT SaveReportToTemplate(BSTR _tmpl_name);
```

**C#**

```
public void SaveReportToTemplate(String _tmpl_name);
```

**Visual Basic**

```
Public Sub SaveReportToTemplate(_tmpl_name As String)
```

**Description**

Function saves printout to the template with a specified name. If a template with this name already exists its contents will be replaced. Available since version 1.7.

## II.3 IRobotVariableMngr

**Class Hierarchy**

**C++**

```
interface IRobotVariableMngr : IDispatch;
```

**C#**

```
public interface IRobotVariableMngr;
```

**Visual Basic**

```
Public Interface IRobotVariableMngr
```

**Description**

Manager of variables (parameters) used in printout templates.

### II.3.1 IRobotVariableMngr Members

The following tables list the members exposed by IRobotVariableMngr.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	AddExtension (🔗 see page 1481)	Function enables extending the manager with an external object providing values for variables.
💡	Delete (🔗 see page 1481)	The function deletes a variable with the given name. .
💡	Exist (🔗 see page 1482)	The function checks if there exists a variable with the indicated name .
💡	GetPredefinedValue (🔗 see page 1482)	Function returns the value of the given standard variable.
💡	GetValue (🔗 see page 1482)	The function returns the value of the indicated variable. If the variable with the indicated name does not exist, the function returns an empty string.
💡	RemoveExtension (🔗 see page 1483)	Function deletes a specified object from the extension list.
💡	SetPredefinedValue (🔗 see page 1483)	Function sets value of a specified standard variable. Some variables are read-only. For such a variable, values cannot be set. Then, this function returns zero (False). .
💡	SetValue (🔗 see page 1483)	The function ascribes a defined value to the variable with the given name. If the variable does not exist, it will be defined. If it does exist, its value will be changed. .

## II.3.2 IRobotVariableMngr Methods

The methods of the IRobotVariableMngr class are listed here.

### Public Methods

	Name	Description
⊕	AddExtension ( [ see page 1481) )	Function enables extending the manager with an external object providing values for variables.
⊕	Delete ( [ see page 1481) )	The function deletes a variable with the given name. .
⊕	Exist ( [ see page 1482) )	The function checks if there exists a variable with the indicated name .
⊕	GetPredefinedValue ( [ see page 1482) )	Function returns the value of the given standard variable.
⊕	GetValue ( [ see page 1482) )	The function returns the value of the indicated variable. If the variable with the indicated name does not exist, the function returns an empty string.
⊕	RemoveExtension ( [ see page 1483) )	Function deletes a specified object from the extension list.
⊕	SetPredefinedValue ( [ see page 1483) )	Function sets value of a specified standard variable. Some variables are read-only. For such a variable, values cannot be set. Then, this function returns zero (False). .
⊕	SetValue ( [ see page 1483) )	The function ascribes a defined value to the variable with the given name. If the variable does not exist, it will be defined. If it does exist, its value will be changed. .

### II.3.2.1 AddExtension

#### C++

```
HRESULT AddExtension(IRobotVariableMngrExtension* _ext_obj);
```

#### C#

```
public void AddExtension(IRobotVariableMngrExtension _ext_obj);
```

#### Visual Basic

```
Public Sub AddExtension(ByRef _ext_obj As IRobotVariableMngrExtension)
```

#### Description

Function enables extending the manager with an external object providing values for variables.

### II.3.2.2 Delete

#### C++

```
HRESULT Delete(BSTR _var_name);
```

#### C#

```
public void Delete(String _var_name);
```

#### Visual Basic

```
Public Sub Delete(_var_name As String)
```

#### Description

The function deletes a variable with the given name. .

### II.3.2.3 Exist

**C++**

```
HRESULT Exist(BSTR _var_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist(String _var_name);
```

**Visual Basic**

```
Public Function Exist(_var_name As String) As Boolean
```

**Description**

The function checks if there exists a variable with the indicated name .

### II.3.2.4 GetPredefinedValue

**C++**

```
HRESULT GetPredefinedValue(IRobotVariablePredefinedId _var_id, BSTR* ret);
```

**C#**

```
public String GetPredefinedValue(IRobotVariablePredefinedId _var_id);
```

**Visual Basic**

```
Public Function GetPredefinedValue(_var_id As IRobotVariablePredefinedId) As String
```

**Description**

Function returns the value of the given standard variable.

### II.3.2.5 GetValue

**C++**

```
HRESULT GetValue(BSTR _var_name, BSTR* ret);
```

**C#**

```
public String GetValue(String _var_name);
```

**Visual Basic**

```
Public Function GetValue(_var_name As String) As String
```

**Description**

The function returns the value of the indicated variable. If the variable with the indicated name does not exist, the function returns an empty string.

### II.3.2.6 RemoveExtension

**C++**

```
HRESULT RemoveExtension(IRobotVariableMngrExtension* _ext_obj);
```

**C#**

```
public void RemoveExtension(IRobotVariableMngrExtension _ext_obj);
```

**Visual Basic**

```
Public Sub RemoveExtension(ByRef _ext_obj As IRobotVariableMngrExtension)
```

## Description

Function deletes a specified object from the extension list.

### II.3.2.7 SetPredefinedValue

#### C++

```
HRESULT SetPredefinedValue(IRobotVariablePredefinedId _var_id, BSTR _var_value,
VARIANT_BOOL* ret);
```

#### C#

```
public bool SetPredefinedValue(IRobotVariablePredefinedId _var_id, String _var_value);
```

#### Visual Basic

```
Public Function SetPredefinedValue(_var_id As IRobotVariablePredefinedId, _var_value As
String) As Boolean
```

## Description

Function sets value of a specified standard variable. Some variables are read-only. For such a variable, values cannot be set. Then, this function returns zero (False). .

### II.3.2.8 SetValue

#### C++

```
HRESULT SetValue(BSTR _var_name, BSTR _var_value);
```

#### C#

```
public void SetValue(String _var_name, String _var_value);
```

#### Visual Basic

```
Public Sub SetValue(_var_name As String, _var_value As String)
```

## Description

The function ascribes a defined value to the variable with the given name. If the variable does not exist, it will be defined. If it does exist, its value will be changed. .

## II.4 IRobotVariablePredefinedId

#### C++

```
enum IRobotVariablePredefinedId;
```

#### C#

```
public enum IRobotVariablePredefinedId;
```

#### Visual Basic

```
Public Enum IRobotVariablePredefinedId
```

## Members

Members	Description
I_VPI_REPORT_PAGE_NUMBER = 0	Number of a print-out page (initial or current during printing).
I_VPI_DATE = 1	Current date.
I_VPI_TIME = 2	Current time.
I_VPI_ROBOT_APP = 3	Program name .

I_VPI_ROBOT_VER = 4	Program version.
I_VPI_ROBOT_PROVIDER = 5	Name of the program supplier (dealer).
I_VPI_USER_NAME = 6	User's name .
I_VPI_USER_ADDRESS = 7	User's address.
I_VPI_PROJECT_NAME = 8	Project name .
I_VPI_PROJECT_FILE = 9	Name of the file containing the project.
I_VPI_PROJECT_DIRECTORY = 10	Folder where the file with the project is stored.
I_VPI_PROJECT_SIZE = 11	Size of the file with the project.
I_VPI_PROJECT_CREATED = 12	Date when the project file was created.
I_VPI_PROJECT_MODIFIED = 13	Date of the last modification of the project file.
I_VPI_PROJECT_VERSION = 14	Project version.

#### Description

Set of identifiers for standard (previously prepared for the user) variables. .

## II.5 IRobotPageSetup

#### Class Hierarchy

#### C++

```
interface IRobotPageSetup : IDispatch;
```

#### C#

```
public interface IRobotPageSetup;
```

#### Visual Basic

```
Public Interface IRobotPageSetup
```

#### Description

Interface allows reading and modifying current page settings and enables template management. The Robot program allows definition of many templates storing page settings, which are differentiated by name. At the given moment one of the templates is set as current (default). .

### II.5.1 IRobotPageSetup Members

The following tables list the members exposed by IRobotPageSetup.

#### Public Fields

	Name	Description
◆	Footer ( [ see page 1486)	Footer on each printout page .
◆	FromEdgeFooter ( [ see page 1487)	Distance of footer from the bottom edge of paper in millimeters.
◆	FromEdgeHeader ( [ see page 1487)	Distance of header from the top edge of paper in millimeters.
◆	Gutter ( [ see page 1487)	Gutter in millimeters.
◆	Header ( [ see page 1488)	Header on each printout page.
◆	IsCurrent ( [ see page 1488)	Flag that enables setting page parameters as the current ones .
◆	MarginBottom ( [ see page 1488)	Bottom margin in millimeters.
◆	MarginLeft ( [ see page 1488)	Left margin in millimeters.
◆	MarginRight ( [ see page 1489)	Right margin in millimeters.
◆	MarginTop ( [ see page 1489)	Top margin in millimeters.
◆	PageOrientation ( [ see page 1489)	Page setup.
◆	PaperHeight ( [ see page 1490)	Paper height in millimeters.

❖	PaperSize ( <a href="#">see page 1490</a> )	Identifier of paper size.
❖	PaperWidth ( <a href="#">see page 1490</a> )	Paper width in millimeters.
❖	StartPageNumber ( <a href="#">see page 1491</a> )	Initial page number Attention: Initial page number is not stored in a template. Setting a value of the initial page number means the same as setting a value of the variable named PAGE_NUMBER by means of the IRobotVariableMngr ( <a href="#">see page 1480</a> ) interface.
❖	TemplateCount ( <a href="#">see page 1491</a> )	Number of the remembered templates with page settings .
❖	TemplateName ( <a href="#">see page 1491</a> )	Template name.
❖	TextFrame ( <a href="#">see page 1492</a> )	Page contents frame.
❖	TitlePage ( <a href="#">see page 1492</a> )	Printout title page .
❖	Toc ( <a href="#">see page 1492</a> )	Parameters of the table of contents.
❖	Variables ( <a href="#">see page 1493</a> )	Manager of variables (parameters).

#### Public Methods

	Name	Description
❖	GetTemplateName ( <a href="#">see page 1493</a> )	Function returns the name of a template with the specified index. Templates are indexed with values from 1 to TemplateCount ( <a href="#">see page 1491</a> ). .
❖	Load ( <a href="#">see page 1493</a> )	Function reads page settings from the specified template. .
❖	LoadCurrent ( <a href="#">see page 1494</a> )	Function reads current page settings. .
❖	Save ( <a href="#">see page 1494</a> )	Function saves current page settings. .
❖	SaveAs ( <a href="#">see page 1494</a> )	Function saves current page settings as a template with the specified name. .

#### II.5.2 IRobotPageSetup Fields

The fields of the IRobotPageSetup class are listed here.

#### Public Fields

	Name	Description
❖	Footer ( <a href="#">see page 1486</a> )	Footer on each printout page .
❖	FromEdgeFooter ( <a href="#">see page 1487</a> )	Distance of footer from the bottom edge of paper in millimeters.
❖	FromEdgeHeader ( <a href="#">see page 1487</a> )	Distance of header from the top edge of paper in millimeters.
❖	Gutter ( <a href="#">see page 1487</a> )	Gutter in millimeters.
❖	Header ( <a href="#">see page 1488</a> )	Header on each printout page.
❖	IsCurrent ( <a href="#">see page 1488</a> )	Flag that enables setting page parameters as the current ones .
❖	MarginBottom ( <a href="#">see page 1488</a> )	Bottom margin in millimeters.
❖	MarginLeft ( <a href="#">see page 1488</a> )	Left margin in millimeters.
❖	MarginRight ( <a href="#">see page 1489</a> )	Right margin in millimeters.
❖	MarginTop ( <a href="#">see page 1489</a> )	Top margin in millimeters.
❖	PageOrientation ( <a href="#">see page 1489</a> )	Page setup.
❖	PaperHeight ( <a href="#">see page 1490</a> )	Paper height in millimeters.
❖	PaperSize ( <a href="#">see page 1490</a> )	Identifier of paper size.
❖	PaperWidth ( <a href="#">see page 1490</a> )	Paper width in millimeters.
❖	StartPageNumber ( <a href="#">see page 1491</a> )	Initial page number Attention: Initial page number is not stored in a template. Setting a value of the initial page number means the same as setting a value of the variable named PAGE_NUMBER by means of the IRobotVariableMngr ( <a href="#">see page 1480</a> ) interface.
❖	TemplateCount ( <a href="#">see page 1491</a> )	Number of the remembered templates with page settings .
❖	TemplateName ( <a href="#">see page 1491</a> )	Template name.
❖	TextFrame ( <a href="#">see page 1492</a> )	Page contents frame.
❖	TitlePage ( <a href="#">see page 1492</a> )	Printout title page .

	Toc (see page 1492)	Parameters of the table of contents.
	Variables (see page 1493)	Manager of variables (parameters).

### II.5.2.1 Footer

#### C++

```
HRESULT get_Footer(IRobotReportStdElementRtf**);
```

#### C#

```
public IRobotReportStdElementRtf Footer { get; }
```

#### Visual Basic

```
Public ReadOnly Footer As IRobotReportStdElementRtf
```

#### Description

Footer on each printout page .

#### Version

Available since version 2.5.

### II.5.2.2 FromEdgeFooter

#### C++

```
HRESULT get_FromEdgeFooter(double*);  
HRESULT put_FromEdgeFooter(double);
```

#### C#

```
public double FromEdgeFooter { get; set; }
```

#### Visual Basic

```
Public FromEdgeFooter As Double
```

#### Description

Distance of footer from the bottom edge of paper in millimeters.

#### Version

Available since version 5.5.

### II.5.2.3 FromEdgeHeader

#### C++

```
HRESULT get_FromEdgeHeader(double*);  
HRESULT put_FromEdgeHeader(double);
```

#### C#

```
public double FromEdgeHeader { get; set; }
```

#### Visual Basic

```
Public FromEdgeHeader As Double
```

#### Description

Distance of header from the top edge of paper in millimeters.

#### Version

Available since version 5.5.

#### II.5.2.4 Gutter

##### C++

```
HRESULT get_Gutter(double*);  
HRESULT put_Gutter(double);
```

##### C#

```
public double Gutter { get; set; }
```

##### Visual Basic

```
Public Gutter As Double
```

##### Description

Gutter in millimeters.

##### Version

Available since version 5.5.

#### II.5.2.5 Header

##### C++

```
HRESULT get_Header(IRobotReportStdElementRtf**);
```

##### C#

```
public IRobotReportStdElementRtf Header { get; }
```

##### Visual Basic

```
Public ReadOnly Header As IRobotReportStdElementRtf
```

##### Description

Header on each printout page.

##### Version

Available since version 2.5.

#### II.5.2.6 IsCurrent

##### C++

```
HRESULT get_IsCurrent(VARIANT_BOOL*);  
HRESULT put_IsCurrent(VARIANT_BOOL);
```

##### C#

```
public bool IsCurrent { get; set; }
```

##### Visual Basic

```
Public IsCurrent As Boolean
```

##### Description

Flag that enables setting page parameters as the current ones .

##### Version

Available since version 5.5.

#### II.5.2.7 MarginBottom

##### C++

```
HRESULT get_MarginBottom(double*);
```

```
HRESULT put_MarginBottom(double);  
  
C#  
public double MarginBottom { get; set; }
```

**Visual Basic**

```
Public MarginBottom As Double
```

**Description**

Bottom margin in millimeters.

**Version**

Available since version 5.5.

**II.5.2.8 MarginLeft****C++**

```
HRESULT get_MarginLeft(double*);  
HRESULT put_MarginLeft(double);
```

**C#**

```
public double MarginLeft { get; set; }
```

**Visual Basic**

```
Public MarginLeft As Double
```

**Description**

Left margin in millimeters.

**Version**

Available since version 5.5.

**II.5.2.9 MarginRight****C++**

```
HRESULT get_MarginRight(double*);  
HRESULT put_MarginRight(double);
```

**C#**

```
public double MarginRight { get; set; }
```

**Visual Basic**

```
Public MarginRight As Double
```

**Description**

Right margin in millimeters.

**Version**

Available since version 5.5.

**II.5.2.10 MarginTop****C++**

```
HRESULT get_MarginTop(double*);  
HRESULT put_MarginTop(double);
```

**C#**

```
public double MarginTop { get; set; }
```

**Visual Basic**

```
Public MarginTop As double
```

**Description**

Top margin in millimeters.

**Version**

Available since version 5.5.

**II.5.2.11 PageOrientation****C++**

```
HRESULT get_PageOrientation(IRobotPageSetupOrientation* );
HRESULT put_PageOrientation(IRobotPageSetupOrientation);
```

**C#**

```
public IRobotPageSetupOrientation PageOrientation { get; set; }
```

**Visual Basic**

```
Public PageOrientation As IRobotPageSetupOrientation
```

**Description**

Page setup.

**Version**

Available since version 5.5.

**II.5.2.12 PaperHeight****C++**

```
HRESULT get_PaperHeight(long* );
HRESULT put_PaperHeight(long);
```

**C#**

```
public long PaperHeight { get; set; }
```

**Visual Basic**

```
Public PaperHeight As long
```

**Description**

Paper height in millimeters.

**Version**

Available since version 5.5.

**II.5.2.13 PaperSize****C++**

```
HRESULT get_PaperSize(long* );
HRESULT put_PaperSize(long);
```

**C#**

```
public long PaperSize { get; set; }
```

**Visual Basic**

```
Public PaperSize As long
```

**Description**

Identifier of paper size.

**Version**

Available since version 5.5.

**II.5.2.14 PaperWidth****C++**

```
HRESULT get_PaperWidth(long*);  
HRESULT put_PaperWidth(long);
```

**C#**

```
public long PaperWidth { get; set; }
```

**Visual Basic**

```
Public PaperWidth As long
```

**Description**

Paper width in millimeters.

**Version**

Available since version 5.5.

**II.5.2.15 StartPageNumber****C++**

```
HRESULT get_StartPageNumber(long*);  
HRESULT put_StartPageNumber(long);
```

**C#**

```
public long StartPageNumber { get; set; }
```

**Visual Basic**

```
Public StartPageNumber As long
```

**Description**

Initial page number Attention: Initial page number is not stored in a template. Setting a value of the initial page number means the same as setting a value of the variable named PAGE\_NUMBER by means of the IRobotVariableMngr ( see page 1480) interface.

**Version**

Available since version 5.5.

**II.5.2.16 TemplateCount****C++**

```
HRESULT get_TemplateCount(long*);
```

**C#**

```
public long TemplateCount { get; }
```

**Visual Basic**

```
Public ReadOnly TemplateCount As long
```

**Description**

Number of the remembered templates with page settings .

**Version**

Available since version 5.5.

**II.5.2.17 TemplateName****C++**

```
HRESULT get_TemplateName(BSTR*);
```

**C#**

```
public String TemplateName { get; }
```

**Visual Basic**

```
Public ReadOnly TemplateName As String
```

**Description**

Template name.

**Version**

Available since version 5.5.

**II.5.2.18 TextFrame****C++**

```
HRESULT get_TextFrame(IRobotPageSetupFrameType*);  
HRESULT put_TextFrame(IRobotPageSetupFrameType);
```

**C#**

```
public IRobotPageSetupFrameType TextFrame { get; set; }
```

**Visual Basic**

```
Public TextFrame As IRobotPageSetupFrameType
```

**Description**

Page contents frame.

**Version**

Available since version 5.5.

**II.5.2.19 TitlePage****C++**

```
HRESULT get_TitlePage(IRobotReportStdElementRtf**);
```

**C#**

```
public IRobotReportStdElementRtf TitlePage { get; }
```

**Visual Basic**

```
Public ReadOnly TitlePage As IRobotReportStdElementRtf
```

**Description**

Printout title page .

**Version**

Available since version 2.5.

## II.5.2.20 Toc

### C++

```
HRESULT get_Toc(IRobotPageSetupTableOfContents**);
```

### C#

```
public IRobotPageSetupTableOfContents Toc { get; }
```

### Visual Basic

```
Public ReadOnly Toc As IRobotPageSetupTableOfContents
```

### Description

Parameters of the table of contents.

### Version

Available since version 5.5.

## II.5.2.21 Variables

### C++

```
HRESULT get_Variables(IRobotVariableMngr**);
```

### C#

```
public IRobotVariableMngr Variables { get; }
```

### Visual Basic

```
Public ReadOnly Variables As IRobotVariableMngr
```

### Description

Manager of variables (parameters).

## II.5.3 IRobotPageSetup Methods

The methods of the IRobotPageSetup class are listed here.

### Public Methods

	Name	Description
✳️	GetTemplateName (✉ see page 1493)	Function returns the name of a template with the specified index. Templates are indexed with values from 1 to TemplateCount (✉ see page 1491). .
✳️	Load (✉ see page 1493)	Function reads page settings from the specified template. .
✳️	LoadCurrent (✉ see page 1494)	Function reads current page settings. .
✳️	Save (✉ see page 1494)	Function saves current page settings. .
✳️	SaveAs (✉ see page 1494)	Function saves current page settings as a template with the specified name. .

### II.5.3.1 GetTemplateName

#### C++

```
HRESULT GetTemplateName(long _idx, BSTR* ret);
```

#### C#

```
public String GetTemplateName(long _idx);
```

**Visual Basic**

```
Public Function GetTemplateName(_idx As long) As String
```

**Description**

Function returns the name of a template with the specified index. Templates are indexed with values from 1 to TemplateCount (see page 1491). .

**Version**

Available since version 5.5.

**II.5.3.2 Load****C++**

```
HRESULT Load(BSTR _template_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool Load(String _template_name);
```

**Visual Basic**

```
Public Function Load(_template_name As String) As Boolean
```

**Description**

Function reads page settings from the specified template. .

**Version**

Available since version 5.5.

**II.5.3.3 LoadCurrent****C++**

```
HRESULT LoadCurrent(VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadCurrent();
```

**Visual Basic**

```
Public Function LoadCurrent() As Boolean
```

**Description**

Function reads current page settings. .

**Version**

Available since version 5.5.

**II.5.3.4 Save****C++**

```
HRESULT Save(VARIANT_BOOL* ret);
```

**C#**

```
public bool Save();
```

**Visual Basic**

```
Public Function Save() As Boolean
```

**Description**

Function saves current page settings. .

**Version**

Available since version 5.5.

**II.5.3.5 SaveAs****C++**

```
HRESULT SaveAs(BSTR _template_name, VARIANT_BOOL* ret);
```

**C#**

```
public bool SaveAs(String _template_name);
```

**Visual Basic**

```
Public Function SaveAs(_template_name As String) As Boolean
```

**Description**

Function saves current page settings as a template with the specified name. .

**Version**

Available since version 5.5.

**II.6 IRobotOutputFormat****C++**

```
enum IRobotOutputFormat;
```

**C#**

```
public enum IRobotOutputFormat;
```

**Visual Basic**

```
Public Enum IRobotOutputFormat
```

**Members**

Members	Description
I_OFF_TEXT = 1	Available since version 1.7.
I_OFF_RTF = 2	Available since version 1.7.
I_OFF_RTF_JPEG = 3	RTF format file with the inserted JPEG format graphics. Available since version 5.5.
I_OFF_HTML = 4	HTML format file. Available since version 7.5.
I_OFF_PNG = 5	PNG format file. Available since version 12.

**Description**

Available file formats to which a printout may be directed. .

**II.7 IRobotReportTemplateMngr****Class Hierarchy**

**C++**

```
interface IRobotReportTemplateMngr : IDispatch;
```

**C#**

```
public interface IRobotReportTemplateMngr;
```

**Visual Basic**

```
Public Interface IRobotReportTemplateMngr
```

**Description**

There is a possibility to save a previously composed printout as a template, which may be used many times afterwards in the same or different project. Printout templates are recognized by names. All templates available in the current project are managed by the template manager. .

**II.7.1 IRobotReportTemplateMngr Members**

The following tables list the members exposed by IRobotReportTemplateMngr.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1496)	Number of available printout templates Available since version 1.7.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Find (see page 1496)	Function finds a printout template with the specified name and returns its index. If the printout template is not found, then zero value is returned. Available since version 1.7.
◆	Get (see page 1497)	Function returns name of a template with a specified index. Printout templates are indexed from 1 to Count (see page 1496). Available since version 1.7.
◆	Remove (see page 1497)	Function deletes a printout template with the specified name. Available since version 1.7.

**II.7.2 IRobotReportTemplateMngr Fields**

The fields of the IRobotReportTemplateMngr class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1496)	Number of available printout templates Available since version 1.7.

**II.7.2.1 Count****C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of available printout templates Available since version 1.7.

## II.7.3 IRobotReportTemplateMngr Methods

The methods of the IRobotReportTemplateMngr class are listed here.

### Public Methods

	Name	Description
💡	Find (see page 1496)	Function finds a printout template with the specified name and returns its index. If the printout template is not found, then zero value is returned. Available since version 1.7.
💡	Get (see page 1497)	Function returns name of a template with a specified index. Printout templates are indexed from 1 to Count (see page 1496). Available since version 1.7.
💡	Remove (see page 1497)	Function deletes a printout template with the specified name. Available since version 1.7.

### II.7.3.1 Find

#### C++

```
HRESULT Find(BSTR _tmpl_name, long* ret);
```

#### C#

```
public long Find(String _tmpl_name);
```

#### Visual Basic

```
Public Function Find(_tmpl_name As String) As long
```

#### Description

Function finds a printout template with the specified name and returns its index. If the printout template is not found, then zero value is returned. Available since version 1.7.

### II.7.3.2 Get

#### C++

```
HRESULT Get(long _idx, BSTR* ret);
```

#### C#

```
public String Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As String
```

#### Description

Function returns name of a template with a specified index. Printout templates are indexed from 1 to Count (see page 1496). Available since version 1.7.

### II.7.3.3 Remove

#### C++

```
HRESULT Remove(BSTR _tmpl_name);
```

#### C#

```
public void Remove(String _tmpl_name);
```

**Visual Basic**

```
Public Sub Remove(_tmpl_name As String)
```

**Description**

Function deletes a printout template with the specified name. Available since version 1.7.

**II.8 IRobotReportStdElementRtf****Class Hierarchy****C++**

```
interface IRobotReportStdElementRtf : IDispatch;
```

**C#**

```
public interface IRobotReportStdElementRtf;
```

**Visual Basic**

```
Public Interface IRobotReportStdElementRtf
```

**Description**

Standard (predefined) printout element based on an RTF format template file.

**Version**

Available since version 2.5.

**II.8.1 IRobotReportStdElementRtf Members**

The following tables list the members exposed by IRobotReportStdElementRtf.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Active ( <a href="#">see page 1498</a> )	Flag indicating if the element is active (i.e. it will be added to the printout).
❖	Frame ( <a href="#">see page 1498</a> )	Frame type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	LoadFromFile ( <a href="#">see page 1499</a> )	Function reads in the printout element contents from the indicated file. .
❖	RestoreDefaults ( <a href="#">see page 1499</a> )	Function restores the standard contents of a printout element..
❖	SaveToFile ( <a href="#">see page 1500</a> )	Function saves an element to a file of selected format.

**II.8.2 IRobotReportStdElementRtf Fields**

The fields of the IRobotReportStdElementRtf class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Active ( <a href="#">see page 1498</a> )	Flag indicating if the element is active (i.e. it will be added to the printout).
❖	Frame ( <a href="#">see page 1498</a> )	Frame type.

**II.8.2.1 Active****C++**

```
HRESULT get_Active(VARIANT_BOOL* );
HRESULT put_Active(VARIANT_BOOL);
```

**C#**

```
public bool Active { get; set; }
```

**Visual Basic**

```
Public Active As Boolean
```

**Description**

Flag indicating if the element is active (i.e. it will be added to the printout).

**Version**

Available since version 2.5.

**II.8.2.2 Frame****C++**

```
HRESULT get_Frame(IRobotPageSetupFrameType* );
HRESULT put_Frame(IRobotPageSetupFrameType);
```

**C#**

```
public IRobotPageSetupFrameType Frame { get; set; }
```

**Visual Basic**

```
Public Frame As IRobotPageSetupFrameType
```

**Description**

Frame type.

**Version**

Available since version 5.5.

**II.8.3 IRobotReportStdElementRtf Methods**

The methods of the IRobotReportStdElementRtf class are listed here.

**Public Methods**

	Name	Description
➊	LoadFromFile (see page 1499)	Function reads in the printout element contents from the indicated file. .
➋	RestoreDefaults (see page 1499)	Function restores the standard contents of a printout element. .
➌	SaveToFile (see page 1500)	Function saves an element to a file of selected format.

**II.8.3.1 LoadFromFile****C++**

```
HRESULT LoadFromFile(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool LoadFromFile(String _file_path);
```

**Visual Basic**

```
Public Function LoadFromFile(_file_path As String) As Boolean
```

**Description**

Function reads in the printout element contents from the indicated file. .

**Version**

Available since version 2.5.

### II.8.3.2 RestoreDefaults

#### C++

```
HRESULT RestoreDefaults();
```

#### C#

```
public void RestoreDefaults();
```

#### Visual Basic

```
Public Sub RestoreDefaults()
```

#### Description

Function restores the standard contents of a printout element. .

#### Version

Available since version 2.5.

### II.8.3.3 SaveToFile

#### C++

```
HRESULT SaveToFile(BSTR _filepath, IRobotOutputFormat _format, VARIANT_BOOL* ret);
```

#### C#

```
public bool SaveToFile(String _filepath, IRobotOutputFormat _format);
```

#### Visual Basic

```
Public Function SaveToFile(_filepath As String, _format As IRobotOutputFormat) As Boolean
```

#### Description

Function saves an element to a file of selected format.

#### Version

Available since version 5.5.

## II.9 IRobotReportItemList

#### Class Hierarchy

#### C++

```
interface IRobotReportItemList : IDispatch;
```

#### C#

```
public interface IRobotReportItemList;
```

#### Visual Basic

```
Public Interface IRobotReportItemList
```

#### Description

List of printout components.

#### Version

Available since version 5.5.

## II.9.1 IRobotReportItemList Members

The following tables list the members exposed by IRobotReportItemList.

### Public Fields

	Name	Description
◆	Count ( [ see page 1501 )	Number of elements on the list.

### Public Methods

	Name	Description
◆	Get ( [ see page 1501 )	Function returns a printout component with the specified index. Printout components are indexed from 1 to Count ( [ see page 1501 ) ..

## II.9.2 IRobotReportItemList Fields

The fields of the IRobotReportItemList class are listed here.

### Public Fields

	Name	Description
◆	Count ( [ see page 1501 )	Number of elements on the list.

### II.9.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As Long
```

#### Description

Number of elements on the list.

#### Version

Available since version 5.5.

## II.9.3 IRobotReportItemList Methods

The methods of the IRobotReportItemList class are listed here.

### Public Methods

	Name	Description
◆	Get ( [ see page 1501 )	Function returns a printout component with the specified index. Printout components are indexed from 1 to Count ( [ see page 1501 ) ..

### II.9.3.1 Get

#### C++

```
HRESULT Get(long _idx, IRobotReportItem** ret);
```

#### C#

```
public IRobotReportItem Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotReportItem
```

**Description**

Function returns a printout component with the specified index. Printout components are indexed from 1 to Count (see page 1501). .

**Version**

Available since version 5.5.

**II.10 IRobotReportItem****Class Hierarchy****C++**

```
interface IRobotReportItem : IDispatch;
```

**C#**

```
public interface IRobotReportItem;
```

**Visual Basic**

```
Public Interface IRobotReportItem
```

**Description**

Interface that describes a printout element.

**Version**

Available since version 5.5.

**II.10.1 IRobotReportItem Members**

The following tables list the members exposed by IRobotReportItem.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	ExcludeFromReport (see page 1503)	
❖	HasNoteAfter (see page 1503)	Flag indicating that at a note has been inserted after the printout component .
❖	HasNoteBefore (see page 1503)	Flag indicating that a note has been inserted before the printout component .
❖	IncludeDateTime (see page 1504)	
❖	NoteAfter (see page 1504)	Note inserted after the printout component .
❖	NoteBefore (see page 1504)	Note inserted before the printout component.
❖	StartFromNewPage (see page 1505)	Flag that indicates if an element should be printed on a new page.
❖	Title (see page 1505)	Element title in RTF format.
❖	TitleText (see page 1505)	Element title.
❖	Type (see page 1505)	Type of printout element.

## Public Methods

	Name	Description
❖	CreateView ( [ see page 1506)	Fuction creates and returns a view corresponding to the printout component. A type of the created view depends on the contents of the printout component. If access to any of the defined interfaces for views is impossible, then the IRobotPrintable ( [ see page 1470) interface is returned. In case an error occurs, the empty reference is returned (zero for the C language, Nothing for Visual Basic). Function makes the interface accessible for the object reperesented by the printout component of I_RIT_VIEW type. .
❖	GetPageTemplate ( [ see page 1506)	Function returns the name of a page template for the printout component of I_RIT_PAGE_TEMPLATE type. .

## II.10.2 IRobotReportItem Fields

The fields of the IRobotReportItem class are listed here.

### Public Fields

	Name	Description
❖	ExcludeFromReport ( [ see page 1503)	
❖	HasNoteAfter ( [ see page 1503)	Flag indicating that at a note has been inserted after the printout component .
❖	HasNoteBefore ( [ see page 1503)	Flag indicating that a note has been inserted before the printout component .
❖	IncludeDateTime ( [ see page 1504)	
❖	NoteAfter ( [ see page 1504)	Note inserted after the printout component .
❖	NoteBefore ( [ see page 1504)	Note inserted before the printout component.
❖	StartFromNewPage ( [ see page 1505)	Flag that indicates if an element should be printed on a new page.
❖	Title ( [ see page 1505)	Element title in RTF format.
❖	TitleText ( [ see page 1505)	Element title.
❖	Type ( [ see page 1505)	Type of printout element.

### II.10.2.1 ExcludeFromReport

#### C++

```
HRESULT get_ExcludeFromReport(VARIANT_BOOL* );
HRESULT put_ExcludeFromReport(VARIANT_BOOL);
```

#### C#

```
public bool ExcludeFromReport { get; set; }
```

#### Visual Basic

```
Public ExcludeFromReport As Boolean
```

#### Version

Available since version 14.5.

### II.10.2.2 HasNoteAfter

#### C++

```
HRESULT get_HasNoteAfter(VARIANT_BOOL* );
```

#### C#

```
public bool HasNoteAfter { get; }
```

**Visual Basic**

```
Public ReadOnly HasNoteAfter As Boolean
```

**Description**

Flag indicating that at a note has been inserted after the printout component .

**Version**

Available since version 5.5.

**II.10.2.3 HasNoteBefore****C++**

```
HRESULT get_HasNoteBefore(VARIANT_BOOL* );
```

**C#**

```
public bool HasNoteBefore { get; }
```

**Visual Basic**

```
Public ReadOnly HasNoteBefore As Boolean
```

**Description**

Flag indicating that a note has been inserted before the printout component .

**Version**

Available since version 5.5.

**II.10.2.4 IncludeDateTime****C++**

```
HRESULT get_IncludeDateTime(VARIANT_BOOL* );
```

**C#**

```
public bool IncludeDateTime { get; }
```

**Visual Basic**

```
Public ReadOnly IncludeDateTime As Boolean
```

**Version**

Available since version 12.

**II.10.2.5 NoteAfter****C++**

```
HRESULT get_NoteAfter(IRobotRtfView** );
```

**C#**

```
public IRobotRtfView NoteAfter { get; }
```

**Visual Basic**

```
Public ReadOnly NoteAfter As IRobotRtfView
```

**Description**

Note inserted after the printout component .

**Version**

Available since version 5.5.

### II.10.2.6 NoteBefore

#### C++

```
HRESULT get_NoteBefore(IRobotRtfView**);
```

#### C#

```
public IRobotRtfView NoteBefore { get; }
```

#### Visual Basic

```
Public ReadOnly NoteBefore As IRobotRtfView
```

#### Description

Note inserted before the printout component.

#### Version

Available since version 5.5.

### II.10.2.7 StartFromNewPage

#### C++

```
HRESULT get_StartFromNewPage(VARIANT_BOOL*);  
HRESULT put_StartFromNewPage(VARIANT_BOOL);
```

#### C#

```
public bool StartFromNewPage { get; set; }
```

#### Visual Basic

```
Public StartFromNewPage As Boolean
```

#### Description

Flag that indicates if an element should be printed on a new page.

#### Version

Available since version 5.5.

### II.10.2.8 Title

#### C++

```
HRESULT get_Title(IRobotRtfView**);
```

#### C#

```
public IRobotRtfView Title { get; }
```

#### Visual Basic

```
Public ReadOnly Title As IRobotRtfView
```

#### Description

Element title in RTF format.

#### Version

Available since version 5.5.

### II.10.2.9 TitleText

#### C++

```
HRESULT get_TitleText(BSTR*);
```

**C#**

```
public String TitleText { get; }
```

**Visual Basic**

```
Public ReadOnly TitleText As String
```

**Description**

Element title.

**Version**

Available since version 5.5.

**II.10.2.10 Type****C++**

```
HRESULT get_Type(IRobotReportItemType* );
```

**C#**

```
public IRobotReportItemType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotReportItemType
```

**Description**

Type of printout element.

**Version**

Available since version 5.5.

**II.10.3 IRobotReportItem Methods**

The methods of the IRobotReportItem class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	CreateView (see page 1506)	Fuction creates and returns a view corresponding to the printout component. A type of the created view depends on the contents of the printout component. If access to any of the defined interfaces for views is impossible, then the IRobotPrintable (see page 1470) interface is returned. In case an error occurs, the empty reference is returned (zero for the C language, Nothing for Visual Basic). Function makes the interface accessible for the object reperesented by the printout component of I_RIT_VIEW type. .
💡	GetPageTemplate (see page 1506)	Function returns the name of a page template for the printout component of I_RIT_PAGE_TEMPLATE type. .

**II.10.3.1 CreateView****C++**

```
HRESULT CreateView(IDispatch** ret);
```

**C#**

```
public IDispatch* CreateView();
```

**Visual Basic**

```
Public Function CreateView() As IDispatch*
```

## Description

Fuction creates and returns a view corresponding to the printout component. A type of the created view depends on the contents of the printout component. If access to any of the defined interfaces for views is impossible, then the IRobotPrintable (see page 1470) interface is returned. In case an error occurs, the empty reference is returned (zero for the C language, Nothing for Visual Basic). Function makes the interface accessible for the object reperesented by the printout component of I\_RIT\_VIEW type..

## Version

Available since version 5.5.

### II.10.3.2 GetPageTemplate

#### C++

```
HRESULT GetPageTemplate(BSTR* ret);
```

#### C#

```
public String GetPageTemplate();
```

#### Visual Basic

```
Public Function GetPageTemplate() As String
```

## Description

Function returns the name of a page template for the printout component of I\_RIT\_PAGE\_TEMPLATE type. .

## Version

Available since version 5.5.

## II.11 IRobotPageSetupOrientation

#### C++

```
enum IRobotPageSetupOrientation;
```

#### C#

```
public enum IRobotPageSetupOrientation;
```

#### Visual Basic

```
Public Enum IRobotPageSetupOrientation
```

## Members

Members	Description
I_PSO_PORTRAIT = 1	Available since version 5.5.
I_PSO_LANDSCAPE = 2	Available since version 5.5.

## Description

Page setup.

## Version

Available since version 5.5.

## II.12 IRobotPageSetupFrameType

#### C++

```
enum IRobotPageSetupFrameType;
```

**C#**

```
public enum IRobotPageSetupFrameType;
```

**Visual Basic**

```
Public Enum IRobotPageSetupFrameType
```

**Members**

Members	Description
I_PSFT_NONE = 0	No frame. Available since version 5.5.
I_PSFT_FRAME = 1	Frame. Available since version 5.5.
I_PSFT_SEPARATION_LINE = 2	Separation line. Available since version 5.5.

**Description**

Types of frames used for decoration of individual page elements. .

**Version**

Available since version 5.5.

## II.13 IRobotPageSetupTableOfContents

**Class Hierarchy****C++**

```
interface IRobotPageSetupTableOfContents : IDispatch;
```

**C#**

```
public interface IRobotPageSetupTableOfContents;
```

**Visual Basic**

```
Public Interface IRobotPageSetupTableOfContents
```

**Description**

Interface that describes the table of contents.

**Version**

Available since version 5.5.

### II.13.1 IRobotPageSetupTableOfContents Members

The following tables list the members exposed by IRobotPageSetupTableOfContents.

**Public Fields**

	Name	Description
◆	Active ( [ see page 1508)	Flag that switches on/off the table of contents.
◆	IncludeTitle ( [ see page 1509)	
◆	Location ( [ see page 1509)	

### II.13.2 IRobotPageSetupTableOfContents Fields

The fields of the IRobotPageSetupTableOfContents class are listed here.

**Public Fields**

	Name	Description
◆	Active ( [ see page 1508)	Flag that switches on/off the table of contents.
◆	IncludeTitle ( [ see page 1509)	
◆	Location ( [ see page 1509)	

**II.13.2.1 Active****C++**

```
HRESULT get_Active(VARIANT_BOOL* );
HRESULT put_Active(VARIANT_BOOL);
```

**C#**

```
public bool Active { get; set; }
```

**Visual Basic**

```
Public Active As Boolean
```

**Description**

Flag that switches on/off the table of contents.

**Version**

Available since version 5.5.

**II.13.2.2 IncludeTitle****C++**

```
HRESULT get_IncludeTitle(VARIANT_BOOL* );
HRESULT put_IncludeTitle(VARIANT_BOOL);
```

**C#**

```
public bool IncludeTitle { get; set; }
```

**Visual Basic**

```
Public IncludeTitle As Boolean
```

**Version**

Available since version 5.5.

**II.13.2.3 Location****C++**

```
HRESULT get_Location(IRobotPageSetupTocLocation* );
HRESULT put_Location(IRobotPageSetupTocLocation);
```

**C#**

```
public IRobotPageSetupTocLocation Location { get; set; }
```

**Visual Basic**

```
Public Location As IRobotPageSetupTocLocation
```

**Version**

Available since version 5.5.

## II.14 IRobotPageSetupTocLocation

### C++

```
enum IRobotPageSetupTocLocation;
```

### C#

```
public enum IRobotPageSetupTocLocation;
```

### Visual Basic

```
Public Enum IRobotPageSetupTocLocation
```

### Members

Members	Description
I_PSTL_BEGINNING = 1	At the beginning. Available since version 5.5.
I_PSTL_END = 2	At the end. Available since version 5.5.

### Version

Available since version 5.5.

## II.15 IRobotReportItemType

### C++

```
enum IRobotReportItemType;
```

### C#

```
public enum IRobotReportItemType;
```

### Visual Basic

```
Public Enum IRobotReportItemType
```

### Members

Members	Description
I_RIT_BLANK_PAGE = 1	Printout component that represents a blank page. Available since version 5.5.
I_RIT_PAGE_TEMPLATE = 2	Printout component that enforces change of the page template . Available since version 5.5.
I_RIT_VIEW = 3	Printout component representing a view or other object that may be printed. Available since version 5.5.

### Description

Available types of printout components.

### Version

Available since version 5.5.

## II.16 IRobotExternalPreviewFormat

### C++

```
enum IRobotExternalPreviewFormat;
```

### C#

```
public enum IRobotExternalPreviewFormat;
```

### Visual Basic

```
Public Enum IRobotExternalPreviewFormat
```

### Members

Members	Description
EPF_HTML = 1	Print preview in the html file browser. Available since version 6.4.
EPF_MS_OFFICE = 2	Presentation of a printout in the Word program of the Microsoft Office package; it requires installing the Microsoft Office package. Available since version 6.4.
EPF_OPEN_OFFICE = 3	Presentation of a printout in the text editor of the Open Office package; it requires installing the Open Office package. Available since version 6.4.

### Description

External format of a print preview.

### Version

Available since version 6.4.

## II.17 IRobotVariableMngrExtension

### Class Hierarchy

### C++

```
interface IRobotVariableMngrExtension : IDispatch;
```

### C#

```
public interface IRobotVariableMngrExtension;
```

### Visual Basic

```
Public Interface IRobotVariableMngrExtension
```

### Description

Interface of external extension for the manager of variables (parameters) used in printouts.

### II.17.1 IRobotVariableMngrExtension Members

The following tables list the members exposed by IRobotVariableMngrExtension.

### Public Methods

	Name	Description
!	GetIndexedValue (see page 1512)	Function returns a value of the variable indexed for a given value of an index. If the object does not support the variable of a given name, then function returns FALSE value (zero).

	GetValue ( <a href="#">see page 1512</a> )	Function fills out a value of the variable of a given name. If the object does not support this variable, then function returns FALSE value (zero).
-----------------------------------------------------------------------------------	--------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------

## II.17.2 IRobotVariableMngrExtension Methods

The methods of the IRobotVariableMngrExtension class are listed here.

### Public Methods

	Name	Description
	GetIndexedValue ( <a href="#">see page 1512</a> )	Function returns a value of the variable indexed for a given value of an index. If the object does not support the variable of a given name, then function returns FALSE value (zero).
	GetValue ( <a href="#">see page 1512</a> )	Function fills out a value of the variable of a given name. If the object does not support this variable, then function returns FALSE value (zero).

### II.17.2.1 GetIndexedValue

#### C++

```
HRESULT GetIndexedValue(BSTR _var_name, long _cur_index, long _max_index, BSTR _var_value,
VARIANT_BOOL* ret);
```

#### C#

```
public bool GetIndexedValue(String _var_name, long _cur_index, long _max_index, String
_var_value);
```

#### Visual Basic

```
Public Function GetIndexedValue(_var_name As String, _cur_index As long, _max_index As
long, ByRef _var_value As String) As Boolean
```

#### Description

Function returns a value of the variable indexed for a given value of an index. If the object does not support the variable of a given name, then function returns FALSE value (zero).

### II.17.2.2 GetValue

#### C++

```
HRESULT GetValue(BSTR _var_name, BSTR _var_value, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetValue(String _var_name, String _var_value);
```

#### Visual Basic

```
Public Function GetValue(_var_name As String, ByRef _var_value As String) As Boolean
```

#### Description

Function fills out a value of the variable of a given name. If the object does not support this variable, then function returns FALSE value (zero).

## III Structural axes

### Enumerations

	Name	Description
	IRobotStructuralAxisLabelType ( <a href="#">see page 1513</a> )	Available methods of structure axis description.

	IRobotStructuralAxisGridType ( <a href="#">see page 1518</a> )	Available types of structure axis grids.
-----------------------------------------------------------------------------------	----------------------------------------------------------------	------------------------------------------

## Interfaces

	Name	Description
 	IRobotStructuralAxisSequenceList ( <a href="#">see page 1513</a> )	List of structure axis sequences.
 	IRobotStructuralAxisGrid ( <a href="#">see page 1518</a> )	Structure axis grid.
 	IRobotStructuralAxisGridCartesian ( <a href="#">see page 1520</a> )	Cartesian grid of structure axes.
 	IRobotStructuralAxisGridMngr ( <a href="#">see page 1524</a> )	Manager of structure axis grids.

## III.1 IRobotStructuralAxisLabelType

### C++

```
enum IRobotStructuralAxisLabelType;
```

### C#

```
public enum IRobotStructuralAxisLabelType;
```

### Visual Basic

```
Public Enum IRobotStructuralAxisLabelType
```

### Members

Members	Description
I_SALT_123 = 1	Description with successive numbers.
I_SALT_ABC = 2	Description with successive alphabet letters.
I_SALT_VALUE = 3	Description with values.
I_SALT_DEFINE = 4	Description with a defined character string.
I_SALT_VARIOUS = 5	Application of different methods of axis description.

### Description

Available methods of structure axis description.

## III.2 IRobotStructuralAxisSequenceList

### Class Hierarchy

### C++

```
interface IRobotStructuralAxisSequenceList : IDispatch;
```

### C#

```
public interface IRobotStructuralAxisSequenceList;
```

### Visual Basic

```
Public Interface IRobotStructuralAxisSequenceList
```

### Description

List of structure axis sequences.

### III.2.1 IRobotStructuralAxisSequenceList Members

The following tables list the members exposed by IRobotStructuralAxisSequenceList.

#### Public Fields

	Name	Description
◆	AxisCount ( <a href="#">see page 1514</a> )	Number of defined structure axes.
◆	SequenceCount ( <a href="#">see page 1514</a> )	Number of defined structure axis sequences.
◆	StartPosition ( <a href="#">see page 1515</a> )	Beginning point of the list.

#### Public Methods

	Name	Description
◆	AddSequence ( <a href="#">see page 1515</a> )	Function inserts a new structure axis sequence. While creating the first sequence, the start axis will also be created at the position StartPosition ( <a href="#">see page 1515</a> ).
◆	Clear ( <a href="#">see page 1516</a> )	Function deletes all structure axes.
◆	DeleteSequence ( <a href="#">see page 1516</a> )	Function deletes the structure axis sequence of a given index.
◆	FindAxisByPos ( <a href="#">see page 1516</a> )	Function returns an index of the axis of a given position. If no structure axis has been defined at the given position, then function returns zero value.
◆	GetAxis ( <a href="#">see page 1516</a> )	Function returns data for the structure axis of a given index.
◆	GetSequence ( <a href="#">see page 1517</a> )	Function returns data for the sequence of a given index.
◆	SetAxisLabel ( <a href="#">see page 1517</a> )	Function changes a description for the structure axis of a given index.
◆	SetLabelFormat ( <a href="#">see page 1517</a> )	Function sets the current method of structure axis description.
◆	SingleOutAxis ( <a href="#">see page 1518</a> )	Function steers with singling out of a structure axis in graphical views.

### III.2.2 IRobotStructuralAxisSequenceList Fields

The fields of the IRobotStructuralAxisSequenceList class are listed here.

#### Public Fields

	Name	Description
◆	AxisCount ( <a href="#">see page 1514</a> )	Number of defined structure axes.
◆	SequenceCount ( <a href="#">see page 1514</a> )	Number of defined structure axis sequences.
◆	StartPosition ( <a href="#">see page 1515</a> )	Beginning point of the list.

#### III.2.2.1 AxisCount

##### C++

```
HRESULT get_AxisCount(long*);
```

##### C#

```
public long AxisCount { get; }
```

##### Visual Basic

```
Public ReadOnly AxisCount As long
```

##### Description

Number of defined structure axes.

#### III.2.2.2 SequenceCount

##### C++

```
HRESULT get_SequenceCount(long*);
```

**C#**

```
public long SequenceCount { get; }
```

**Visual Basic**

```
Public ReadOnly SequenceCount As Long
```

**Description**

Number of defined structure axis sequences.

**III.2.2.3 StartPosition****C++**

```
HRESULT get_StartPosition(double*);  
HRESULT put_StartPosition(double);
```

**C#**

```
public double StartPosition { get; set; }
```

**Visual Basic**

```
Public StartPosition As Double
```

**Description**

Beginning point of the list.

**III.2.3 IRobotStructuralAxisSequenceList Methods**

The methods of the IRobotStructuralAxisSequenceList class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	AddSequence (see page 1515)	Function inserts a new structure axis sequence. While creating the first sequence, the start axis will also be created at the position StartPosition (see page 1515).
💡	Clear (see page 1516)	Function deletes all structure axes.
💡	DeleteSequence (see page 1516)	Function deletes the structure axis sequence of a given index.
💡	FindAxisByPos (see page 1516)	Function returns an index of the axis of a given position. If no structure axis has been defined at the given position, then function returns zero value.
💡	GetAxis (see page 1516)	Function returns data for the structure axis of a given index.
💡	GetSequence (see page 1517)	Function returns data for the sequence of a given index.
💡	SetAxisLabel (see page 1517)	Function changes a description for the structure axis of a given index.
💡	SetLabelFormat (see page 1517)	Function sets the current method of structure axis description.
💡	SingleOutAxis (see page 1518)	Function steers with singling out of a structure axis in graphical views.

**III.2.3.1 AddSequence****C++**

```
HRESULT AddSequence(double _distance, long _repeat_number);
```

**C#**

```
public void AddSequence(double _distance, long _repeat_number);
```

**Visual Basic**

```
Public Sub AddSequence(_distance As Double, _repeat_number As Long)
```

## Description

Function inserts a new structure axis sequence. While creating the first sequence, the start axis will also be created at the position StartPosition (see page 1515).

### III.2.3.2 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

Function deletes all structure axes.

### III.2.3.3 DeleteSequence

#### C++

```
HRESULT DeleteSequence(long _idx);
```

#### C#

```
public void DeleteSequence(long _idx);
```

#### Visual Basic

```
Public Sub DeleteSequence(_idx As long)
```

#### Description

Function deletes the structure axis sequence of a given index.

### III.2.3.4 FindAxisByPos

#### C++

```
HRESULT FindAxisByPos(double _pos, long* ret);
```

#### C#

```
public long FindAxisByPos(double _pos);
```

#### Visual Basic

```
Public Function FindAxisByPos(_pos As double) As long
```

#### Description

Function returns an index of the axis of a given position. If no structure axis has been defined at the given position, then function returns zero value.

### III.2.3.5 GetAxis

#### C++

```
HRESULT GetAxis(long _idx, BSTR _label, double* _position, bool* _singleout, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetAxis(long _idx, String _label, double* _position, bool* _singleout);
```

**Visual Basic**

```
Public Function GetAxis(_idx As long, ByRef _label As String, ByRef _position As double*,  
ByRef _singleout As bool*) As Boolean
```

**Description**

Function returns data for the structure axis of a given index.

**III.2.3.6 GetSequence****C++**

```
HRESULT GetSequence(long _idx, double* _distance, long* _repeat_number);
```

**C#**

```
public void GetSequence(long _idx, double* _distance, long* _repeat_number);
```

**Visual Basic**

```
Public Sub GetSequence(_idx As long, ByRef _distance As double*, ByRef _repeat_number As  
long*)
```

**Description**

Function returns data for the sequence of a given index.

**III.2.3.7 SetAxisLabel****C++**

```
HRESULT SetAxisLabel(long _idx, BSTR _label);
```

**C#**

```
public void SetAxisLabel(long _idx, String _label);
```

**Visual Basic**

```
Public Sub SetAxisLabel(_idx As long, _label As String)
```

**Description**

Function changes a description for the structure axis of a given index.

**III.2.3.8 SetLabelFormat****C++**

```
HRESULT SetLabelFormat(IRobotStructuralAxisLabelType _label_type, BSTR _label_def = "",  
VARIANT_BOOL _reversed = false);
```

**C#**

```
public void SetLabelFormat(IRobotStructuralAxisLabelType _label_type, String _label_def =  
"", bool _reversed = false);
```

**Visual Basic**

```
Public Sub SetLabelFormat(_label_type As IRobotStructuralAxisLabelType, Optional _label_def  
As String = "", Optional _reversed As Boolean = false)
```

**Description**

Function sets the current method of structure axis description.

### III.2.3.9 SingleOutAxis

**C++**

```
HRESULT SingleOutAxis(long _idx, VARIANT_BOOL _singleout = true);
```

**C#**

```
public void SingleOutAxis(long _idx, bool _singleout = true);
```

**Visual Basic**

```
Public Sub SingleOutAxis(_idx As long, Optional _singleout As Boolean = true)
```

**Description**

Function steers with singling out of a structure axis in graphical views.

## III.3 IRobotStructuralAxisGridType

**C++**

```
enum IRobotStructuralAxisGridType;
```

**C#**

```
public enum IRobotStructuralAxisGridType;
```

**Visual Basic**

```
Public Enum IRobotStructuralAxisGridType
```

**Members**

Members	Description
I_SAGT_CARTESIAN = 1	Cartesian grid.
I_SAGT_CYLINDRICAL = 2	Cylindrical grid.

**Description**

Available types of structure axis grids.

## III.4 IRobotStructuralAxisGrid

**Class Hierarchy**

**C++**

```
interface IRobotStructuralAxisGrid : IDispatch;
```

**C#**

```
public interface IRobotStructuralAxisGrid;
```

**Visual Basic**

```
Public Interface IRobotStructuralAxisGrid
```

**Description**

Structure axis grid.

### III.4.1 IRobotStructuralAxisGrid Members

The following tables list the members exposed by IRobotStructuralAxisGrid.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Name ( <a href="#">see page 1519</a> )	Grid name.
❖	Type ( <a href="#">see page 1519</a> )	Grid type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Save ( <a href="#">see page 1520</a> )	Function saves the grid definition.

**III.4.2 IRobotStructuralAxisGrid Fields**

The fields of the IRobotStructuralAxisGrid class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Name ( <a href="#">see page 1519</a> )	Grid name.
❖	Type ( <a href="#">see page 1519</a> )	Grid type.

**III.4.2.1 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Grid name.

**III.4.2.2 Type****C++**

```
HRESULT get_Type(IRobotStructuralAxisGridType* );
```

**C#**

```
public IRobotStructuralAxisGridType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotStructuralAxisGridType
```

**Description**

Grid type.

**III.4.3 IRobotStructuralAxisGrid Methods**

The methods of the IRobotStructuralAxisGrid class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Save ( <a href="#">see page 1520</a> )	Function saves the grid definition.

### III.4.3.1 Save

**C++**

```
HRESULT Save();
```

**C#**

```
public void Save();
```

**Visual Basic**

```
Public Sub Save()
```

**Description**

Function saves the grid definition.

## III.5 IRobotStructuralAxisGridCartesian

**Class Hierarchy**

**C++**

```
interface IRobotStructuralAxisGridCartesian : IRobotStructuralAxisGrid;
```

**C#**

```
public interface IRobotStructuralAxisGridCartesian : IRobotStructuralAxisGrid;
```

**Visual Basic**

```
Public Interface IRobotStructuralAxisGridCartesian
```

**Description**

Cartesian grid of structure axes.

### III.5.1 IRobotStructuralAxisGridCartesian Members

The following tables list the members exposed by IRobotStructuralAxisGridCartesian.

**Public Fields**

	Name	Description
❖	IncludeStoreysInZ (see page 1521)	Flag indicating that the sequence of structure axes defined on the OZ axis, should take stories into account automatically.
❖	Name (see page 1519)	Grid name.
❖	RotationAngle (see page 1521)	Angle of grid rotation.
❖	RotationAxis (see page 1522)	Axis of grid rotation.
❖	StoreysInZ (see page 1522)	Selection of stories included on the OZ axis, if the IncludeStoreysInZ (see page 1521) flag has a True value.
❖	Type (see page 1519)	Grid type.
❖	X (see page 1522)	Sequence of structure axes defined on the OX axis.
❖	Y (see page 1522)	Sequence of structure axes defined on the OY axis.
❖	Z (see page 1523)	Sequence of structure axes defined on the OZ axis.

**Public Methods**

	Name	Description
❖	GetRelativeToPoint (see page 1523)	Function returns information about the point with respect to which the grid coordinates were defined. If (0,0,0) is the reference point, the function returns the False value.

	Save (see page 1520)	Function saves the grid definition.
	SetRelativeToPoint (see page 1523)	Function specifies coordinates of the point with respect to which the grid coordinates were defined.

### III.5.2 IRobotStructuralAxisGridCartesian Fields

The fields of the IRobotStructuralAxisGridCartesian class are listed here.

#### Public Fields

	Name	Description
	IncludeStoreysInZ (see page 1521)	Flag indicating that the sequence of structure axes defined on the OZ axis, should take stories into account automatically.
	RotationAngle (see page 1521)	Angle of grid rotation.
	RotationAxis (see page 1522)	Axis of grid rotation.
	StoreysInZ (see page 1522)	Selection of stories included on the OZ axis, if the IncludeStoreysInZ (see page 1521) flag has a True value.
	X (see page 1522)	Sequence of structure axes defined on the OX axis.
	Y (see page 1522)	Sequence of structure axes defined on the OY axis.
	Z (see page 1523)	Sequence of structure axes defined on the OZ axis.

#### III.5.2.1 IncludeStoreysInZ

##### C++

```
HRESULT get_IncludeStoreysInZ(VARIANT_BOOL* );
HRESULT put_IncludeStoreysInZ(VARIANT_BOOL);
```

##### C#

```
public bool IncludeStoreysInZ { get; set; }
```

##### Visual Basic

```
Public IncludeStoreysInZ As Boolean
```

##### Description

Flag indicating that the sequence of structure axes defined on the OZ axis, should take stories into account automatically.

##### Version

Available since version 9.5.

#### III.5.2.2 RotationAngle

##### C++

```
HRESULT get_RotationAngle(double* );
HRESULT put_RotationAngle(double);
```

##### C#

```
public double RotationAngle { get; set; }
```

##### Visual Basic

```
Public RotationAngle As Double
```

##### Description

Angle of grid rotation.

#### III.5.2.3 RotationAxis

##### C++

```
HRESULT get_RotationAxis(IRobotGeoCoordinateAxis* );
```

```
HRESULT put_RotationAxis(IRobotGeoCoordinateAxis);
```

**C#**

```
public IRobotGeoCoordinateAxis RotationAxis { get; set; }
```

**Visual Basic**

```
Public RotationAxis As IRobotGeoCoordinateAxis
```

**Description**

Axis of grid rotation.

### III.5.2.4 StoreysInZ

**C++**

```
HRESULT get_StoreysInZ(IRobotStoreySelection**);
```

**C#**

```
public IRobotStoreySelection StoreysInZ { get; }
```

**Visual Basic**

```
Public ReadOnly StoreysInZ As IRobotStoreySelection
```

**Description**

Selection of stories included on the OZ axis, if the IncludeStoreysInZ (see page 1521) flag has a True value.

**Version**

Available since version 9.7.

### III.5.2.5 X

**C++**

```
HRESULT get_X(IRobotStructuralAxisSequenceList**);
```

**C#**

```
public IRobotStructuralAxisSequenceList X { get; }
```

**Visual Basic**

```
Public ReadOnly X As IRobotStructuralAxisSequenceList
```

**Description**

Sequence of structure axes defined on the OX axis.

### III.5.2.6 Y

**C++**

```
HRESULT get_Y(IRobotStructuralAxisSequenceList**);
```

**C#**

```
public IRobotStructuralAxisSequenceList Y { get; }
```

**Visual Basic**

```
Public ReadOnly Y As IRobotStructuralAxisSequenceList
```

**Description**

Sequence of structure axes defined on the OY axis.

### III.5.2.7 Z

#### C++

```
HRESULT get_Z(IRobotStructuralAxisSequenceList**);
```

#### C#

```
public IRobotStructuralAxisSequenceList Z { get; }
```

#### Visual Basic

```
Public ReadOnly Z As IRobotStructuralAxisSequenceList
```

#### Description

Sequence of structure axes defined on the OZ axis.

## III.5.3 IRobotStructuralAxisGridCartesian Methods

The methods of the IRobotStructuralAxisGridCartesian class are listed here.

#### Public Methods

	Name	Description
💡	GetRelativeToPoint (see page 1523)	Function returns information about the point with respect to which the grid coordinates were defined. If (0,0,0) is the reference point, the function returns the False value.
💡	SetRelativeToPoint (see page 1523)	Function specifies coordinates of the point with respect to which the grid coordinates were defined.

### III.5.3.1 GetRelativeToPoint

#### C++

```
HRESULT GetRelativeToPoint(double* _x, double* _y, double* _z, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetRelativeToPoint(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Function GetRelativeToPoint(ByRef _x As Double*, ByRef _y As Double*, ByRef _z As Double*) As Boolean
```

#### Description

Function returns information about the point with respect to which the grid coordinates were defined. If (0,0,0) is the reference point, the function returns the False value.

#### Version

Available since version 9.2.

### III.5.3.2 SetRelativeToPoint

#### C++

```
HRESULT SetRelativeToPoint(VARIANT_BOOL _set, double _x = 0, double _y = 0, double _z = 0);
```

#### C#

```
public void SetRelativeToPoint(bool _set, double _x = 0, double _y = 0, double _z = 0);
```

#### Visual Basic

```
Public Sub SetRelativeToPoint(_set As Boolean, Optional _x As Double = 0, Optional _y As Double = 0, Optional _z As Double = 0)
```

## Description

Function specifies coordinates of the point with respect to which the grid coordinates were defined.

## Version

Available since version 9.2.

## III.6 IRobotStructuralAxisGridMngr

### Class Hierarchy

#### C++

```
interface IRobotStructuralAxisGridMngr : IDispatch;
```

#### C#

```
public interface IRobotStructuralAxisGridMngr;
```

### Visual Basic

```
Public Interface IRobotStructuralAxisGridMngr
```

## Description

Manager of structure axis grids.

## III.6.1 IRobotStructuralAxisGridMngr Members

The following tables list the members exposed by IRobotStructuralAxisGridMngr.

### Public Fields

	Name	Description
➊	Count (see page 1525)	Number of defined grids.

### Public Methods

	Name	Description
➊	Activate (see page 1525)	Function activates/deactivates the grid of a given index.
➋	Clear (see page 1525)	Function deletes all grids.
➌	Create (see page 1526)	Function creates and returns a new blank grid of a defined type.
➍	Delete (see page 1526)	Function deletes the grid of a given index.
➎	FindByIndex (see page 1526)	Function returns an index of the grid of a given name or a zero value if the grid is not found.
➏	Get (see page 1527)	Function makes available the grid of a given index.
➐	GetByName (see page 1527)	Function makes available the grid of a given name.
➑	IsActive (see page 1527)	Function returns activity state of the grid of a given index.

## III.6.2 IRobotStructuralAxisGridMngr Fields

The fields of the IRobotStructuralAxisGridMngr class are listed here.

### Public Fields

	Name	Description
➊	Count (see page 1525)	Number of defined grids.

### III.6.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As Long
```

**Description**

Number of defined grids.

**III.6.3 IRobotStructuralAxisGridMngr Methods**

The methods of the IRobotStructuralAxisGridMngr class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
➊	Activate (see page 1525)	Function activates/deactivates the grid of a given index.
➋	Clear (see page 1525)	Function deletes all grids.
➌	Create (see page 1526)	Function creates and returns a new blank grid of a defined type.
➍	Delete (see page 1526)	Function deletes the grid of a given index.
➎	FindByName (see page 1526)	Function returns an index of the grid of a given name or a zero value if the grid is not found.
➏	Get (see page 1527)	Function makes available the grid of a given index.
➐	GetByName (see page 1527)	Function makes available the grid of a given name.
➑	IsActive (see page 1527)	Function returns activity state of the grid of a given index.

**III.6.3.1 Activate****C++**

```
HRESULT Activate(long _idx, VARIANT_BOOL _activate = true);
```

**C#**

```
public void Activate(long _idx, bool _activate = true);
```

**Visual Basic**

```
Public Sub Activate(_idx As Long, Optional _activate As Boolean = true)
```

**Description**

Function activates/deactivates the grid of a given index.

**III.6.3.2 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**Description**

Function deletes all grids.

### III.6.3.3 Create

**C++**

```
HRESULT Create(IRobotStructuralAxisGridType _type, BSTR _name, IRobotStructuralAxisGrid** ret);
```

**C#**

```
public IRobotStructuralAxisGrid Create(IRobotStructuralAxisGridType _type, String _name);
```

**Visual Basic**

```
Public Function Create(_type As IRobotStructuralAxisGridType, _name As String) As IRobotStructuralAxisGrid
```

**Description**

Function creates and returns a new blank grid of a defined type.

### III.6.3.4 Delete

**C++**

```
HRESULT Delete(long _idx);
```

**C#**

```
public void Delete(long _idx);
```

**Visual Basic**

```
Public Sub Delete(_idx As long)
```

**Description**

Function deletes the grid of a given index.

### III.6.3.5 FindByName

**C++**

```
HRESULT FindByName(BSTR _name, long* ret);
```

**C#**

```
public long FindByName(String _name);
```

**Visual Basic**

```
Public Function FindByName(_name As String) As long
```

**Description**

Function returns an index of the grid of a given name or a zero value if the grid is not found.

### III.6.3.6 Get

**C++**

```
HRESULT Get(long _idx, IRobotStructuralAxisGrid** ret);
```

**C#**

```
public IRobotStructuralAxisGrid Get(long _idx);
```

**Visual Basic**

```
Public Function Get(_idx As long) As IRobotStructuralAxisGrid
```

**Description**

Function makes available the grid of a given index.

**III.6.3.7 GetByName****C++**

```
HRESULT GetByName(BSTR _name, IRobotStructuralAxisGrid** ret);
```

**C#**

```
public IRobotStructuralAxisGrid GetByName(String _name);
```

**Visual Basic**

```
Public Function GetByName(_name As String) As IRobotStructuralAxisGrid
```

**Description**

Function makes available the grid of a given name.

**III.6.3.8 IsActive****C++**

```
HRESULT IsActive(long _idx, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsActive(long _idx);
```

**Visual Basic**

```
Public Function IsActive(_idx As long) As Boolean
```

**Description**

Function returns activity state of the grid of a given index.

**Application - main object of the model****Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotDirectory ( <a href="#">see page 1536</a> )	The set of identifiers representing folders used by Robot program.
	IRobotLanguage ( <a href="#">see page 1537</a> )	
	IRobotQuitOption ( <a href="#">see page 1537</a> )	Set of identifiers describing the manner of exiting the program. .
	IRobotDirectoryExtension ( <a href="#">see page 1538</a> )	Set of identifiers representing extensions for Robot program folders. .
	IRobotLicenseEntitlement ( <a href="#">see page 1539</a> )	
	IRobotLicenseEntitlementStatus ( <a href="#">see page 1539</a> )	
	IRobotCalculationsType ( <a href="#">see page 1540</a> )	

## Interfaces

	Name	Description
↳	IRobotApplication ( <a href="#">see page 1528</a> )	Interface representing the entire application is the main object of the Robot Object Model. .
↳	IRobotPreferences ( <a href="#">see page 1533</a> )	Interface describing the current settings for the entire application. .
↳	IRobotPreferencesEvents ( <a href="#">see page 1538</a> )	Events related to the change of settings for the whole application.

# IIRobotApplication

## Class Hierarchy

### C++

```
interface IRobotApplication : IDispatch;
```

### C#

```
public interface IRobotApplication;
```

### Visual Basic

```
Public Interface IRobotApplication
```

## Description

Interface representing the entire application is the main object of the Robot Object Model. .

## I.1 IIRobotApplication Members

The following tables list the members exposed by IRobotApplication.

### Public Fields

	Name	Description
❖	CmpntFactory ( <a href="#">see page 1529</a> )	
❖	Interactive ( <a href="#">see page 1530</a> )	Flag indicating if interactive work with the program is expected .
❖	Is360 ( <a href="#">see page 1530</a> )	
❖	Kernel ( <a href="#">see page 1530</a> )	Component enabling access to the object of the Robot application by means of the IRobotKernel ( <a href="#">see page 2195</a> ) interface.
❖	Preferences ( <a href="#">see page 1530</a> )	The object describing application parameters. .
❖	ProgramVersion ( <a href="#">see page 1531</a> )	Designation of the Robot program version .
❖	Project ( <a href="#">see page 1531</a> )	Current project .
❖	UserControl ( <a href="#">see page 1531</a> )	Flag used for giving/taking back program navigation to/from the interactive user .
❖	Version ( <a href="#">see page 1532</a> )	Indicator of Robot Object Model saved in the following format [Version].[Extended] (e.g. 1.02) .
❖	Visible ( <a href="#">see page 1532</a> )	Flag of the main Robot window visibility .
❖	Window ( <a href="#">see page 1532</a> )	Main window of Robot program Available since version 1.7.

### Public Methods

	Name	Description
❖	LicenseCheckEntitlement ( <a href="#">see page 1533</a> )	The function returns status of a given entitlement.
❖	Quit ( <a href="#">see page 1533</a> )	Function closes the program. Available since version 1.7.

## I.2 IRobotApplication Fields

The fields of the IRobotApplication class are listed here.

### Public Fields

	Name	Description
◆	CmpntFactory (see page 1529)	
◆	Interactive (see page 1530)	Flag indicating if interactive work with the program is expected .
◆	Is360 (see page 1530)	
◆	Kernel (see page 1530)	Component enabling access to the object of the Robot application by means of the IRobotKernel (see page 2195) interface.
◆	Preferences (see page 1530)	The object describing application parameters. .
◆	ProgramVersion (see page 1531)	Designation of the Robot program version .
◆	Project (see page 1531)	Current project .
◆	UserControl (see page 1531)	Flag used for giving/taking back program navigation to/from the interactive user .
◆	Version (see page 1532)	Indicator of Robot Object Model saved in the following format [Version].[Extended] (e.g. 1.02) .
◆	Visible (see page 1532)	Flag of the main Robot window visibility .
◆	Window (see page 1532)	Main window of Robot program Available since version 1.7.

### I.2.1 CmpntFactory

#### C++

```
HRESULT get_CmpntFactory(IRobotComponentFactory**);
```

#### C#

```
public IRobotComponentFactory CmpntFactory { get; }
```

#### Visual Basic

```
Public ReadOnly CmpntFactory As IRobotComponentFactory
```

#### Version

Available since version 2.5.

### I.2.2 Interactive

#### C++

```
HRESULT get_Interactive(VARIANT_BOOL*);  
HRESULT put_Interactive(VARIANT_BOOL);
```

#### C#

```
public bool Interactive { get; set; }
```

#### Visual Basic

```
Public Interactive As Boolean
```

#### Description

Flag indicating if interactive work with the program is expected .

### I.2.3 Is360

#### C++

```
HRESULT get_Is360(VARIANT_BOOL*);
```

**C#**

```
public bool Is360 { get; }
```

**Visual Basic**

```
Public ReadOnly Is360 As Boolean
```

**Version**

Available since version 13.4.

## I.2.4 Kernel

**C++**

```
HRESULT get_Kernel(IRobotKernel**);
```

**C#**

```
public IRobotKernel Kernel { get; }
```

**Visual Basic**

```
Public ReadOnly Kernel As IRobotKernel
```

**Description**

Component enabling access to the object of the Robot application by means of the IRobotKernel (see page 2195) interface.

**Version**

Available since version 2.5.

## I.2.5 Preferences

**C++**

```
HRESULT get_Preferences(IRobotPreferences**);
```

**C#**

```
public IRobotPreferences Preferences { get; }
```

**Visual Basic**

```
Public ReadOnly Preferences As IRobotPreferences
```

**Description**

The object describing application parameters. .

## I.2.6 ProgramVersion

**C++**

```
HRESULT get_ProgramVersion(BSTR*);
```

**C#**

```
public String ProgramVersion { get; }
```

**Visual Basic**

```
Public ReadOnly ProgramVersion As String
```

**Description**

Designation of the Robot program version .

**Version**

Available since version 3.

## I.2.7 Project

### C++

```
HRESULT get_Project(IRobotProject**);
```

### C#

```
public IRobotProject Project { get; }
```

### Visual Basic

```
Public ReadOnly Project As IRobotProject
```

### Description

Current project .

## I.2.8 UserControl

### C++

```
HRESULT get_UserControl(VARIANT_BOOL*);  
HRESULT put_UserControl(VARIANT_BOOL);
```

### C#

```
public bool UserControl { get; set; }
```

### Visual Basic

```
Public UserControl As Boolean
```

### Description

Flag used for giving/taking back program navigation to/from the interactive user .

### Version

Available since version 3.

## I.2.9 Version

### C++

```
HRESULT get_Version(double*);
```

### C#

```
public double Version { get; }
```

### Visual Basic

```
Public ReadOnly Version As Double
```

### Description

Indicator of Robot Object Model saved in the following format [Version].[Extended] (e.g. 1.02) .

## I.2.10 Visible

### C++

```
HRESULT get_Visible(VARIANT_BOOL*);  
HRESULT put_Visible(VARIANT_BOOL);
```

### C#

```
public bool Visible { get; set; }
```

**Visual Basic**

```
Public Visible As Boolean
```

**Description**

Flag of the main Robot window visibility .

**I.2.11 Window****C++**

```
HRESULT get_Window(IRobotWindow**);
```

**C#**

```
public IRobotWindow Window { get; }
```

**Visual Basic**

```
Public ReadOnly Window As IRobotWindow
```

**Description**

Main window of Robot program Available since version 1.7.

**I.3 IRobotApplication Methods**

The methods of the IRobotApplication class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	LicenseCheckEntitlement (see page 1533)	The function returns status of a given entitlement.
💡	Quit (see page 1533)	Function closes the program. Available since version 1.7.

**I.3.1 LicenseCheckEntitlement****C++**

```
HRESULT LicenseCheckEntitlement(IRobotLicenseEntitlement _entitlement,
IRobotLicenseEntitlementStatus* ret);
```

**C#**

```
public IRobotLicenseEntitlementStatus LicenseCheckEntitlement(IRobotLicenseEntitlement
_entitlement);
```

**Visual Basic**

```
Public Function LicenseCheckEntitlement(_entitlement As IRobotLicenseEntitlement) As
IRobotLicenseEntitlementStatus
```

**Description**

The function returns status of a given entitlement.

**Version**

Available since version 13.4.

**I.3.2 Quit****C++**

```
HRESULT Quit(IRobotQuitOption _quit_option);
```

**C#**

```
public void Quit(IRobotQuitOption _quit_option);
```

**Visual Basic**

```
Public Sub Quit(_quit_option As IRobotQuitOption)
```

**Description**

Function closes the program. Available since version 1.7.

## II IRobotPreferences

**Class Hierarchy****C++**

```
interface IRobotPreferences : IDispatch;
```

**C#**

```
public interface IRobotPreferences;
```

**Visual Basic**

```
Public Interface IRobotPreferences
```

**Description**

Interface describing the current settings for the entire application. .

### II.1 IRobotPreferences Members

The following tables list the members exposed by IRobotPreferences.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CalculationsType (see page 1534)	
❖	CloudCalculationsEnabled (see page 1534)	
❖	Multiprocessing (see page 1535)	Flag activating parallel processing - multiprocessing.
❖	OpenGL (see page 1535)	Flag that enables switching on/off OpenGL graphics.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	GetDirectory (see page 1536)	The function returns full path to the desired Robot folder. .
❖	GetLanguage (see page 1536)	Function returns the current setting of the specified language. Code (number) of the appropriate country is returned (e.g. 33 for France).

### II.2 IRobotPreferences Fields

The fields of the IRobotPreferences class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CalculationsType (see page 1534)	

◆	CloudCalculationsEnabled (see page 1534)	
◆	Multiprocessing (see page 1535)	Flag activating parallel processing - multiprocessing.
◆	OpenGL (see page 1535)	Flag that enables switching on/off OpenGL graphics.

## II.2.1 CalculationsType

**C++**

```
HRESULT get_CalculationsType(IRobotCalculationsType* );
HRESULT put_CalculationsType(IRobotCalculationsType);
```

**C#**

```
public IRobotCalculationsType CalculationsType { get; set; }
```

**Visual Basic**

```
Public CalculationsType As IRobotCalculationsType
```

**Version**

Available since version 13.4.

## II.2.2 CloudCalculationsEnabled

**C++**

```
HRESULT get_CloudCalculationsEnabled(VARIANT_BOOL* );
HRESULT put_CloudCalculationsEnabled(VARIANT_BOOL);
```

**C#**

```
public bool CloudCalculationsEnabled { get; set; }
```

**Visual Basic**

```
Public CloudCalculationsEnabled As Boolean
```

**Version**

Available since version 13.4.

## II.2.3 Multiprocessing

**C++**

```
HRESULT get_Multiprocessing(VARIANT_BOOL* );
HRESULT put_Multiprocessing(VARIANT_BOOL);
```

**C#**

```
public bool Multiprocessing { get; set; }
```

**Visual Basic**

```
Public Multiprocessing As Boolean
```

**Description**

Flag activating parallel processing - multiprocessing.

**Version**

Available since version 8.5.

## II.2.4 OpenGL

**C++**

```
HRESULT get_OpenGL(VARIANT_BOOL* );
```

```
HRESULT put_OpenGL(VARIANT_BOOL);
```

**C#**

```
public bool OpenGL { get; set; }
```

**Visual Basic**

```
Public OpenGL As Boolean
```

**Description**

Flag that enables switching on/off OpenGL graphics.

**Version**

Available since version 5.5.

## II.3 IRobotPreferences Methods

The methods of the IRobotPreferences class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
光环	GetDirectory ( <a href="#">see page 1536</a> )	The function returns full path to the desired Robot folder. .
光环	GetLanguage ( <a href="#">see page 1536</a> )	Function returns the current setting of the specified language. Code (number) of the appropriate country is returned (e.g. 33 for France).

### II.3.1 GetDirectory

**C++**

```
HRESULT GetDirectory(IRobotDirectory _robot_dir, BSTR* ret);
```

**C#**

```
public String GetDirectory(IRobotDirectory _robot_dir);
```

**Visual Basic**

```
Public Function GetDirectory(_robot_dir As IRobotDirectory) As String
```

**Description**

The function returns full path to the desired Robot folder. .

### II.3.2 GetLanguage

**C++**

```
HRESULT GetLanguage(IRobotLanguage _lang_id, int* ret);
```

**C#**

```
public int GetLanguage(IRobotLanguage _lang_id);
```

**Visual Basic**

```
Public Function GetLanguage(_lang_id As IRobotLanguage) As int
```

**Description**

Function returns the current setting of the specified language. Code (number) of the appropriate country is returned (e.g. 33 for France).

## III IRobotDirectory

### C++

```
enum IRobotDirectory;
```

### C#

```
public enum IRobotDirectory;
```

### Visual Basic

```
Public Enum IRobotDirectory
```

### Members

Members	Description
I_D_EXE = 1	Folder with executable files.
I_D_RES = 2	Folder with resources.
I_D_CONF = 3	Folder with configuration files.
I_D_USER_CONF = 4	Folder with user's configuration files .
I_D_TEMPLATE = 5	Folder with template files .
I_D_HELP = 6	Folder with help files.
I_D_USER_OUTPUT = 7	User's target (start-up) folder .
I_D_MAIN = 8	The main Robot folder .
I_D_USER_MAIN = 9	Main user folder.
I_D_USER_PROJECTS = 10	Project folder.
I_D_USER_TEMPLATE = 11	Available since version 3.

### Description

The set of identifiers representing folders used by Robot program.

## IV IRobotLanguage

### C++

```
enum IRobotLanguage;
```

### C#

```
public enum IRobotLanguage;
```

### Visual Basic

```
Public Enum IRobotLanguage
```

### Members

Members	Description
I_L_WORK = 1	Interface language.
I_L_PRINTOUT = 2	Printout language.
I_L_REGIONAL = 3	Regional language.

## V IRobotQuitOption

### C++

```
enum IRobotQuitOption;
```

### C#

```
public enum IRobotQuitOption;
```

### Visual Basic

```
Public Enum IRobotQuitOption
```

### Members

Members	Description
I_QO_DISCARD_CHANGES = 0	Changes made in the project will not be saved Available since version 1.7.
I_QO_PROMPT_TO_SAVE_CHANGES = 1	If changes that require saving to file have been made in the project, the program will display a message asking if the changes are to be saved Available since version 1.7.
I_QO_SAVE_CHANGES = 2	Changes will be saved automatically Available since version 1.7.

### Description

Set of identifiers describing the manner of exiting the program. .

## VI IRobotDirectoryExtension

### C++

```
enum IRobotDirectoryExtension;
```

### C#

```
public enum IRobotDirectoryExtension;
```

### Visual Basic

```
Public Enum IRobotDirectoryExtension
```

### Members

Members	Description
I_DE_TEMPLATE = 51	Extension for the TEMPLATE folder. Available since version 3.5.
I_DE_USER_CONF = 52	Available since version 3.5.
I_DE_CONF = 53	Available since version 3.5.
I_DE_USER_TEMPLATE = 54	Available since version 3.5.

### Description

Set of identifiers representing extensions for Robot program folders. .

### Version

Available since version 3.5.

## VII IRobotPreferencesEvents

### Class Hierarchy

#### C++

```
interface IRobotPreferencesEvents : IDispatch;
```

#### C#

```
public interface IRobotPreferencesEvents;
```

### Visual Basic

```
Public Interface IRobotPreferencesEvents
```

### Description

Events related to the change of settings for the whole application.

## VII.1 IRobotPreferencesEvents Members

The following tables list the members exposed by IRobotPreferencesEvents.

### Public Methods

	Name	Description
💡	OnDialogOK (see page 1539)	Event generated after accepting changes made in the program settings dialog box by the user.

## VII.2 IRobotPreferencesEvents Methods

The methods of the IRobotPreferencesEvents class are listed here.

### Public Methods

	Name	Description
💡	OnDialogOK (see page 1539)	Event generated after accepting changes made in the program settings dialog box by the user.

### VII.2.1 OnDialogOK

#### C++

```
HRESULT OnDialogOK();
```

#### C#

```
public void OnDialogOK();
```

### Visual Basic

```
Public Sub OnDialogOK()
```

### Description

Event generated after accepting changes made in the program settings dialog box by the user.

## VIII IRobotLicenseEntitlement

#### C++

```
enum IRobotLicenseEntitlement;
```

**C#**

```
public enum IRobotLicenseEntitlement;
```

**Visual Basic**

```
Public Enum IRobotLicenseEntitlement
```

**Members**

Members	Description
I_LE_LOCAL_SOLVE = 0	Entitlement enabling to perform calculations on the local machine. Available since version 13.4.
I_LE_CLOUD_SOLVE = 1	Entitlement enabling to perform calculations in the calculation cloud. Available since version 13.4.

**Version**

Available since version 13.4.

## IX IRobotLicenseEntitlementStatus

**C++**

```
enum IRobotLicenseEntitlementStatus;
```

**C#**

```
public enum IRobotLicenseEntitlementStatus;
```

**Visual Basic**

```
Public Enum IRobotLicenseEntitlementStatus
```

**Members**

Members	Description
ILES_ENTITLED = 0	Available since version 13.4.
ILES_NOT_ENTITLED = 1	No entitlement. Available since version 13.4.
ILES_UNKNOWN = 2	It is impossible to determine the possessed entitlements because there are problems with a network connection. Available since version 13.4.
ILES_NOT_SIGNED_IN = 3	It is impossible to determine the possessed entitlements because the user is not logged in. Available since version 14.

**Version**

Available since version 13.4.

## X IRobotCalculationsType

**C++**

```
enum IRobotCalculationsType;
```

**C#**

```
public enum IRobotCalculationsType;
```

## Visual Basic

```
Public Enum IRobotCalculationsType
```

## Members

Members	Description
I_CT_CALCULATE_LOCALLY = 0	Calculations will be performed locally on the user machine. Available since version 13.4.
I_CT_CALCULATE_IN_CLOUD = 1	Calculations will be performed in the calculation cloud. Available since version 13.4.

## Version

Available since version 13.4.

# Add-ins Manager

## Interfaces

	Name	Description
▪	IRobotAddIn (see page 1541)	Definition of the Robot program extension. This interface must implement each component that is integrated with the Robot program and that extends its functionality. .
▪	IRobotAddInMngr (see page 1544)	Add-ins Manager keeps tracks to all add-ins currently registered in Robot.
▪	IRobotCmdList (see page 1545)	List of information about commands.
▪	IRobotCmdInfo (see page 1547)	Information about the command and the menu option linked with it.
▪	IRobotAddInRegistrar (see page 1549)	Information about each Robot program extension must be saved as an appropriate position in the system register. RobotAddInRegistrar facilitates the process of extension registration.

## IIRobotAddIn

### Class Hierarchy

#### C++

```
interface IRobotAddIn : IDispatch;
```

#### C#

```
public interface IRobotAddIn;
```

## Visual Basic

```
Public Interface IRobotAddIn
```

### Description

Definition of the Robot program extension. This interface must implement each component that is integrated with the Robot program and that extends its functionality. .

## I.1 IIRobotAddIn Members

The following tables list the members exposed by IIRobotAddIn.

## Public Methods

	Name	Description
💡	Connect (🔗 see page 1542)	Function will be called up by the Robot application when the extension is connected to the program: for the first time - upon user's request or during first activation of the program after installing the extension; for the next time - while activating the Robot program again. The extension should remember the access interface to the Robot application (RobotApplication) so that the extension can use its functionality. The extension, while being connected, is ascribed its identifier which should be remembered since it will be needed during subsequent communication with the Robot program. While connecting the extension to the application for... more (🔗 see page 1542)
💡	Disconnect (🔗 see page 1543)	This function will be called up before the extension is disconnected with the Robot program (e.g. while exiting the application or upon user's request who wants no longer to use the extension). If there is any reason for the extension not to be disconnected with the program at the given moment, the function should return zero value (False). .
💡	DoCommand (🔗 see page 1543)	Function will be called up by the Robot program if a user selects the menu option linked with the command supported by this extension. .
💡	GetExpectedVersion (🔗 see page 1543)	Function should return the number of the RobotOS model version which is used by the extension. It will enable warning a user in a situation when the extension that is being installed by the user expects a newer Robot program version than the one currently used by the user. .
💡	InstallCommands (🔗 see page 1544)	Function will be called up by the application to obtain information about the commands that are made available by the extension. List specified by the parameter should be filled out with the information about all the commands defined by the extension. If the operation is performed successfully, the function should return a value different from zero (True). .

## I.2 IRobotAddIn Methods

The methods of the IRobotAddIn class are listed here.

### Public Methods

	Name	Description
💡	Connect (🔗 see page 1542)	Function will be called up by the Robot application when the extension is connected to the program: for the first time - upon user's request or during first activation of the program after installing the extension; for the next time - while activating the Robot program again. The extension should remember the access interface to the Robot application (RobotApplication) so that the extension can use its functionality. The extension, while being connected, is ascribed its identifier which should be remembered since it will be needed during subsequent communication with the Robot program. While connecting the extension to the application for... more (🔗 see page 1542)
💡	Disconnect (🔗 see page 1543)	This function will be called up before the extension is disconnected with the Robot program (e.g. while exiting the application or upon user's request who wants no longer to use the extension). If there is any reason for the extension not to be disconnected with the program at the given moment, the function should return zero value (False). .
💡	DoCommand (🔗 see page 1543)	Function will be called up by the Robot program if a user selects the menu option linked with the command supported by this extension. .
💡	GetExpectedVersion (🔗 see page 1543)	Function should return the number of the RobotOS model version which is used by the extension. It will enable warning a user in a situation when the extension that is being installed by the user expects a newer Robot program version than the one currently used by the user. .

	InstallCommands (see page 1544)	Function will be called up by the application to obtain information about the commands that are made available by the extension. List specified by the parameter should be filled out with the information about all the commands defined by the extension. If the operation is performed successfully, the function should return a value different from zero (True). .
-----------------------------------------------------------------------------------	---------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## I.2.1 Connect

### C++

```
HRESULT Connect(IRobotApplication _robot_app, long _add_in_id, VARIANT_BOOL _first_time,
VARIANT_BOOL* ret);
```

### C#

```
public bool Connect(IRobotApplication _robot_app, long _add_in_id, bool _first_time);
```

### Visual Basic

```
Public Function Connect(_robot_app As IRobotApplication, _add_in_id As long, _first_time As Boolean) As Boolean
```

### Description

Function wil be called up by the Robot application when the extension is connected to the program: for the first time - upon user's request or during first activation of the program after installing the extension; for the next time - while activating the Robot program again. The extension should remember the access interface to the Robot application (RobotApplication) so that the extension can use its functionality. The extension, while being connected, is ascribed its identifier which should be remembered since it will be needed during subsequent communication with the Robot program. While connecting the extension to the application for the first time it is possible to perform additional initializing operations. If all is performed correctly, then function should return a value different from zero (True).

## I.2.2 Disconnect

### C++

```
HRESULT Disconnect(VARIANT_BOOL* ret);
```

### C#

```
public bool Disconnect();
```

### Visual Basic

```
Public Function Disconnect() As Boolean
```

### Description

This function will be called up before the extension is disconnected with the Robot program (e.g. while exiting the application or upon user's request who wants no longer to use the extension). If there is any reason for the extension not to be disconnected with the program at the given moment, the function should return zero value (False). .

## I.2.3 DoCommand

### C++

```
HRESULT DoCommand(long _cmd_id);
```

### C#

```
public void DoCommand(long _cmd_id);
```

**Visual Basic**

```
Public Sub DoCommand(_cmd_id As long)
```

**Description**

Function will be called up by the Robot program if a user selects the menu option linked with the command supported by this extension. .

**I.2.4 GetExpectedVersion****C++**

```
HRESULT GetExpectedVersion(double* ret);
```

**C#**

```
public double GetExpectedVersion();
```

**Visual Basic**

```
Public Function GetExpectedVersion() As double
```

**Description**

Function should return the number of the RobotOS model version which is used by the extension. It will enable warning a user in a situation when the extension that is being installed by the user expects a newer Robot program version than the one currently used by the user. .

**I.2.5 InstallCommands****C++**

```
HRESULT InstallCommands(IRobotCmdList* _cmd_list, VARIANT_BOOL* ret);
```

**C#**

```
public bool InstallCommands(IRobotCmdList _cmd_list);
```

**Visual Basic**

```
Public Function InstallCommands(ByRef _cmd_list As IRobotCmdList) As Boolean
```

**Description**

Function will be called up by the application to obtain information about the commands that are made available by the extension. List specified by the parameter should be filled out with the information about all the commands defined by the extension. If the operation is performed successfully, the function should return a value different from zero (True). .

**II IRobotAddInMngr****Class Hierarchy****C++**

```
interface IRobotAddInMngr : IDispatch;
```

**C#**

```
public interface IRobotAddInMngr;
```

**Visual Basic**

```
Public Interface IRobotAddInMngr
```

## Description

Add-ins Manager keeps tracks to all add-ins currently registered in Robot.

## II.1 IRobotAddInMngr Members

The following tables list the members exposed by IRobotAddInMngr.

### Public Methods

	Name	Description
	InstallCommand ( <a href="#">see page 1545</a> )	

## II.2 IRobotAddInMngr Methods

The methods of the IRobotAddInMngr class are listed here.

### Public Methods

	Name	Description
	InstallCommand ( <a href="#">see page 1545</a> )	

### II.2.1 InstallCommand

#### C++

```
HRESULT InstallCommand(long _add_in_id, long _cmd_id, BSTR _cmd_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool InstallCommand(long _add_in_id, long _cmd_id, String _cmd_name);
```

#### Visual Basic

```
Public Function InstallCommand(_add_in_id As long, _cmd_id As long, _cmd_name As String) As Boolean
```

## III IRobotCmdList

### Class Hierarchy

#### C++

```
interface IRobotCmdList : IDispatch;
```

#### C#

```
public interface IRobotCmdList;
```

#### Visual Basic

```
Public Interface IRobotCmdList
```

### Description

List of information about commands.

## III.1 IRobotCmdList Members

The following tables list the members exposed by IRobotCmdList.

## Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1546</a> )	Number of elements of the list Available since version 1.7.

## Public Methods

	Name	Description
◆	Get ( <a href="#">see page 1546</a> )	Function returns the specified list element. List elements are indexed from 1 to Count ( <a href="#">see page 1546</a> ). Available since version 1.7.
◆	New ( <a href="#">see page 1546</a> )	Function creates a new list element and adds it at the end. The index of the element on the list is returned. Available since version 1.7.

## III.2 IRobotCmdList Fields

The fields of the IRobotCmdList class are listed here.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1546</a> )	Number of elements of the list Available since version 1.7.

### III.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of elements of the list Available since version 1.7.

## III.3 IRobotCmdList Methods

The methods of the IRobotCmdList class are listed here.

### Public Methods

	Name	Description
◆	Get ( <a href="#">see page 1546</a> )	Function returns the specified list element. List elements are indexed from 1 to Count ( <a href="#">see page 1546</a> ). Available since version 1.7.
◆	New ( <a href="#">see page 1546</a> )	Function creates a new list element and adds it at the end. The index of the element on the list is returned. Available since version 1.7.

### III.3.1 Get

#### C++

```
HRESULT Get(long _idx, IRobotCmdInfo** ret);
```

#### C#

```
public IRobotCmdInfo Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As IRobotCmdInfo
```

## Description

Function returns the specified list element. List elements are indexed from 1 to Count (see page 1546). Available since version 1.7.

### III.3.2 New

#### C++

```
HRESULT New(long _cmd_id, BSTR _cmd_name, long* ret);
```

#### C#

```
public long New(long _cmd_id, String _cmd_name);
```

#### Visual Basic

```
Public Function New(_cmd_id As long, _cmd_name As String) As long
```

## Description

Function creates a new list element and adds it at the end. The index of the element on the list is returned. Available since version 1.7.

## IV IRobotCmdInfo

### Class Hierarchy

#### C++

```
interface IRobotCmdInfo : IDispatch;
```

#### C#

```
public interface IRobotCmdInfo;
```

#### Visual Basic

```
Public Interface IRobotCmdInfo
```

## Description

Information about the command and the menu option linked with it.

### IV.1 IRobotCmdInfo Members

The following tables list the members exposed by IRobotCmdInfo.

#### Public Fields

	Name	Description
❖	Id (see page 1547)	Command identifier Available since version 1.7.
❖	MenuChecked (see page 1548)	Indicator of marking the menu option linked with the command 0 - not marked 1 - marked 2 - intermediate state Available since version 1.7.
❖	MenuEnabled (see page 1548)	Flag indicating if the menu option linked with the command is available Available since version 1.7.
❖	Name (see page 1548)	Name of the menu option linked with the command Available since version 1.7.

## IV.2 IRobotCmdInfo Fields

The fields of the IRobotCmdInfo class are listed here.

### Public Fields

	Name	Description
◆	Id (see page 1547)	Command identifier Available since version 1.7.
◆	MenuChecked (see page 1548)	Indicator of marking the menu option linked with the command 0 - not marked 1 - marked 2 - intermediate state Available since version 1.7.
◆	MenuEnabled (see page 1548)	Flag indicating if the menu option linked with the command is available Available since version 1.7.
◆	Name (see page 1548)	Name of the menu option linked with the command Available since version 1.7.

### IV.2.1 Id

#### C++

```
HRESULT get_Id(long* );
HRESULT put_Id(long);
```

#### C#

```
public long Id { get; set; }
```

#### Visual Basic

```
Public Id As long
```

#### Description

Command identifier Available since version 1.7.

### IV.2.2 MenuChecked

#### C++

```
HRESULT get_MenuChecked(long* );
HRESULT put_MenuChecked(long);
```

#### C#

```
public long MenuChecked { get; set; }
```

#### Visual Basic

```
Public MenuChecked As long
```

#### Description

Indicator of marking the menu option linked with the command 0 - not marked 1 - marked 2 - intermediate state

Available since version 1.7.

### IV.2.3 MenuEnabled

#### C++

```
HRESULT get_MenuEnabled(VARIANT_BOOL* );
HRESULT put_MenuEnabled(VARIANT_BOOL);
```

#### C#

```
public bool MenuEnabled { get; set; }
```

**Visual Basic**

```
Public MenuEnabled As Boolean
```

**Description**

Flag indicating if the menu option linked with the command is available Available since version 1.7.

**IV.2.4 Name****C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Name of the menu option linked with the command Available since version 1.7.

**V IRobotAddInRegistrar****Class Hierarchy****C++**

```
interface IRobotAddInRegistrar : IDispatch;
```

**C#**

```
public interface IRobotAddInRegistrar;
```

**Visual Basic**

```
Public Interface IRobotAddInRegistrar
```

**Description**

Information about each Robot program extension must be saved as an appropriate position in the system register. RobotAddInRegistrar facilitates the process of extension registration.

**V.1 IRobotAddInRegistrar Members**

The following tables list the members exposed by IRobotAddInRegistrar.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Guid (↗ see page 1550)	GUID identifier of the main extension component (presented interchangeably with ProgId) Available since version 1.7.
◆	ProductName (↗ see page 1550)	Extension name Available since version 1.7.
◆	ProgId (↗ see page 1550)	Identifier of the main extension component (presented interchangeably with Guid (↗ see page 1550)) Available since version 1.7.
◆	ProviderName (↗ see page 1551)	Name of the extension supplier Available since version 1.7.

## Public Methods

	Name	Description
💡	InstallMenu (see page 1551)	Function adds menu with the specified name to all Robot program views. The commands specified in the list will be the menu options. They will be supported by the extension whose ProgId or Guid (see page 1550) have been set earlier. If identifier of the extension supporting commands from the list is not specified, the function returns zero value (False). Available since version 1.7.
💡	Register (see page 1552)	Function saves information about the extension to the register. If registration is performed successfully, a value different from zero is returned (True). Registration failure may result from lack of settings concerning supplier name or product as well as identifier (Guid (see page 1550) or ProgId). Available since version 1.7.
💡	Unregister (see page 1552)	Function deletes information about the extension from the register. Available since version 1.7.

## V.2 IRobotAddInRegistrar Fields

The fields of the IRobotAddInRegistrar class are listed here.

### Public Fields

	Name	Description
❖	Guid (see page 1550)	GUID identifier of the main extension component (presented interchangeably with ProgId) Available since version 1.7.
❖	ProductName (see page 1550)	Extension name Available since version 1.7.
❖	ProgId (see page 1550)	Identifier of the main extension component (presented interchangeably with Guid (see page 1550)) Available since version 1.7.
❖	ProviderName (see page 1551)	Name of the extension supplier Available since version 1.7.

### V.2.1 Guid

#### C++

```
HRESULT get_Guid(BSTR* );
HRESULT put_Guid(BSTR);
```

#### C#

```
public String Guid { get; set; }
```

#### Visual Basic

```
Public Guid As String
```

#### Description

GUID identifier of the main extension component (presented interchangeably with ProgId) Available since version 1.7.

### V.2.2 ProductName

#### C++

```
HRESULT get_ProductName(BSTR* );
HRESULT put_ProductName(BSTR);
```

#### C#

```
public String ProductName { get; set; }
```

#### Visual Basic

```
Public ProductName As String
```

**Description**

Extension name Available since version 1.7.

**V.2.3 ProgId****C++**

```
HRESULT get_ProgId(BSTR* );
HRESULT put_ProgId(BSTR);
```

**C#**

```
public String ProgId { get; set; }
```

**Visual Basic**

```
Public ProgId As String
```

**Description**

Identifier of the main extension component (presented interchangeably with Guid (see page 1550)) Available since version 1.7.

**V.2.4 ProviderName****C++**

```
HRESULT get_ProviderName(BSTR* );
HRESULT put_ProviderName(BSTR);
```

**C#**

```
public String ProviderName { get; set; }
```

**Visual Basic**

```
Public ProviderName As String
```

**Description**

Name of the extension supplier Available since version 1.7.

**V.3 IRobotAddInRegistrar Methods**

The methods of the IRobotAddInRegistrar class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	InstallMenu (see page 1551)	Function adds menu with the specified name to all Robot program views. The commands specified in the list will be the menu options. They will be supported by the extension whose ProgId or Guid (see page 1550) have been set earlier. If identifier of the extension supporting commands from the list is not specified, the function returns zero value (False). Available since version 1.7.
💡	Register (see page 1552)	Function saves information about the extension to the register. If registration is performed successfully, a value different from zero is returned (True). Registration failure may result from lack of settings concerning supplier name or product as well as identifier (Guid (see page 1550) or ProgId). Available since version 1.7.
💡	Unregister (see page 1552)	Function deletes information about the extension from the register. Available since version 1.7.

### V.3.1 InstallMenu

C++

```
HRESULT InstallMenu(BSTR _menu_name, IRobotCmdList* _options, VARIANT_BOOL* ret);
```

C#

```
public bool InstallMenu(String _menu_name, IRobotCmdList _options);
```

Visual Basic

```
Public Function InstallMenu(_menu_name As String, ByRef _options As IRobotCmdList) As Boolean
```

Description

Function adds menu with the specified name to all Robot program views. The commands specified in the list will be the menu options. They will be supported by the extension whose ProgId or Guid (see page 1550) have been set earlier. If identifier of the extension supporting commands from the list is not specified, the function returns zero value (False). Available since version 1.7.

### V.3.2 Register

C++

```
HRESULT Register(VARIANT_BOOL* ret);
```

C#

```
public bool Register();
```

Visual Basic

```
Public Function Register() As Boolean
```

Description

Function saves information about the extension to the register. If registration is performed successfully, a value different from zero is returned (True). Registration failure may result from lack of settings concerning supplier name or product as well as identifier (Guid (see page 1550) or ProgId). Available since version 1.7.

### V.3.3 Unregister

C++

```
HRESULT Unregister();
```

C#

```
public void Unregister();
```

Visual Basic

```
Public Sub Unregister()
```

Description

Function deletes information about the extension from the register. Available since version 1.7.

# Calculation module

## Enumerations

	Name	Description
	IRobotEquationSolvingMethod (see page 1559)	Available methods of equation set solution.
	IRobotStructureAutoVerificationType (see page 1564)	Available types of automatic structure verification.
	IRobotSparseMSolverMethod (see page 1565)	
	IRobotIterativePreconditionerType (see page 1566)	
	IRobotIterativeSolverMethod (see page 1570)	
	IRobotIterativeSolverMemoryUsage (see page 1571)	
	IRobotStructureAnalysisModalParticipationCoeff (see page 1571)	Available methods of participation coefficient calculation.
	IRobotSeismicResultsPanelDirection (see page 1576)	
	IRobotCalcMessageSource (see page 1585)	
	IRobotCalcMessageSeverityLevel (see page 1585)	
	IRobotCalculationStatus (see page 1586)	
	IRobotCalculationMode (see page 1586)	Available methods of performing calculations.

## Interfaces

	Name	Description
	IRobotCalcEngine (see page 1553)	Interface representing the calculation module of Robot program. .
	IRobotStructureAnalysisParams (see page 1560)	Parameters of structure analysis.
	IRobotSparseMSolverParams (see page 1565)	
	IRobotIterativeSolverParams (see page 1566)	
	IRobotSeismicResultsSaveParams (see page 1571)	Parameters of saving quadratic combination results for seismic analysis in a project.
	IRobotModelGenerationParams (see page 1577)	Generation parameters for structure calculation model.
	IRobotBucklingDeformationParams (see page 1581)	Parameters of buckling deformation.
	IRobotCalcEngineEvents (see page 1584)	Events generated by the RobotCalcEngine calculation module.

## IRobotCalcEngine

### Class Hierarchy

### C++

```
interface IRobotCalcEngine : IDispatch;
```

**C#**

```
public interface IRobotCalcEngine;
```

**Visual Basic**

```
Public Interface IRobotCalcEngine
```

**Description**

Interface representing the calculation module of Robot program. .

## I.1 IRobotCalcEngine Members

The following tables list the members exposed by IRobotCalcEngine.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AnalysisParams (see page 1555)	Parameters of structure analysis.
◆	AutoFreezeResults (see page 1555)	Flag enforcing automatic freezing of results after calculations .
◆	AutoGenerateModel (see page 1555)	Flag steering automatic model generation that directly precedes structure calculations.
◆	BucklingDeformation (see page 1556)	Buckling deformation.
◆	DAM (see page 1556)	.
◆	GenerationParams (see page 1556)	Parameters of calculation model generation.
◆	SaveResultsInExternalFile (see page 1556)	Flag enforcing result saving in the external file *.RT_.
◆	SeismicResultsSaveParams (see page 1557)	Saving parameters for seismic analysis results .
◆	StatusWindowParent (see page 1557)	Handle to parent window for the calculation status window .
◆	UseStatusWindow (see page 1557)	Flag that steers the status window display while performing time-consuming calculations .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Calculate (see page 1558)	The function recalculates the entire structure. If there appeared an error during calculations, the function returns zero value (False), otherwise it returns a non-zero value (True) .
◆	CalculateEx (see page 1558)	The functions performs structure analysis. If the calculations succeeded and results are available the status I_CS_COMPLETED is returned. Otherwise, the appropriate status with info about reasons of error(s) is returned.
◆	GenerateModel (see page 1559)	Function generates a structure calculation model. .
◆	StopCalculation (see page 1559)	Function causes interrupting the currently performed calculations.

## I.2 IRobotCalcEngine Fields

The fields of the IRobotCalcEngine class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AnalysisParams (see page 1555)	Parameters of structure analysis.

◆	AutoFreezeResults ( [ see page 1555) )	Flag enforcing automatic freezing of results after calculations .
◆	AutoGenerateModel ( [ see page 1555) )	Flag steering automatic model generation that directly precedes structure calculations.
◆	BucklingDeformation ( [ see page 1556) )	Buckling deformation.
◆	DAM ( [ see page 1556) )	.
◆	GenerationParams ( [ see page 1556) )	Parameters of calculation model generation.
◆	SaveResultsInExternalFile ( [ see page 1556) )	Flag enforcing result saving in the external file *.RT_.
◆	SeismicResultsSaveParams ( [ see page 1557) )	Saving parameters for seismic analysis results .
◆	StatusWindowParent ( [ see page 1557) )	Handle to parent window for the calculation status window .
◆	UseStatusWindow ( [ see page 1557) )	Flag that steers the status window display while performing time-consuming calculations .

## I.2.1 AnalysisParams

### C++

```
HRESULT get_AnalysisParams(IRobotStructureAnalysisParams**);
```

### C#

```
public IRobotStructureAnalysisParams AnalysisParams { get; }
```

### Visual Basic

```
Public ReadOnly AnalysisParams As IRobotStructureAnalysisParams
```

### Description

Parameters of structure analysis.

### Version

Available since version 3.

## I.2.2 AutoFreezeResults

### C++

```
HRESULT get_AutoFreezeResults(VARIANT_BOOL*);  
HRESULT put_AutoFreezeResults(VARIANT_BOOL);
```

### C#

```
public bool AutoFreezeResults { get; set; }
```

### Visual Basic

```
Public AutoFreezeResults As Boolean
```

### Description

Flag enforcing automatic freezing of results after calculations .

### Version

Available since version 3.

### I.2.3 AutoGenerateModel

#### C++

```
HRESULT get_AutoGenerateModel(VARIANT_BOOL* );
HRESULT put_AutoGenerateModel(VARIANT_BOOL);
```

#### C#

```
public bool AutoGenerateModel { get; set; }
```

#### Visual Basic

```
Public AutoGenerateModel As Boolean
```

#### Description

Flag steering automatic model generation that directly precedes structure calculations.

#### Version

Available since version 2.5.

### I.2.4 BucklingDeformation

#### C++

```
HRESULT get_BucklingDeformation(IRobotBucklingDeformationParams** );
```

#### C#

```
public IRobotBucklingDeformationParams BucklingDeformation { get; }
```

#### Visual Basic

```
Public ReadOnly BucklingDeformation As IRobotBucklingDeformationParams
```

#### Description

Buckling deformation.

#### Version

Available since version 3.

### I.2.5 DAM

#### C++

```
HRESULT get_DAM(IRobotDAMCalcModule** );
```

#### C#

```
public IRobotDAMCalcModule DAM { get; }
```

#### Visual Basic

```
Public ReadOnly DAM As IRobotDAMCalcModule
```

#### Description

#### Version

Available since version 15.

### I.2.6 GenerationParams

#### C++

```
HRESULT get_GenerationParams(IRobotModelGenerationParams** );
```

**C#**

```
public IRobotModelGenerationParams GenerationParams { get; }
```

**Visual Basic**

```
Public ReadOnly GenerationParams As IRobotModelGenerationParams
```

**Description**

Parameters of calculation model generation.

**Version**

Available since version 3.

**I.2.7 SaveResultsInExternalFile****C++**

```
HRESULT get_SaveResultsInExternalFile(VARIANT_BOOL* );
HRESULT put_SaveResultsInExternalFile(VARIANT_BOOL);
```

**C#**

```
public bool SaveResultsInExternalFile { get; set; }
```

**Visual Basic**

```
Public SaveResultsInExternalFile As Boolean
```

**Description**

Flag enforcing result saving in the external file \*.RT\_.

**Version**

Available since version 3.

**I.2.8 SeismicResultsSaveParams****C++**

```
HRESULT get_SeismicResultsSaveParams( IRobotSeismicResultsSaveParams** );
```

**C#**

```
public IRobotSeismicResultsSaveParams SeismicResultsSaveParams { get; }
```

**Visual Basic**

```
Public ReadOnly SeismicResultsSaveParams As IRobotSeismicResultsSaveParams
```

**Description**

Saving parameters for seismic analysis results .

**Version**

Available since version 3.

**I.2.9 StatusWindowParent****C++**

```
HRESULT get_StatusWindowParent(long* );
HRESULT put_StatusWindowParent(long);
```

**C#**

```
public long StatusWindowParent { get; set; }
```

**Visual Basic**

```
Public StatusWindowParent As long
```

**Description**

Handle to parent window for the calculation status window .

**Version**

Available since version 5.5.

**I.2.10 UseStatusWindow****C++**

```
HRESULT get_UseStatusWindow(VARIANT_BOOL* );
HRESULT put_UseStatusWindow(VARIANT_BOOL);
```

**C#**

```
public bool UseStatusWindow { get; set; }
```

**Visual Basic**

```
Public UseStatusWindow As Boolean
```

**Description**

Flag that steers the status window display while performing time-consuming calculations .

**Version**

Available since version 5.5.

**I.3 IRobotCalcEngine Methods**

The methods of the IRobotCalcEngine class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Calculate ( [ see page 1558])	The function recalculates the entire structure. If there appeared an error during calculations, the function returns zero value (False), otherwise it returns a non-zero value (True) .
💡	CalculateEx ( [ see page 1558])	The functions performs structure analysis. If the calculations succeeded and results are available the status I_CS_COMPLETED is returned. Otherwise, the appropriate status with info about reasons of error(s) is returned.
💡	GenerateModel ( [ see page 1559])	Function generates a structure calculation model. .
💡	StopCalculation ( [ see page 1559])	Function causes interrupting the currently performed calculations.

**I.3.1 Calculate****C++**

```
HRESULT Calculate(VARIANT_BOOL* ret);
```

**C#**

```
public bool Calculate();
```

**Visual Basic**

```
Public Function Calculate() As Boolean
```

## Description

The function recalculates the entire structure. If there appeared an error during calculations, the function returns zero value (False), otherwise it returns a non-zero value (True) .

### I.3.2 CalculateEx

#### C++

```
HRESULT CalculateEx(IRobotCalculationMode _calc_mode, IRobotCalculationStatus* ret);
```

#### C#

```
public IRobotCalculationStatus CalculateEx(IRobotCalculationMode _calc_mode);
```

#### Visual Basic

```
Public Function CalculateEx(_calc_mode As IRobotCalculationMode) As IRobotCalculationStatus
```

#### Description

The functions performs structure analysis. If the calculations succeeded and results are available the status I\_CS\_COMPLETED is returned. Otherwise, the appropriate status with info about reasons of error(s) is returned.

#### Version

Available since version 13.4.

### I.3.3 GenerateModel

#### C++

```
HRESULT GenerateModel();
```

#### C#

```
public void GenerateModel();
```

#### Visual Basic

```
Public Sub GenerateModel()
```

#### Description

Function generates a structure calculation model. .

#### Version

Available since version 3.5.

### I.3.4 StopCalculation

#### C++

```
HRESULT StopCalculation();
```

#### C#

```
public void StopCalculation();
```

#### Visual Basic

```
Public Sub StopCalculation()
```

#### Description

Function causes interrupting the currently performed calculations.

**Version**

Available since version 13.4.

## II IRobotEquationSolvingMethod

**C++**

```
enum IRobotEquationSolvingMethod;
```

**C#**

```
public enum IRobotEquationSolvingMethod;
```

**Visual Basic**

```
Public Enum IRobotEquationSolvingMethod
```

**Members**

Members	Description
I_ESM_FRONTAL = 0	Available since version 3.
I_ESM_SKYLINE = 1	Available since version 3.
I_ESM_SPARSE = 4	Available since version 3.
I_ESM_SPARSE_M = 5	Available since version 3.
I_ESM_ITERATIVE = 2	Available since version 3.
I_ESM_AUTO = 3	Available since version 3.
I_ESM_MULTI_THREADED = 7	Available since version 12.

**Description**

Available methods of equation set solution.

**Version**

Available since version 3.

## III IRobotStructureAnalysisParams

**Class Hierarchy****C++**

```
interface IRobotStructureAnalysisParams : IDispatch;
```

**C#**

```
public interface IRobotStructureAnalysisParams;
```

**Visual Basic**

```
Public Interface IRobotStructureAnalysisParams
```

**Description**

Parameters of structure analysis.

**Version**

Available since version 3.

## III.1 IRobotStructureAnalysisParams Members

The following tables list the members exposed by IRobotStructureAnalysisParams.

### Public Fields

	Name	Description
◆	AutoBarMerging ( <a href="#">see page 1561</a> )	Automatic bar merging during import of geometry.
◆	AutoVerification ( <a href="#">see page 1561</a> )	Selected type of automatic structure verification.
◆	DSCAlgorithm ( <a href="#">see page 1562</a> )	DSC algorithm.
◆	EquationSolvingMethod ( <a href="#">see page 1562</a> )	Selected method of equation set solution .
◆	FictitiousRigidityCoeff ( <a href="#">see page 1562</a> )	Fictitious rigidity coefficient.
◆	IgnoreWarnings ( <a href="#">see page 1563</a> )	
◆	IterativeParams ( <a href="#">see page 1563</a> )	
◆	ModalParticipationCoeff ( <a href="#">see page 1563</a> )	Method of participation coefficient calculation.
◆	RLINKElements ( <a href="#">see page 1564</a> )	
◆	SparseMParams ( <a href="#">see page 1564</a> )	

## III.2 IRobotStructureAnalysisParams Fields

The fields of the IRobotStructureAnalysisParams class are listed here.

### Public Fields

	Name	Description
◆	AutoBarMerging ( <a href="#">see page 1561</a> )	Automatic bar merging during import of geometry.
◆	AutoVerification ( <a href="#">see page 1561</a> )	Selected type of automatic structure verification.
◆	DSCAlgorithm ( <a href="#">see page 1562</a> )	DSC algorithm.
◆	EquationSolvingMethod ( <a href="#">see page 1562</a> )	Selected method of equation set solution .
◆	FictitiousRigidityCoeff ( <a href="#">see page 1562</a> )	Fictitious rigidity coefficient.
◆	IgnoreWarnings ( <a href="#">see page 1563</a> )	
◆	IterativeParams ( <a href="#">see page 1563</a> )	
◆	ModalParticipationCoeff ( <a href="#">see page 1563</a> )	Method of participation coefficient calculation.
◆	RLINKElements ( <a href="#">see page 1564</a> )	
◆	SparseMParams ( <a href="#">see page 1564</a> )	

### III.2.1 AutoBarMerging

#### C++

```
HRESULT get_AutoBarMerging(VARIANT_BOOL* );
HRESULT put_AutoBarMerging(VARIANT_BOOL );
```

#### C#

```
public bool AutoBarMerging { get; set; }
```

#### Visual Basic

```
Public AutoBarMerging As Boolean
```

**Description**

Automatic bar merging during import of geometry.

**Version**

Available since version 3.

### III.2.2 AutoVerification

**C++**

```
HRESULT get_AutoVerification(IRobotStructureAutoVerificationType* );
HRESULT put_AutoVerification(IRobotStructureAutoVerificationType);
```

**C#**

```
public IRobotStructureAutoVerificationType AutoVerification { get; set; }
```

**Visual Basic**

```
Public AutoVerification As IRobotStructureAutoVerificationType
```

**Description**

Selected type of automatic structure verification.

**Version**

Available since version 3.

### III.2.3 DSCAlgorithm

**C++**

```
HRESULT get_DSCAlgorithm(VARIANT_BOOL* );
HRESULT put_DSCAlgorithm(VARIANT_BOOL);
```

**C#**

```
public bool DSCAlgorithm { get; set; }
```

**Visual Basic**

```
Public DSCAlgorithm As Boolean
```

**Description**

DSC algorithm.

**Version**

Available since version 3.

### III.2.4 EquationSolvingMethod

**C++**

```
HRESULT get_EquationSolvingMethod(IRobotEquationSolvingMethod* );
HRESULT put_EquationSolvingMethod(IRobotEquationSolvingMethod);
```

**C#**

```
public IRobotEquationSolvingMethod EquationSolvingMethod { get; set; }
```

**Visual Basic**

```
Public EquationSolvingMethod As IRobotEquationSolvingMethod
```

**Description**

Selected method of equation set solution .

**Version**

Available since version 3.

**III.2.5 FictitiousRigidityCoeff****C++**

```
HRESULT get_FictitiousRigidityCoeff(double* );
HRESULT put_FictitiousRigidityCoeff(double);
```

**C#**

```
public double FictitiousRigidityCoeff { get; set; }
```

**Visual Basic**

```
Public FictitiousRigidityCoeff As Double
```

**Description**

Fictitious rigidity coefficient.

**Version**

Available since version 3.

**III.2.6 IgnoreWarnings****C++**

```
HRESULT get_IgnoreWarnings(VARIANT_BOOL* );
HRESULT put_IgnoreWarnings(VARIANT_BOOL);
```

**C#**

```
public bool IgnoreWarnings { get; set; }
```

**Visual Basic**

```
Public IgnoreWarnings As Boolean
```

**Version**

Available since version 3.

**III.2.7 IterativeParams****C++**

```
HRESULT get_IterativeParams(IRobotIterativeSolverParams** );
```

**C#**

```
public IRobotIterativeSolverParams IterativeParams { get; }
```

**Visual Basic**

```
Public ReadOnly IterativeParams As IRobotIterativeSolverParams
```

**Version**

Available since version 3.

**III.2.8 ModalParticipationCoeff****C++**

```
HRESULT get_ModalParticipationCoeff(IRobotStructureAnalysisModalParticipationCoeff* );
HRESULT put_ModalParticipationCoeff(IRobotStructureAnalysisModalParticipationCoeff);
```

**C#**

```
public IRobotStructureAnalysisModalParticipationCoeff ModalParticipationCoeff { get; set; }
```

**Visual Basic**

```
Public ModalParticipationCoeff As IRobotStructureAnalysisModalParticipationCoeff
```

**Description**

Method of participation coefficient calculation.

**Version**

Available since version 3.

### III.2.9 RLINKElements

**C++**

```
HRESULT get_RLINKElements(VARIANT_BOOL* );
HRESULT put_RLINKElements(VARIANT_BOOL );
```

**C#**

```
public bool RLINKElements { get; set; }
```

**Visual Basic**

```
Public RLINKElements As Boolean
```

**Version**

Available since version 11.

### III.2.10 SparseMParams

**C++**

```
HRESULT get_SparseMParams(IRobotSparseMSolverParams** );
```

**C#**

```
public IRobotSparseMSolverParams SparseMParams { get; }
```

**Visual Basic**

```
Public ReadOnly SparseMParams As IRobotSparseMSolverParams
```

**Version**

Available since version 3.

## IV IRobotStructureAutoVerificationType

**C++**

```
enum IRobotStructureAutoVerificationType;
```

**C#**

```
public enum IRobotStructureAutoVerificationType;
```

**Visual Basic**

```
Public Enum IRobotStructureAutoVerificationType
```

## Members

Members	Description
I_SAVT_NONE = 0	Available since version 3.
I_SAVT_ERRORS_ONLY = 1	Available since version 3.
I_SAVT_ERRORS_AND_WARNINGS = 2	Available since version 3.

## Description

Available types of automatic structure verification.

## Version

Available since version 3.

# V IRobotSparseMSolverParams

## Class Hierarchy

### C++

```
interface IRobotSparseMSolverParams : IDispatch;
```

### C#

```
public interface IRobotSparseMSolverParams;
```

## Visual Basic

```
Public Interface IRobotSparseMSolverParams
```

## Version

Available since version 3.

## V.1 IRobotSparseMSolverParams Members

The following tables list the members exposed by IRobotSparseMSolverParams.

### Public Fields

	Name	Description
❖	Method ( <a href="#">see page 1565</a> )	

## V.2 IRobotSparseMSolverParams Fields

The fields of the IRobotSparseMSolverParams class are listed here.

### Public Fields

	Name	Description
❖	Method ( <a href="#">see page 1565</a> )	

### V.2.1 Method

#### C++

```
HRESULT get_Method(IRobotSparseMSolverMethod* );
HRESULT put_Method(IRobotSparseMSolverMethod* );
```

#### C#

```
public IRobotSparseMSolverMethod Method { get; set; }
```

**Visual Basic**

```
Public Method As IRobotSparseMSolverMethod
```

**Version**

Available since version 3.

## VI IRobotSparseMSolverMethod

**C++**

```
enum IRobotSparseMSolverMethod;
```

**C#**

```
public enum IRobotSparseMSolverMethod;
```

**Visual Basic**

```
Public Enum IRobotSparseMSolverMethod
```

**Members**

Members	Description
I_SMSM_NDM = 3	Nested dissection method. Available since version 3.
I_SMSM_MDA = 4	Minimal degrees algorithm (MDA). Available since version 3.

**Version**

Available since version 3.

## VII IRobotIterativePredicitionerType

**C++**

```
enum IRobotIterativePredicitionerType;
```

**C#**

```
public enum IRobotIterativePredicitionerType;
```

**Visual Basic**

```
Public Enum IRobotIterativePredicitionerType
```

**Members**

Members	Description
I_IPT_DIAGONAL = 0	Available since version 3.
I_IPT_GAUSS = 1	Available since version 3.
I_IPT_CHOLESKY = 2	Available since version 3.
I_IPT_ICCF = 3	Available since version 3.

**Version**

Available since version 3.

## VIII IRobotIterativeSolverParams

### Class Hierarchy

#### C++

```
interface IRobotIterativeSolverParams : IDispatch;
```

#### C#

```
public interface IRobotIterativeSolverParams;
```

### Visual Basic

```
Public Interface IRobotIterativeSolverParams
```

### Version

Available since version 3.

## VIII.1 IRobotIterativeSolverParams Members

The following tables list the members exposed by IRobotIterativeSolverParams.

### Public Fields

	Name	Description
◆	AggregationLevelsCount ( <a href="#">see page 1567</a> )	Number of aggregation levels.
◆	AnalyseDiagonale ( <a href="#">see page 1568</a> )	Diagonal analysis.
◆	CalcKMatrix ( <a href="#">see page 1568</a> )	Calculation of the K matrix.
◆	InternalIterationsCount ( <a href="#">see page 1568</a> )	Number of internal iterations.
◆	MemoryUsage ( <a href="#">see page 1569</a> )	
◆	Method ( <a href="#">see page 1569</a> )	
◆	Multilevel ( <a href="#">see page 1569</a> )	
◆	PredconditionerType ( <a href="#">see page 1570</a> )	
◆	Tolerance ( <a href="#">see page 1570</a> )	Tolerance definition.

## VIII.2 IRobotIterativeSolverParams Fields

The fields of the IRobotIterativeSolverParams class are listed here.

### Public Fields

	Name	Description
◆	AggregationLevelsCount ( <a href="#">see page 1567</a> )	Number of aggregation levels.
◆	AnalyseDiagonale ( <a href="#">see page 1568</a> )	Diagonal analysis.
◆	CalcKMatrix ( <a href="#">see page 1568</a> )	Calculation of the K matrix.
◆	InternalIterationsCount ( <a href="#">see page 1568</a> )	Number of internal iterations.
◆	MemoryUsage ( <a href="#">see page 1569</a> )	
◆	Method ( <a href="#">see page 1569</a> )	
◆	Multilevel ( <a href="#">see page 1569</a> )	

◆	PredconditionerType (see page 1570)	
◆	Tolerance (see page 1570)	Tolerance definition.

### VIII.2.1 AggregationLevelsCount

#### C++

```
HRESULT get_AggregationLevelsCount(long* );
HRESULT put_AggregationLevelsCount(long);
```

#### C#

```
public long AggregationLevelsCount { get; set; }
```

#### Visual Basic

```
Public AggregationLevelsCount As Long
```

#### Description

Number of aggregation levels.

#### Version

Available since version 3.

### VIII.2.2 AnalyseDiagonale

#### C++

```
HRESULT get_AnalyseDiagonale(VARIANT_BOOL* );
HRESULT put_AnalyseDiagonale(VARIANT_BOOL);
```

#### C#

```
public bool AnalyseDiagonale { get; set; }
```

#### Visual Basic

```
Public AnalyseDiagonale As Boolean
```

#### Description

Diagonal analysis.

#### Version

Available since version 3.

### VIII.2.3 CalcKMatrix

#### C++

```
HRESULT get_CalcKMatrix(VARIANT_BOOL* );
HRESULT put_CalcKMatrix(VARIANT_BOOL);
```

#### C#

```
public bool CalcKMatrix { get; set; }
```

#### Visual Basic

```
Public CalcKMatrix As Boolean
```

#### Description

Calculation of the K matrix.

#### Version

Available since version 3.

### VIII.2.4 InternalIterationsCount

#### C++

```
HRESULT get_InternalIterationsCount(long* );
HRESULT put_InternalIterationsCount(long);
```

#### C#

```
public long InternalIterationsCount { get; set; }
```

#### Visual Basic

```
Public InternalIterationsCount As Long
```

#### Description

Number of internal iterations.

#### Version

Available since version 3.

### VIII.2.5 MemoryUsage

#### C++

```
HRESULT get_MemoryUsage(IRobotIterativeSolverMemoryUsage* );
HRESULT put_MemoryUsage(IRobotIterativeSolverMemoryUsage);
```

#### C#

```
public IRobotIterativeSolverMemoryUsage MemoryUsage { get; set; }
```

#### Visual Basic

```
Public MemoryUsage As IRobotIterativeSolverMemoryUsage
```

#### Version

Available since version 3.

### VIII.2.6 Method

#### C++

```
HRESULT get_Method(IRobotIterativeSolverMethod* );
HRESULT put_Method(IRobotIterativeSolverMethod);
```

#### C#

```
public IRobotIterativeSolverMethod Method { get; set; }
```

#### Visual Basic

```
Public Method As IRobotIterativeSolverMethod
```

#### Version

Available since version 3.

### VIII.2.7 Multilevel

#### C++

```
HRESULT get_Multilevel(VARIANT_BOOL* );
HRESULT put_Multilevel(VARIANT_BOOL);
```

#### C#

```
public bool Multilevel { get; set; }
```

**Visual Basic**

```
Public Multilevel As Boolean
```

**Version**

Available since version 3.

**VIII.2.8 PredicitionerType****C++**

```
HRESULT get_PredicitionerType(IRobotIterativePredicitionerType* );
HRESULT put_PredicitionerType(IRobotIterativePredicitionerType);
```

**C#**

```
public IRobotIterativePredicitionerType PredicitionerType { get; set; }
```

**Visual Basic**

```
Public PredicitionerType As IRobotIterativePredicitionerType
```

**Version**

Available since version 3.

**VIII.2.9 Tolerance****C++**

```
HRESULT get_Tolerance(double* );
HRESULT put_Tolerance(double);
```

**C#**

```
public double Tolerance { get; set; }
```

**Visual Basic**

```
Public Tolerance As Double
```

**Description**

Tolerance definition.

**Version**

Available since version 3.

**IX IRobotIterativeSolverMethod****C++**

```
enum IRobotIterativeSolverMethod;
```

**C#**

```
public enum IRobotIterativeSolverMethod;
```

**Visual Basic**

```
Public Enum IRobotIterativeSolverMethod
```

**Members**

Members	Description
I_ISM_0 = 0	Available since version 3.
I_ISM_1 = 1	Available since version 3.

I_ISM_2 = 2	Available since version 3.
-------------	----------------------------

#### Version

Available since version 3.

## X IRobotIterativeSolverMemoryUsage

#### C++

```
enum IRobotIterativeSolverMemoryUsage;
```

#### C#

```
public enum IRobotIterativeSolverMemoryUsage;
```

#### Visual Basic

```
Public Enum IRobotIterativeSolverMemoryUsage
```

#### Members

Members	Description
I_ISMU_MIN = 0	Available since version 3.
I_ISMU_1_2 = 2	Available since version 3.
I_ISMU_1_4 = 1	Available since version 3.
I_ISMU_MAX = 3	Available since version 3.

#### Version

Available since version 3.

## XI IRobotStructureAnalysisModalParticipationCoeff

#### C++

```
enum IRobotStructureAnalysisModalParticipationCoeff;
```

#### C#

```
public enum IRobotStructureAnalysisModalParticipationCoeff;
```

#### Visual Basic

```
Public Enum IRobotStructureAnalysisModalParticipationCoeff
```

#### Members

Members	Description
I_SAMPC_SUM_ABSOLUTE_VALUES = 0	Sum of absolute values. Available since version 3.
I_SAMPC_SQUARE_ROOT_OF_SUM_SQUARES = 1	Square root of sum of squares. Available since version 3.

#### Description

Available methods of participation coefficient calculation.

#### Version

Available since version 3.

## XII IRobotSeismicResultsSaveParams

### Class Hierarchy

#### C++

```
interface IRobotSeismicResultsSaveParams : IDispatch;
```

#### C#

```
public interface IRobotSeismicResultsSaveParams;
```

### Visual Basic

```
Public Interface IRobotSeismicResultsSaveParams
```

### Description

Parameters of saving quadratic combination results for seismic analysis in a project.

### Version

Available since version 3.

## XII.1 IRobotSeismicResultsSaveParams Members

The following tables list the members exposed by IRobotSeismicResultsSaveParams.

### Public Fields

	Name	Description
◆	Displacements ( <a href="#">see page 1573</a> )	Flag enforcing saving of displacements .
◆	Forces ( <a href="#">see page 1573</a> )	Flag enforcing saving of forces.
◆	LocalDisplacements ( <a href="#">see page 1573</a> )	Local displacements.
◆	NMQ ( <a href="#">see page 1574</a> )	Flag enforcing saving of N membrane forces, M moments and Q shear forces.
◆	OnlyQuadraticCombs ( <a href="#">see page 1574</a> )	Result save only for quadratic combinations.
◆	PanelsDir ( <a href="#">see page 1574</a> )	Direction for panels.
◆	PointNumber ( <a href="#">see page 1575</a> )	Definition of a number of points along the bar length.
◆	Reactions ( <a href="#">see page 1575</a> )	Flag enforcing saving of reactions.
◆	Reduced ( <a href="#">see page 1575</a> )	Flag enforcing saving of reduced results .
◆	Save ( <a href="#">see page 1576</a> )	Flag indicating if results are saved in the project.
◆	Stresses ( <a href="#">see page 1576</a> )	Flag enforcing saving of stresses .

## XII.2 IRobotSeismicResultsSaveParams Fields

The fields of the IRobotSeismicResultsSaveParams class are listed here.

### Public Fields

	Name	Description
◆	Displacements ( <a href="#">see page 1573</a> )	Flag enforcing saving of displacements .
◆	Forces ( <a href="#">see page 1573</a> )	Flag enforcing saving of forces.
◆	LocalDisplacements ( <a href="#">see page 1573</a> )	Local displacements.
◆	NMQ ( <a href="#">see page 1574</a> )	Flag enforcing saving of N membrane forces, M moments and Q shear forces.

❖	OnlyQuadraticCombs (see page 1574)	Result save only for quadratic combinations.
❖	PanelsDir (see page 1574)	Direction for panels.
❖	PointNumber (see page 1575)	Definition of a number of points along the bar length.
❖	Reactions (see page 1575)	Flag enforcing saving of reactions.
❖	Reduced (see page 1575)	Flag enforcing saving of reduced results .
❖	Save (see page 1576)	Flag indicating if results are saved in the project.
❖	Stresses (see page 1576)	Flag enforcing saving of stresses .

## XII.2.1 Displacements

### C++

```
HRESULT get_Displacements(VARIANT_BOOL* );
HRESULT put_Displacements(VARIANT_BOOL);
```

### C#

```
public bool Displacements { get; set; }
```

### Visual Basic

```
Public Displacements As Boolean
```

### Description

Flag enforcing saving of displacements .

### Version

Available since version 3.

## XII.2.2 Forces

### C++

```
HRESULT get_Forces(VARIANT_BOOL* );
HRESULT put_Forces(VARIANT_BOOL);
```

### C#

```
public bool Forces { get; set; }
```

### Visual Basic

```
Public Forces As Boolean
```

### Description

Flag enforcing saving of forces.

### Version

Available since version 3.

## XII.2.3 LocalDisplacements

### C++

```
HRESULT get_LocalDisplacements(VARIANT_BOOL* );
HRESULT put_LocalDisplacements(VARIANT_BOOL);
```

### C#

```
public bool LocalDisplacements { get; set; }
```

### Visual Basic

```
Public LocalDisplacements As Boolean
```

**Description**

Local displacements.

**XII.2.4 NMQ****C++**

```
HRESULT get_NMQ(VARIANT_BOOL* );
HRESULT put_NMQ(VARIANT_BOOL);
```

**C#**

```
public bool NMQ { get; set; }
```

**Visual Basic**

```
Public NMQ As Boolean
```

**Description**

Flag enforcing saving of N membrane forces, M moments and Q shear forces.

**Version**

Available since version 3.

**XII.2.5 OnlyQuadraticCombs****C++**

```
HRESULT get_OnlyQuadraticCombs(VARIANT_BOOL* );
HRESULT put_OnlyQuadraticCombs(VARIANT_BOOL);
```

**C#**

```
public bool OnlyQuadraticCombs { get; set; }
```

**Visual Basic**

```
Public OnlyQuadraticCombs As Boolean
```

**Description**

Result save only for quadratic combinations.

**Version**

Available since version 5.6.

**XII.2.6 PanelsDir****C++**

```
HRESULT get_PanelsDir(IRobotSeismicResultsPanelDirection* );
HRESULT put_PanelsDir(IRobotSeismicResultsPanelDirection);
```

**C#**

```
public IRobotSeismicResultsPanelDirection PanelsDir { get; set; }
```

**Visual Basic**

```
Public PanelsDir As IRobotSeismicResultsPanelDirection
```

**Description**

Direction for panels.

**Version**

Available since version 3.

## XII.2.7 PointNumber

### C++

```
HRESULT get_PointNumber(long* );
HRESULT put_PointNumber(long);
```

### C#

```
public long PointNumber { get; set; }
```

### Visual Basic

```
Public PointNumber As long
```

### Description

Definition of a number of points along the bar length.

### Version

Available since version 3.

## XII.2.8 Reactions

### C++

```
HRESULT get_Reactions(VARIANT_BOOL* );
HRESULT put_Reactions(VARIANT_BOOL);
```

### C#

```
public bool Reactions { get; set; }
```

### Visual Basic

```
Public Reactions As Boolean
```

### Description

Flag enforcing saving of reactions.

### Version

Available since version 3.

## XII.2.9 Reduced

### C++

```
HRESULT get_Reduced(VARIANT_BOOL* );
HRESULT put_Reduced(VARIANT_BOOL);
```

### C#

```
public bool Reduced { get; set; }
```

### Visual Basic

```
Public Reduced As Boolean
```

### Description

Flag enforcing saving of reduced results .

### Version

Available since version 3.

## XII.2.10 Save

### C++

```
HRESULT get_Save(VARIANT_BOOL* );
HRESULT put_Save(VARIANT_BOOL);
```

### C#

```
public bool Save { get; set; }
```

### Visual Basic

```
Public Save As Boolean
```

### Description

Flag indicating if results are saved in the project.

### Version

Available since version 3.

## XII.2.11 Stresses

### C++

```
HRESULT get_Stresses(VARIANT_BOOL* );
HRESULT put_Stresses(VARIANT_BOOL);
```

### C#

```
public bool Stresses { get; set; }
```

### Visual Basic

```
Public Stresses As Boolean
```

### Description

Flag enforcing saving of stresses .

### Version

Available since version 3.

## XIII IRobotSeismicResultsPanelDirection

### C++

```
enum IRobotSeismicResultsPanelDirection;
```

### C#

```
public enum IRobotSeismicResultsPanelDirection;
```

### Visual Basic

```
Public Enum IRobotSeismicResultsPanelDirection
```

### Members

Members	Description
I_SRPD_AUTOMATIC = 0	Available since version 3.
I_SRPD_DIR_X = 1	Available since version 3.
I_SRPD_DIR_Y = 2	Available since version 3.
I_SRPD_DIR_Z = 3	Available since version 3.

**Version**

Available since version 3.

## XIV IRobotModelGenerationParams

**Class Hierarchy****C++**

```
interface IRobotModelGenerationParams : IDispatch;
```

**C#**

```
public interface IRobotModelGenerationParams;
```

**Visual Basic**

```
Public Interface IRobotModelGenerationParams
```

**Description**

Generation parameters for structure calculation model.

**Version**

Available since version 3.

### XIV.1 IRobotModelGenerationParams Members

The following tables list the members exposed by IRobotModelGenerationParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AssemblingCase (↗ see page 1578)	Assembling case number.
❖	GenerateNodes_BarsAndFiniteElems (↗ see page 1578)	Node generation at intersections of bars and finite elements.
❖	GenerateNodes_DiagonalBars (↗ see page 1579)	Node generation at intersections of inclined bars.
❖	GenerateNodes_VertHorizBars (↗ see page 1579)	Node generation at intersections of vertical and horizontal bars.
❖	MaxElementLength (↗ see page 1579)	Maximum length of calculation element.
❖	NeglectedBars (↗ see page 1580)	List of bars to be disregarded during generation of the calculation model.
❖	NeglectedGeoObjects (↗ see page 1580)	List of geometrical objects to be disregarded during generation of the calculation model.
❖	ToleranceAutomatic (↗ see page 1580)	Enforcing the automatic value for tolerance.
❖	ToleranceValue (↗ see page 1581)	Tolerance value.

### XIV.2 IRobotModelGenerationParams Fields

The fields of the IRobotModelGenerationParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AssemblingCase (↗ see page 1578)	Assembling case number.
❖	GenerateNodes_BarsAndFiniteElems (↗ see page 1578)	Node generation at intersections of bars and finite elements.

◆	GenerateNodes_DiagonalBars ( <a href="#">see page 1579</a> )	Node generation at intersections of inclined bars.
◆	GenerateNodes_VertHorizBars ( <a href="#">see page 1579</a> )	Node generation at intersections of vertical and horizontal bars.
◆	MaxElementLength ( <a href="#">see page 1579</a> )	Maximum length of calculation element.
◆	NeglectedBars ( <a href="#">see page 1580</a> )	List of bars to be disregarded during generation of the calculation model.
◆	NeglectedGeoObjects ( <a href="#">see page 1580</a> )	List of geometrical objects to be disregarded during generation of the calculation model.
◆	ToleranceAutomatic ( <a href="#">see page 1580</a> )	Enforcing the automatic value for tolerance.
◆	ToleranceValue ( <a href="#">see page 1581</a> )	Tolerance value.

## XIV.2.1 AssemblingCase

### C++

```
HRESULT get_AssemblingCase(long* );
HRESULT put_AssemblingCase(long);
```

### C#

```
public long AssemblingCase { get; set; }
```

### Visual Basic

```
Public AssemblingCase As long
```

### Description

Assembling case number.

### Version

Available since version 3.

## XIV.2.2 GenerateNodes\_BarsAndFiniteElems

### C++

```
HRESULT get_GenerateNodes_BarsAndFiniteElems(VARIANT_BOOL* );
HRESULT put_GenerateNodes_BarsAndFiniteElems(VARIANT_BOOL);
```

### C#

```
public bool GenerateNodes_BarsAndFiniteElems { get; set; }
```

### Visual Basic

```
Public GenerateNodes_BarsAndFiniteElems As Boolean
```

### Description

Node generation at intersections of bars and finite elements.

### Version

Available since version 3.

## XIV.2.3 GenerateNodes\_DiagonalBars

### C++

```
HRESULT get_GenerateNodes_DiagonalBars(VARIANT_BOOL* );
HRESULT put_GenerateNodes_DiagonalBars(VARIANT_BOOL);
```

**C#**

```
public bool GenerateNodes_DiagonalBars { get; set; }
```

**Visual Basic**

```
Public GenerateNodes_DiagonalBars As Boolean
```

**Description**

Node generation at intersections of inclined bars.

**Version**

Available since version 3.

**XIV.2.4 GenerateNodes\_VertHorizBars****C++**

```
HRESULT get_GenerateNodes_VertHorizBars(VARIANT_BOOL* );
HRESULT put_GenerateNodes_VertHorizBars(VARIANT_BOOL);
```

**C#**

```
public bool GenerateNodes_VertHorizBars { get; set; }
```

**Visual Basic**

```
Public GenerateNodes_VertHorizBars As Boolean
```

**Description**

Node generation at intersections of vertical and horizontal bars.

**Version**

Available since version 3.

**XIV.2.5 MaxElementLength****C++**

```
HRESULT get_MaxElementLength(double* );
HRESULT put_MaxElementLength(double);
```

**C#**

```
public double MaxElementLength { get; set; }
```

**Visual Basic**

```
Public MaxElementLength As Double
```

**Description**

Maximum length of calculation element.

**Version**

Available since version 3.

**XIV.2.6 NeglectedBars****C++**

```
HRESULT get_NeglectedBars(BSTR* );
HRESULT put_NeglectedBars(BSTR);
```

**C#**

```
public String NeglectedBars { get; set; }
```

**Visual Basic**

```
Public NeglectedBars As String
```

**Description**

List of bars to be disregarded during generation of the calculation model.

**Version**

Available since version 3.

## XIV.2.7 NeglectedGeoObjects

**C++**

```
HRESULT get_NeglectedGeoObjects(BSTR* );
HRESULT put_NeglectedGeoObjects(BSTR);
```

**C#**

```
public String NeglectedGeoObjects { get; set; }
```

**Visual Basic**

```
Public NeglectedGeoObjects As String
```

**Description**

List of geometrical objects to be disregarded during generation of the calculation model.

**Version**

Available since version 3.

## XIV.2.8 ToleranceAutomatic

**C++**

```
HRESULT get_ToleranceAutomatic(VARIANT_BOOL* );
HRESULT put_ToleranceAutomatic(VARIANT_BOOL);
```

**C#**

```
public bool ToleranceAutomatic { get; set; }
```

**Visual Basic**

```
Public ToleranceAutomatic As Boolean
```

**Description**

Enforcing the automatic value for tolerance.

**Version**

Available since version 3.

## XIV.2.9 ToleranceValue

**C++**

```
HRESULT get_ToleranceValue(double* );
HRESULT put_ToleranceValue(double);
```

**C#**

```
public double ToleranceValue { get; set; }
```

**Visual Basic**

```
Public ToleranceValue As double
```

**Description**

Tolerance value.

**Version**

Available since version 3.

## XV IRobotBucklingDeformationParams

**Class Hierarchy****C++**

```
interface IRobotBucklingDeformationParams : IDispatch;
```

**C#**

```
public interface IRobotBucklingDeformationParams;
```

**Visual Basic**

```
Public Interface IRobotBucklingDeformationParams
```

**Description**

Parameters of buckling deformation.

**Version**

Available since version 3.

### XV.1 IRobotBucklingDeformationParams Members

The following tables list the members exposed by IRobotBucklingDeformationParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CaseNumber (see page 1582)	Number of selected buckling analysis case (zero value indicates that no case has been selected).
❖	MaxDisplacement (see page 1582)	Maximum displacement.
❖	OmitCaseForDeformations (see page 1582)	Flag indicating if a selected buckling case is to be disregarded for a structure including deformations .

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	GetModeCoeff (see page 1583)	Function returns coefficient for the indicated mode.
❖	SetModeCoeff (see page 1583)	Function defines coefficient for the indicated mode..

### XV.2 IRobotBucklingDeformationParams Fields

The fields of the IRobotBucklingDeformationParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CaseNumber (see page 1582)	Number of selected buckling analysis case (zero value indicates that no case has been selected).
❖	MaxDisplacement (see page 1582)	Maximum displacement.

	OmitCaseForDeformations (see page 1582)	Flag indicating if a selected buckling case is to be disregarded for a structure including deformations .
-----------------------------------------------------------------------------------	-----------------------------------------	-----------------------------------------------------------------------------------------------------------

## XV.2.1 CaseNumber

### C++

```
HRESULT get_CaseNumber(long* );
HRESULT put_CaseNumber(long);
```

### C#

```
public long CaseNumber { get; set; }
```

### Visual Basic

```
Public CaseNumber As long
```

### Description

Number of selected buckling analysis case (zero value indicates that no case has been selected).

### Version

Available since version 3.

## XV.2.2 MaxDisplacement

### C++

```
HRESULT get_MaxDisplacement(double* );
HRESULT put_MaxDisplacement(double);
```

### C#

```
public double MaxDisplacement { get; set; }
```

### Visual Basic

```
Public MaxDisplacement As double
```

### Description

Maximum displacement.

### Version

Available since version 3.

## XV.2.3 OmitCaseForDeformations

### C++

```
HRESULT get_OmitCaseForDeformations(VARIANT_BOOL* );
HRESULT put_OmitCaseForDeformations(VARIANT_BOOL);
```

### C#

```
public bool OmitCaseForDeformations { get; set; }
```

### Visual Basic

```
Public OmitCaseForDeformations As Boolean
```

### Description

Flag indicating if a selected buckling case is to be disregarded for a structure including deformations .

### Version

Available since version 3.

## XV.3 IRobotBucklingDeformationParams Methods

The methods of the IRobotBucklingDeformationParams class are listed here.

### Public Methods

	Name	Description
光环	GetModeCoeff ( <a href="#">see page 1583</a> )	Function returns coefficient for the indicated mode.
光环	SetModeCoeff ( <a href="#">see page 1583</a> )	Function defines coefficient for the indicated mode. .

### XV.3.1 GetModeCoeff

#### C++

```
HRESULT GetModeCoeff(long _mode, double* ret);
```

#### C#

```
public double GetModeCoeff(long _mode);
```

#### Visual Basic

```
Public Function GetModeCoeff(_mode As long) As double
```

#### Description

Function returns coefficient for the indicated mode.

#### Version

Available since version 3.

### XV.3.2 SetModeCoeff

#### C++

```
HRESULT SetModeCoeff(long _mode, double _coeff);
```

#### C#

```
public void SetModeCoeff(long _mode, double _coeff);
```

#### Visual Basic

```
Public Sub SetModeCoeff(_mode As long, _coeff As double)
```

#### Description

Function defines coefficient for the indicated mode. .

#### Version

Available since version 3.

## XVI IRobotCalcEngineEvents

### Class Hierarchy

#### C++

```
interface IRobotCalcEngineEvents : IDispatch;
```

#### C#

```
public interface IRobotCalcEngineEvents;
```

**Visual Basic**

```
Public Interface IRobotCalcEngineEvents
```

**Description**

Events generated by the RobotCalcEngine calculation module.

**Version**

Available since version 13.4.

## XVI.1 IRobotCalcEngineEvents Members

The following tables list the members exposed by IRobotCalcEngineEvents.

**Public Methods**

	Name	Description
💡	CalcMessage (see page 1584)	Function makes the messages from verification of computational model and structure analysis available.

## XVI.2 IRobotCalcEngineEvents Methods

The methods of the IRobotCalcEngineEvents class are listed here.

**Public Methods**

	Name	Description
💡	CalcMessage (see page 1584)	Function makes the messages from verification of computational model and structure analysis available.

### XVI.2.1 CalcMessage

**C++**

```
HRESULT CalcMessage(IRobotCalcMessageSource _msgSource, RobotCalcMessageSeverity  
_msgSeverity, long _msgId, BSTR _msgText, IRobotObjectType _objType, BSTR _objSel);
```

**C#**

```
public void CalcMessage(IRobotCalcMessageSource _msgSource, RobotCalcMessageSeverity  
_msgSeverity, long _msgId, String _msgText, IRobotObjectType _objType, String _objSel);
```

**Visual Basic**

```
Public Sub CalcMessage(_msgSource As IRobotCalcMessageSource, _msgSeverity As  
RobotCalcMessageSeverity, _msgId As Long, _msgText As String, _objType As IRobotObjectType,  
_objSel As String)
```

**Description**

Function makes the messages from verification of computational model and structure analysis available.

**Version**

Available since version 13.4.

## XVII IRobotCalcMessageSource

**C++**

```
enum IRobotCalcMessageSource;
```

**C#**

```
public enum IRobotCalcMessageSource;
```

**Visual Basic**

```
Public Enum IRobotCalcMessageSource
```

**Members**

Members	Description
I_CMS_VERIFICATION = 3	Verification of the computational model correctness. Available since version 13.4.
I_CMS_CALCULATIONS = 1	Structure analysis. Available since version 13.4.

**Version**

Available since version 13.4.

## XVIII IRobotCalcMessageSeverityLevel

**C++**

```
enum IRobotCalcMessageSeverityLevel;
```

**C#**

```
public enum IRobotCalcMessageSeverityLevel;
```

**Visual Basic**

```
Public Enum IRobotCalcMessageSeverityLevel
```

**Members**

Members	Description
I_CMSL_WARNING = 2	Available since version 13.4.
I_CMSL_ERROR = 3	Available since version 13.4.
I_CMSL_NOTE = 1	Available since version 13.4.

**Version**

Available since version 13.4.

## XIX IRobotCalculationStatus

**C++**

```
enum IRobotCalculationStatus;
```

**C#**

```
public enum IRobotCalculationStatus;
```

**Visual Basic**

```
Public Enum IRobotCalculationStatus
```

## Members

Members	Description
I_CS_COMPLETED = 0	Calculations succeeded; results are available. Available since version 13.4.
I_CS_FAILED_VERIFICATION = 1	Calculations have been stopped because of errors or warnings that occurred during the verification stage; no results. Available since version 13.4.
I_CS_FAILED_CALCULATION = 2	Calculations failed; no results. Available since version 13.4.
I_CS_CANCELLED_BY_USER = 3	Calculations were interrupted by the user; no results. Available since version 13.4.
I_CS_FAILED_GENERATION = 4	An error has occurred during generation of computational model; no results. Available since version 13.4.
I_CS_IN_PROGRESS = -1	Calculations in progress. Available since version 13.4.
I_CS_FAILED_NO_ENTITLEMENT = 5	No entitlements to perform calculations. Available since version 13.4.

## Version

Available since version 13.4.

# XX IRobotCalculationMode

## C++

```
enum IRobotCalculationMode;
```

## C#

```
public enum IRobotCalculationMode;
```

## Visual Basic

```
Public Enum IRobotCalculationMode
```

## Members

Members	Description
I_CM_LOCAL = 0	Calculations performed in the synchronous way on the local machine. Available since version 13.4.
I_CM_LOCAL_ASYNCHRONOUS = 1	Calculations performed in the asynchronous way on the local machine. Available since version 13.4.
I_CM_CLOUD = 2	Calculations performed in the synchronous way in the calculation cloud. Available since version 13.4.
I_CM_CLOUD_ASYNCHRONOUS = 3	Calculations performed in the asynchronous way in the calculation cloud. Available since version 13.4.

## Description

Available methods of performing calculations.

**Version**

Available since version 13.4.

## XXI Direct Analysis Method

Available since version 15.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotDAMAnalysisType ( <a href="#">see page 1589</a> )	.
	IRobotDAMLateralLoadCombType ( <a href="#">see page 1594</a> )	.
	IRobotDAMSteelMembersReductionType ( <a href="#">see page 1597</a> )	.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotDAMCalcModule ( <a href="#">see page 1587</a> )	Calculation module for Direct Analysis Method.
	IRobotDAMNotionalLoads ( <a href="#">see page 1590</a> )	.
	IRobotDAMReducedStiffness ( <a href="#">see page 1594</a> )	.
	IRobotDAMParams ( <a href="#">see page 1597</a> )	.

### XXI.1 IRobotDAMCalcModule

**Class Hierarchy****C++**

```
interface IRobotDAMCalcModule : IDispatch;
```

**C#**

```
public interface IRobotDAMCalcModule;
```

**Visual Basic**

```
Public Interface IRobotDAMCalcModule
```

**Description**

Calculation module for Direct Analysis Method.

**Version**

Available since version 15.

#### XXI.1.1 IRobotDAMCalcModule Members

The following tables list the members exposed by IRobotDAMCalcModule.

**Public Fields**

	<b>Name</b>	<b>Description</b>
	IsActive ( <a href="#">see page 1588</a> )	.
	Params ( <a href="#">see page 1588</a> )	.

## Public Methods

	Name	Description
✖	DeleteModel (see page 1589)	
✖	Run (see page 1589)	

## XXI.1.2 IRobotDAMCalcModule Fields

The fields of the IRobotDAMCalcModule class are listed here.

### Public Fields

	Name	Description
❖	IsActive (see page 1588)	.
❖	Params (see page 1588)	.

### XXI.1.2.1 IsActive

#### C++

```
HRESULT get_IsActive(VARIANT_BOOL* );
HRESULT put_IsActive(VARIANT_BOOL);
```

#### C#

```
public bool IsActive { get; set; }
```

#### Visual Basic

```
Public IsActive As Boolean
```

#### Description

#### Version

Available since version 15.

## XXI.1.2.2 Params

#### C++

```
HRESULT get_Params(IRobotDAMParams** );
```

#### C#

```
public IRobotDAMParams Params { get; }
```

#### Visual Basic

```
Public ReadOnly Params As IRobotDAMParams
```

#### Description

#### Version

Available since version 15.

## XXI.1.3 IRobotDAMCalcModule Methods

The methods of the IRobotDAMCalcModule class are listed here.

### Public Methods

	Name	Description
✖	DeleteModel (see page 1589)	



Run (see page 1589)

### XXI.1.3.1 DeleteModel

#### C++

```
HRESULT DeleteModel();
```

#### C#

```
public void DeleteModel();
```

#### Visual Basic

```
Public Sub DeleteModel()
```

#### Version

Available since version 15.

### XXI.1.3.2 Run

#### C++

```
HRESULT Run(VARIANT_BOOL* ret);
```

#### C#

```
public bool Run();
```

#### Visual Basic

```
Public Function Run() As Boolean
```

#### Version

Available since version 15.

## XXI.2 IRobotDAMAnalysisType

#### C++

```
enum IRobotDAMAnalysisType;
```

#### C#

```
public enum IRobotDAMAnalysisType;
```

#### Visual Basic

```
Public Enum IRobotDAMAnalysisType
```

#### Members

Members	Description
I_DAT_ANSI_AISC_360_10_LRFD = 0	.
I_DAT_ANSI_AISC_360_10_ASD = 1	.
I_DAT_USER_DEFINED = 2	.

#### Description

**Version**

Available since version 15.

## **XXI.3 IRobotDAMNotionalLoads**

**Class Hierarchy****C++**

```
interface IRobotDAMNotionalLoads : IDispatch;
```

**C#**

```
public interface IRobotDAMNotionalLoads;
```

**Visual Basic**

```
Public Interface IRobotDAMNotionalLoads
```

**Description****Version**

Available since version 15.

### **XXI.3.1 IRobotDAMNotionalLoads Members**

The following tables list the members exposed by IRobotDAMNotionalLoads.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	ActiveXN (↗ see page 1591)	.
❖	ActiveXP (↗ see page 1591)	.
❖	ActiveYN (↗ see page 1592)	.
❖	ActiveYP (↗ see page 1592)	.
❖	Coefficient (↗ see page 1592)	.
❖	GravityLoadCombEnabled (↗ see page 1593)	.
❖	LateralLoadCombEnabled (↗ see page 1593)	.
❖	LateralLoadCombType (↗ see page 1593)	.

### **XXI.3.2 IRobotDAMNotionalLoads Fields**

The fields of the IRobotDAMNotionalLoads class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	ActiveXN (↗ see page 1591)	.
❖	ActiveXP (↗ see page 1591)	.
❖	ActiveYN (↗ see page 1592)	.
❖	ActiveYP (↗ see page 1592)	.
❖	Coefficient (↗ see page 1592)	.
❖	GravityLoadCombEnabled (↗ see page 1593)	.

◆	LateralLoadCombEnabled (see page 1593)	.
◆	LateralLoadCombType (see page 1593)	.

### XXI.3.2.1 ActiveXN

#### C++

```
HRESULT get_ActiveXN(VARIANT_BOOL* );
HRESULT put_ActiveXN(VARIANT_BOOL);
```

#### C#

```
public bool ActiveXN { get; set; }
```

#### Visual Basic

```
Public ActiveXN As Boolean
```

#### Description

#### Version

Available since version 15.

### XXI.3.2.2 ActiveXP

#### C++

```
HRESULT get_ActiveXP(VARIANT_BOOL* );
HRESULT put_ActiveXP(VARIANT_BOOL);
```

#### C#

```
public bool ActiveXP { get; set; }
```

#### Visual Basic

```
Public ActiveXP As Boolean
```

#### Description

#### Version

Available since version 15.

### XXI.3.2.3 ActiveYN

#### C++

```
HRESULT get_ActiveYN(VARIANT_BOOL* );
HRESULT put_ActiveYN(VARIANT_BOOL);
```

#### C#

```
public bool ActiveYN { get; set; }
```

#### Visual Basic

```
Public ActiveYN As Boolean
```

#### Description

#### Version

Available since version 15.

#### XXI.3.2.4 ActiveYP

##### C++

```
HRESULT get_ActiveYP(VARIANT_BOOL* );
HRESULT put_ActiveYP(VARIANT_BOOL);
```

##### C#

```
public bool ActiveYP { get; set; }
```

##### Visual Basic

```
Public ActiveYP As Boolean
```

##### Description

##### Version

Available since version 15.

#### XXI.3.2.5 Coefficient

##### C++

```
HRESULT get_Coefficient(double* );
HRESULT put_Coefficient(double);
```

##### C#

```
public double Coefficient { get; set; }
```

##### Visual Basic

```
Public Coefficient As Double
```

##### Description

##### Version

Available since version 15.

#### XXI.3.2.6 GravityLoadCombEnabled

##### C++

```
HRESULT get_GravityLoadCombEnabled(VARIANT_BOOL* );
HRESULT put_GravityLoadCombEnabled(VARIANT_BOOL);
```

##### C#

```
public bool GravityLoadCombEnabled { get; set; }
```

##### Visual Basic

```
Public GravityLoadCombEnabled As Boolean
```

##### Description

##### Version

Available since version 15.

### XXI.3.2.7 LateralLoadCombEnabled

#### C++

```
HRESULT get_LateralLoadCombEnabled(VARIANT_BOOL* );
HRESULT put_LateralLoadCombEnabled(VARIANT_BOOL);
```

#### C#

```
public bool LateralLoadCombEnabled { get; set; }
```

#### Visual Basic

```
Public LateralLoadCombEnabled As Boolean
```

#### Description

#### Version

Available since version 15.

### XXI.3.2.8 LateralLoadCombType

#### C++

```
HRESULT get_LateralLoadCombType(IRobotDAMLateralLoadCombType* );
HRESULT put_LateralLoadCombType(IRobotDAMLateralLoadCombType);
```

#### C#

```
public IRobotDAMLateralLoadCombType LateralLoadCombType { get; set; }
```

#### Visual Basic

```
Public LateralLoadCombType As IRobotDAMLateralLoadCombType
```

#### Description

#### Version

Available since version 15.

## XXI.4 IRobotDAMLateralLoadCombType

#### C++

```
enum IRobotDAMLateralLoadCombType;
```

#### C#

```
public enum IRobotDAMLateralLoadCombType;
```

#### Visual Basic

```
Public Enum IRobotDAMLateralLoadCombType
```

#### Members

Members	Description
I_DLLCT_LATERAL_LOAD_DIR = 0	.
I_DLLCT_ONE_DIR_ONLY = 1	Available since version 15.
I_DLLCT_TWO_DIR_INDEPENDENTLY = 2	Available since version 15.

## Description

### Version

Available since version 15.

## XXI.5 IRobotDAMReducedStiffness

### Class Hierarchy

#### C++

```
interface IRobotDAMReducedStiffness : IDispatch;
```

#### C#

```
public interface IRobotDAMReducedStiffness;
```

### Visual Basic

```
Public Interface IRobotDAMReducedStiffness
```

## Description

### Version

Available since version 15.

## XXI.5.1 IRobotDAMReducedStiffness Members

The following tables list the members exposed by IRobotDAMReducedStiffness.

### Public Fields

	Name	Description
❖	RCBeamsValue (see page 1595)	.
❖	RCColumnsAndWallsValue (see page 1595)	.
❖	RCSlabsValue (see page 1596)	.
❖	SteelMembersReductionType (see page 1596)	.
❖	SteelMembersTauBValue (see page 1596)	.
❖	SteelMembersValue (see page 1597)	.

## XXI.5.2 IRobotDAMReducedStiffness Fields

The fields of the IRobotDAMReducedStiffness class are listed here.

### Public Fields

	Name	Description
❖	RCBeamsValue (see page 1595)	.
❖	RCColumnsAndWallsValue (see page 1595)	.
❖	RCSlabsValue (see page 1596)	.
❖	SteelMembersReductionType (see page 1596)	.

SteelMembersTauBValue (see page 1596)	.
SteelMembersValue (see page 1597)	.

### XXI.5.2.1 RCBeamsValue

#### C++

```
HRESULT get_RCBeamsValue(double*);  
HRESULT put_RCBeamsValue(double);
```

#### C#

```
public double RCBeamsValue { get; set; }
```

#### Visual Basic

```
Public RCBeamsValue As Double
```

#### Description

#### Version

Available since version 15.

### XXI.5.2.2 RCColumnsAndWallsValue

#### C++

```
HRESULT get_RCColumnsAndWallsValue(double*);  
HRESULT put_RCColumnsAndWallsValue(double);
```

#### C#

```
public double RCColumnsAndWallsValue { get; set; }
```

#### Visual Basic

```
Public RCColumnsAndWallsValue As Double
```

#### Description

#### Version

Available since version 15.

### XXI.5.2.3 RCSlabsValue

#### C++

```
HRESULT get_RCSlabsValue(double*);  
HRESULT put_RCSlabsValue(double);
```

#### C#

```
public double RCSlabsValue { get; set; }
```

#### Visual Basic

```
Public RCSlabsValue As Double
```

#### Description

#### Version

Available since version 15.

#### XXI.5.2.4 SteelMembersReductionType

##### C++

```
HRESULT get_SteelMembersReductionType(IRobotDAMSteelMembersReductionType* );
HRESULT put_SteelMembersReductionType(IRobotDAMSteelMembersReductionType);
```

##### C#

```
public IRobotDAMSteelMembersReductionType SteelMembersReductionType { get; set; }
```

##### Visual Basic

```
Public SteelMembersReductionType As IRobotDAMSteelMembersReductionType
```

##### Description

##### Version

Available since version 15.

#### XXI.5.2.5 SteelMembersTauBValue

##### C++

```
HRESULT get_SteelMembersTauBValue(double* );
HRESULT put_SteelMembersTauBValue(double);
```

##### C#

```
public double SteelMembersTauBValue { get; set; }
```

##### Visual Basic

```
Public SteelMembersTauBValue As double
```

##### Description

##### Version

Available since version 15.

#### XXI.5.2.6 SteelMembersValue

##### C++

```
HRESULT get_SteelMembersValue(double* );
HRESULT put_SteelMembersValue(double);
```

##### C#

```
public double SteelMembersValue { get; set; }
```

##### Visual Basic

```
Public SteelMembersValue As double
```

##### Description

##### Version

Available since version 15.

## XXI.6 IRobotDAMSteelMembersReductionType

### C++

```
enum IRobotDAMSteelMembersReductionType;
```

### C#

```
public enum IRobotDAMSteelMembersReductionType;
```

### Visual Basic

```
Public Enum IRobotDAMSteelMembersReductionType
```

### Members

Members	Description
I_DSMRT_FACTOR_0_8 = 0	.
I_DSMRT_FACTOR_0_8_AND_TAU_B = 1	Available since version 15.
I_DSMRT_FACTOR_0_8_AND_NL_INCREASE = 2	Available since version 15.

### Description

### Version

Available since version 15.

## XXI.7 IRobotDAMParams

### Class Hierarchy

### C++

```
interface IRobotDAMParams : IDispatch;
```

### C#

```
public interface IRobotDAMParams;
```

### Visual Basic

```
Public Interface IRobotDAMParams
```

### Description

### Version

Available since version 15.

### XXI.7.1 IRobotDAMParams Members

The following tables list the members exposed by IRobotDAMParams.

### Public Fields

	Name	Description
❖	Analysis (see page 1598)	.
❖	NotionalLoads (see page 1599)	.

	ReducedStiffness ( <a href="#">see page 1599</a> )	.
-----------------------------------------------------------------------------------	----------------------------------------------------	---

**Public Methods**

	Name	Description
	GetNLDPParams ( <a href="#">see page 1600</a> )	
	SetNLDPParams ( <a href="#">see page 1600</a> )	
	Update ( <a href="#">see page 1600</a> )	

**XXI.7.2 IRobotDAMParams Fields**

The fields of the IRobotDAMParams class are listed here.

**Public Fields**

	Name	Description
	Analysis ( <a href="#">see page 1598</a> )	.
	NotionalLoads ( <a href="#">see page 1599</a> )	.
	ReducedStiffness ( <a href="#">see page 1599</a> )	.

**XXI.7.2.1 Analysis****C++**

```
HRESULT get_Analysis(IRobotDAMAnalysisType* );
HRESULT put_Analysis(IRobotDAMAnalysisType);
```

**C#**

```
public IRobotDAMAnalysisType Analysis { get; set; }
```

**Visual Basic**

```
Public Analysis As IRobotDAMAnalysisType
```

**Description****Version**

Available since version 15.

**XXI.7.2.2 NotionalLoads****C++**

```
HRESULT get_NotionalLoads(IRobotDAMNotionalLoads** );
```

**C#**

```
public IRobotDAMNotionalLoads NotionalLoads { get; }
```

**Visual Basic**

```
Public ReadOnly NotionalLoads As IRobotDAMNotionalLoads
```

**Description****Version**

Available since version 15.

### XXI.7.2.3 ReducedStiffness

#### C++

```
HRESULT get_ReducedStiffness(IRobotDAMReducedStiffness**);
```

#### C#

```
public IRobotDAMReducedStiffness ReducedStiffness { get; }
```

#### Visual Basic

```
Public ReadOnly ReducedStiffness As IRobotDAMReducedStiffness
```

#### Description

#### Version

Available since version 15.

## XXI.7.3 IRobotDAMParams Methods

The methods of the IRobotDAMParams class are listed here.

#### Public Methods

	Name	Description
●	GetNLPDParams (see page 1600)	
●	SetNLPDParams (see page 1600)	
●	Update (see page 1600)	

### XXI.7.3.1 GetNLPDParams

#### C++

```
HRESULT GetNLPDParams(IRobotNonlinearAnalysisParams** ret);
```

#### C#

```
public IRobotNonlinearAnalysisParams* GetNLPDParams();
```

#### Visual Basic

```
Public Function GetNLPDParams() As IRobotNonlinearAnalysisParams*
```

#### Version

Available since version 15.

### XXI.7.3.2 SetNLPDParams

#### C++

```
HRESULT SetNLPDParams(IRobotNonlinearAnalysisParams* ret);
```

#### C#

```
public IRobotNonlinearAnalysisParams SetNLPDParams();
```

#### Visual Basic

```
Public Function SetNLPDParams() As IRobotNonlinearAnalysisParams
```

#### Version

Available since version 15.

### XXI.7.3.3 Update

**C++**

```
HRESULT Update();
```

**C#**

```
public void Update();
```

**Visual Basic**

```
Public Sub Update()
```

**Version**

Available since version 15.

## Global data types

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRobotWindowState ( <a href="#">see page 1644</a> )	Available window states.
	IRobotComponentType ( <a href="#">see page 1647</a> )	Available component types that can be created by means of the component factory.
	IRobotDegreeOfFreedom ( <a href="#">see page 1648</a> )	Available degrees of freedom.

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRobotValuesArray ( <a href="#">see page 1627</a> )	Interface representing a one-dimensional table of real numbers.
	IRobotNamesArray ( <a href="#">see page 1629</a> )	The interface describing a one-dimensional table of names represented by character strings.
	IRobotNumbersArray ( <a href="#">see page 1632</a> )	The interface representing a table of integers. Table elements are indexed from 1 to Count ( <a href="#">see page 1633</a> ). .
	IRobotObjectsArray ( <a href="#">see page 1634</a> )	The interface providing access to the table of objects .
	IRobotNumbersCollection ( <a href="#">see page 1636</a> )	Collection of numbers (integers).
	IRobotCodeRegistrar ( <a href="#">see page 1637</a> )	In order to simplify the process of registering external codes extending the Robot program, RobotCodeRegistrar has been defined.
	IRobotWindow ( <a href="#">see page 1641</a> )	General interface describing a window in the Robot program. It describes both the main application window and windows which comprise individual views. .
	IRobotProtectionInfo ( <a href="#">see page 1644</a> )	Object provides access to information about protection of the Robot program and its extensions.
	IRobotComponentFactory ( <a href="#">see page 1645</a> )	Auxiliary component factory. It should be applied instead of the standard New operator for creation of "free" components in case when the application is to cooperate with both the Robot program and Kernel, depending on the user selection. .
	IRobotPointsArray ( <a href="#">see page 1649</a> )	Table of points in the 3D space. .
	IRobotSerializable ( <a href="#">see page 1651</a> )	Interface of the object whose data can be saved in/read from the stream.

	IRobotNumbersDictionary ( <a href="#">see page 1653</a> )	Dictionary of pairs (integer, integer).
-----------------------------------------------------------------------------------	-----------------------------------------------------------	-----------------------------------------

## I Geometric data types

### Enumerations

	Name	Description
	IRobotGeoCoordinateAxis ( <a href="#">see page 1606</a> )	Coordinate axes.
	IRobotGeoCoordinateSystem ( <a href="#">see page 1607</a> )	Types of coordinate systems.
	IRobotGeoSegmentType ( <a href="#">see page 1607</a> )	Types of segments that may compose complex geometrical objects.
	IRobotGeoObjectType ( <a href="#">see page 1608</a> )	Types of geometrical objects.
	IRobotGeoArcDefinitionMethod ( <a href="#">see page 1626</a> )	Available methods of arc definition.
	IRobotGeoCoordinateAxisSense ( <a href="#">see page 1626</a> )	Senses of the coordinate system axes.
	IRobotGeoPlane ( <a href="#">see page 1627</a> )	

### Interfaces

	Name	Description
	IRobotGeoPoint2D ( <a href="#">see page 1602</a> )	The interface describing a point on a plane.
	IRobotGeoPoint3D ( <a href="#">see page 1604</a> )	The interface describing a point in space.
	IRobotGeoCurveDiv ( <a href="#">see page 1608</a> )	The data type describing object discretization.
	IRobotGeoSegment ( <a href="#">see page 1610</a> )	Auxiliary interface for the definition of polylines and contours whose fragments may take the form of arcs or roundings.
	IRobotGeoSegmentLine ( <a href="#">see page 1611</a> )	Auxiliary interface for defining polylines and contours - a segment with or without a rounding.
	IRobotGeoSegmentArc ( <a href="#">see page 1613</a> )	Auxiliary interface for polyline and contour definition - arc.
	IRobotGeoObject ( <a href="#">see page 1615</a> )	The basic interface describing a geometrical object .
	IRobotGeoPolyline ( <a href="#">see page 1616</a> )	Interface describing a polyline.
	IRobotGeoSegmentCollection ( <a href="#">see page 1618</a> )	Collection of segments of geometrical objects.
	IRobotGeoContour ( <a href="#">see page 1618</a> )	Interface describing a contour. .
	IRobotGeoArc ( <a href="#">see page 1620</a> )	Interface describing an arc.
	IRobotGeoCircle ( <a href="#">see page 1621</a> )	Interface describing a circle.
	IRobotGeoPoint3DCollection ( <a href="#">see page 1623</a> )	Collection of points.
	IRobotGeoLayer ( <a href="#">see page 1624</a> )	Definition of a layer as a space sector. Layer is defined by specifying a cutting plane (by means of three points: P1 ( <a href="#">see page 1625</a> ), P2 ( <a href="#">see page 1625</a> ) and P3 ( <a href="#">see page 1625</a> )) and the fourth PDir ( <a href="#">see page 1625</a> ) point determining a half-space direction. Additionally, layer thickness may be determined by specifying the Thickness ( <a href="#">see page 1626</a> ) value. If the layer thickness is not specified, the layer describes the entire half-space.

## I.1 IRobotGeoPoint2D

### Class Hierarchy

#### C++

```
interface IRobotGeoPoint2D : IDispatch;
```

#### C#

```
public interface IRobotGeoPoint2D;
```

### Visual Basic

```
Public Interface IRobotGeoPoint2D
```

### Description

The interface describing a point on a plane.

### I.1.1 IRobotGeoPoint2D Members

The following tables list the members exposed by IRobotGeoPoint2D.

#### Public Fields

	Name	Description
◆	X (see page 1603)	X coordinate.
◆	Y (see page 1603)	Y coordinate.

#### Public Methods

	Name	Description
◆	Get (see page 1604)	The function takes all point coordinates.
◆	Set (see page 1604)	The function sets all point coordinates.

### I.1.2 IRobotGeoPoint2D Fields

The fields of the IRobotGeoPoint2D class are listed here.

#### Public Fields

	Name	Description
◆	X (see page 1603)	X coordinate.
◆	Y (see page 1603)	Y coordinate.

### I.1.2.1 X

#### C++

```
HRESULT get_X(double* );
HRESULT put_X(double);
```

#### C#

```
public double X { get; set; }
```

### Visual Basic

```
Public X As Double
```

### Description

X coordinate.

### I.1.2.2 Y

#### C++

```
HRESULT get_Y(double* );
HRESULT put_Y(double);
```

#### C#

```
public double Y { get; set; }
```

#### Visual Basic

```
Public Y As Double
```

#### Description

Y coordinate.

## I.1.3 IRobotGeoPoint2D Methods

The methods of the IRobotGeoPoint2D class are listed here.

#### Public Methods

	Name	Description
💡	Get (see page 1604)	The function takes all point coordinates.
💡	Set (see page 1604)	The function sets all point coordinates.

### I.1.3.1 Get

#### C++

```
HRESULT Get(double* _x, double* _y);
```

#### C#

```
public void Get(double* _x, double* _y);
```

#### Visual Basic

```
Public Sub Get(ByRef _x As Double*, ByRef _y As Double*)
```

#### Description

The function takes all point coordinates.

### I.1.3.2 Set

#### C++

```
HRESULT Set(double _x, double _y);
```

#### C#

```
public void Set(double _x, double _y);
```

#### Visual Basic

```
Public Sub Set(_x As Double, _y As Double)
```

#### Description

The function sets all point coordinates.

## I.2 IRobotGeoPoint3D

#### Class Hierarchy

**C++**

```
interface IRobotGeoPoint3D : IDispatch;
```

**C#**

```
public interface IRobotGeoPoint3D;
```

**Visual Basic**

```
Public Interface IRobotGeoPoint3D
```

**Description**

The interface describing a point in space.

**I.2.1 IRobotGeoPoint3D Members**

The following tables list the members exposed by IRobotGeoPoint3D.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	X (see page 1605)	X coordinate .
◆	Y (see page 1605)	Y coordinate.
◆	Z (see page 1605)	Z coordinate.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	Get (see page 1606)	The function gives back all point coordinates.
≡	Set (see page 1606)	The function sets all point coordinates..

**I.2.2 IRobotGeoPoint3D Fields**

The fields of the IRobotGeoPoint3D class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	X (see page 1605)	X coordinate .
◆	Y (see page 1605)	Y coordinate.
◆	Z (see page 1605)	Z coordinate.

**I.2.2.1 X****C++**

```
HRESULT get_X(double* );
HRESULT put_X(double);
```

**C#**

```
public double X { get; set; }
```

**Visual Basic**

```
Public X As Double
```

**Description**

X coordinate .

### I.2.2.2 Y

#### C++

```
HRESULT get_Y(double* );
HRESULT put_Y(double);
```

#### C#

```
public double Y { get; set; }
```

#### Visual Basic

```
Public Y As double
```

#### Description

Y coordinate.

### I.2.2.3 Z

#### C++

```
HRESULT get_Z(double* );
HRESULT put_Z(double);
```

#### C#

```
public double Z { get; set; }
```

#### Visual Basic

```
Public Z As double
```

#### Description

Z coordinate.

## I.2.3 IRobotGeoPoint3D Methods

The methods of the IRobotGeoPoint3D class are listed here.

#### Public Methods

	Name	Description
⊕	Get ( see page 1606)	The function gives back all point coordinates.
⊕	Set ( see page 1606)	The function sets all point coordinates..

### I.2.3.1 Get

#### C++

```
HRESULT Get(double* _x, double* _y, double* _z);
```

#### C#

```
public void Get(double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Sub Get(ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

#### Description

The function gives back all point coordinates.

### I.2.3.2 Set

**C++**

```
HRESULT Set(double _x, double _y, double _z);
```

**C#**

```
public void Set(double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub Set(_x As double, _y As double, _z As double)
```

#### Description

The function sets all point coordinates..

## I.3 IRobotGeoCoordinateAxis

**C++**

```
enum IRobotGeoCoordinateAxis;
```

**C#**

```
public enum IRobotGeoCoordinateAxis;
```

**Visual Basic**

```
Public Enum IRobotGeoCoordinateAxis
```

#### Members

Members	Description
I_GCA_OX = 1	Axis OX.
I_GCA_OY = 2	Axis OY.
I_GCA_OZ = 3	Axis OZ.
I_GCA_NONE = 0	No coordinate axis is indicated. Available since version 7.

#### Description

Coordinate axes.

## I.4 IRobotGeoCoordinateSystem

**C++**

```
enum IRobotGeoCoordinateSystem;
```

**C#**

```
public enum IRobotGeoCoordinateSystem;
```

**Visual Basic**

```
Public Enum IRobotGeoCoordinateSystem
```

#### Members

Members	Description
I_GCS_GLOBAL = 0	In global system.
I_GCS_LOCAL_AFTER_OFFSET = 1	Local coordinate system of a bar in the position considering translation.

I_GCS_LOCAL_ORIGINAL = 2	Local coordinate system of a bar.
--------------------------	-----------------------------------

### Description

Types of coordinate systems.

## I.5 IRobotGeoSegmentType

### C++

```
enum IRobotGeoSegmentType;
```

### C#

```
public enum IRobotGeoSegmentType;
```

### Visual Basic

```
Public Enum IRobotGeoSegmentType
```

### Members

Members	Description
I_GST_NONE = 0	Undefined.
I_GST_LINE = 1	Line (with a rounding).
I_GST_ARC = 2	Arc.

### Description

Types of segments that may compose complex geometrical objects.

## I.6 IRobotGeoObjectType

### C++

```
enum IRobotGeoObjectType;
```

### C#

```
public enum IRobotGeoObjectType;
```

### Visual Basic

```
Public Enum IRobotGeoObjectType
```

### Members

Members	Description
I_GOT_NONE = 0	Undefined.
I_GOT_POLYLINE = 1	Polyline.
I_GOT_CONTOUR = 2	Contour.
I_GOT_ARC = 4	Arc.
I_GOT_CIRCLE = 7	Circle.
I_GOT_INTERSECTION = 100	Overlap of objects .

### Description

Types of geometrical objects.

## I.7 IRobotGeoCurveDiv

### Class Hierarchy

**C++**

```
interface IRobotGeoCurveDiv : IDispatch;
```

**C#**

```
public interface IRobotGeoCurveDiv;
```

**Visual Basic**

```
Public Interface IRobotGeoCurveDiv
```

**Description**

The data type describing object discretization.

**I.7.1 IRobotGeoCurveDiv Members**

The following tables list the members exposed by IRobotGeoCurveDiv.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	IsAnalytical ( <a href="#">see page 1609</a> )	Arc defined analytically (arc discretization options are not available).
◆	Length ( <a href="#">see page 1609</a> )	Segment length.
◆	Mode ( <a href="#">see page 1609</a> )	0 - fixed number of divisions 1 - number of divisions per 360 degrees 2 - segment length.
◆	N ( <a href="#">see page 1610</a> )	Number of divisions.

**I.7.2 IRobotGeoCurveDiv Fields**

The fields of the IRobotGeoCurveDiv class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	IsAnalytical ( <a href="#">see page 1609</a> )	Arc defined analytically (arc discretization options are not available).
◆	Length ( <a href="#">see page 1609</a> )	Segment length.
◆	Mode ( <a href="#">see page 1609</a> )	0 - fixed number of divisions 1 - number of divisions per 360 degrees 2 - segment length.
◆	N ( <a href="#">see page 1610</a> )	Number of divisions.

**I.7.2.1 IsAnalytical****C++**

```
HRESULT get_IsAnalytical(VARIANT_BOOL* );
HRESULT put_IsAnalytical(VARIANT_BOOL);
```

**C#**

```
public bool IsAnalytical { get; set; }
```

**Visual Basic**

```
Public IsAnalytical As Boolean
```

**Description**

Arc defined analytically (arc discretization options are not available).

**Version**

Available since version 8.2.

### I.7.2.2 Length

#### C++

```
HRESULT get_Length(double* );
HRESULT put_Length(double);
```

#### C#

```
public double Length { get; set; }
```

#### Visual Basic

```
Public Length As Double
```

#### Description

Segment length.

#### Version

Available since version 8.2.

### I.7.2.3 Mode

#### C++

```
HRESULT get_Mode(long* );
HRESULT put_Mode(long);
```

#### C#

```
public long Mode { get; set; }
```

#### Visual Basic

```
Public Mode As Long
```

#### Description

0 - fixed number of divisions 1 - number of divisions per 360 degrees 2 - segment length.

### I.7.2.4 N

#### C++

```
HRESULT get_N(long* );
HRESULT put_N(long);
```

#### C#

```
public long N { get; set; }
```

#### Visual Basic

```
Public N As Long
```

#### Description

Number of divisions.

## I.8 IRobotGeoSegment

#### Class Hierarchy

#### C++

```
interface IRobotGeoSegment : IDispatch;
```

#### C#

```
public interface IRobotGeoSegment;
```

**Visual Basic**

```
Public Interface IRobotGeoSegment
```

**Description**

Auxiliary interface for the definition of polylines and contours whose fragments may take the form of arcs or roundings.

**I.8.1 IRobotGeoSegment Members**

The following tables list the members exposed by IRobotGeoSegment.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	P1 (see page 1611)	Initial point of a segment (end point is taken from the next segment).
◆	Type (see page 1611)	Segment type.

**I.8.2 IRobotGeoSegment Fields**

The fields of the IRobotGeoSegment class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	P1 (see page 1611)	Initial point of a segment (end point is taken from the next segment).
◆	Type (see page 1611)	Segment type.

**I.8.2.1 P1****C++**

```
HRESULT get_P1(IRobotGeoPoint3D**);
HRESULT put_P1(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D P1 { get; set; }
```

**Visual Basic**

```
Public P1 As IRobotGeoPoint3D
```

**Description**

Initial point of a segment (end point is taken from the next segment).

**I.8.2.2 Type****C++**

```
HRESULT get_Type(IRobotGeoSegmentType*);
```

**C#**

```
public IRobotGeoSegmentType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRobotGeoSegmentType
```

**Description**

Segment type.

## I.9 IRobotGeoSegmentLine

### Class Hierarchy

#### C++

```
interface IRobotGeoSegmentLine : IRobotGeoSegment;
```

#### C#

```
public interface IRobotGeoSegmentLine : IRobotGeoSegment;
```

### Visual Basic

```
Public Interface IRobotGeoSegmentLine
```

#### Description

Auxiliary interface for defining polylines and contours - a segment with or without a rounding.

### I.9.1 IRobotGeoSegmentLine Members

The following tables list the members exposed by IRobotGeoSegmentLine.

#### Public Fields

	Name	Description
◆	Div ( [ see page 1612)	Rounding discretization.
◆	NDiv ( [ see page 1612)	Number of divisions - discretization of the segment.
◆	P1 ( [ see page 1611)	Initial point of a segment (end point is taken from the next segment).
◆	Radius ( [ see page 1612)	Rounding radius.
◆	Round ( [ see page 1613)	Rounded or not.
◆	Type ( [ see page 1611)	Segment type.

### I.9.2 IRobotGeoSegmentLine Fields

The fields of the IRobotGeoSegmentLine class are listed here.

#### Public Fields

	Name	Description
◆	Div ( [ see page 1612)	Rounding discretization.
◆	NDiv ( [ see page 1612)	Number of divisions - discretization of the segment.
◆	Radius ( [ see page 1612)	Rounding radius.
◆	Round ( [ see page 1613)	Rounded or not.

#### I.9.2.1 Div

##### C++

```
HRESULT get_Div(IRobotGeoCurveDiv**);
HRESULT put_Div(IRobotGeoCurveDiv*);
```

##### C#

```
public IRobotGeoCurveDiv Div { get; set; }
```

### Visual Basic

```
Public Div As IRobotGeoCurveDiv
```

#### Description

Rounding discretization.

### I.9.2.2 NDiv

#### C++

```
HRESULT get_NDiv(long* );
HRESULT put_NDiv(long);
```

#### C#

```
public long NDiv { get; set; }
```

#### Visual Basic

```
Public NDiv As long
```

#### Description

Number of divisions - discretization of the segment.

### I.9.2.3 Radius

#### C++

```
HRESULT get_Radius(double* );
HRESULT put_Radius(double);
```

#### C#

```
public double Radius { get; set; }
```

#### Visual Basic

```
Public Radius As double
```

#### Description

Rounding radius.

### I.9.2.4 Round

#### C++

```
HRESULT get_Round(VARIANT_BOOL* );
HRESULT put_Round(VARIANT_BOOL);
```

#### C#

```
public bool Round { get; set; }
```

#### Visual Basic

```
Public Round As Boolean
```

#### Description

Rounded or not.

## I.10 IRobotGeoSegmentArc

### Class Hierarchy

#### C++

```
interface IRobotGeoSegmentArc : IRobotGeoSegment;
```

#### C#

```
public interface IRobotGeoSegmentArc : IRobotGeoSegment;
```

## Visual Basic

```
Public Interface IRobotGeoSegmentArc
```

### Description

Auxiliary interface for polyline and contour definition - arc.

## I.10.1 IRobotGeoSegmentArc Members

The following tables list the members exposed by IRobotGeoSegmentArc.

### Public Fields

	Name	Description
◆	Div ( [ see page 1614 )	Arc discretization.
◆	P1 ( [ see page 1611 )	Initial point of a segment (end point is taken from the next segment).
◆	P2 ( [ see page 1614 )	Point on an arc.
◆	Type ( [ see page 1611 )	Segment type.

### Public Methods

	Name	Description
◆	Set ( [ see page 1614 )	The function sets the initial arc point P1 ( [ see page 1611 ) and a point along the arc P2 ( [ see page 1614 ).

## I.10.2 IRobotGeoSegmentArc Fields

The fields of the IRobotGeoSegmentArc class are listed here.

### Public Fields

	Name	Description
◆	Div ( [ see page 1614 )	Arc discretization.
◆	P2 ( [ see page 1614 )	Point on an arc.

### I.10.2.1 Div

#### C++

```
HRESULT get_Div(IRobotGeoCurveDiv**);
HRESULT put_Div(IRobotGeoCurveDiv*);
```

#### C#

```
public IRobotGeoCurveDiv Div { get; set; }
```

#### Visual Basic

```
Public Div As IRobotGeoCurveDiv
```

### Description

Arc discretization.

### I.10.2.2 P2

#### C++

```
HRESULT get_P2(IRobotGeoPoint3D**);
HRESULT put_P2(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D P2 { get; set; }
```

**Visual Basic**

```
Public P2 As IRobotGeoPoint3D
```

**Description**

Point on an arc.

**I.10.3 IRobotGeoSegmentArc Methods**

The methods of the IRobotGeoSegmentArc class are listed here.

**Public Methods**

	Name	Description
✳	Set (see page 1614)	The function sets the initial arc point P1 (see page 1611) and a point along the arc P2 (see page 1614).

**I.10.3.1 Set****C++**

```
HRESULT Set(IRobotGeoPoint3D* _p1, IRobotGeoPoint3D* _p2);
```

**C#**

```
public void Set(IRobotGeoPoint3D _p1, IRobotGeoPoint3D _p2);
```

**Visual Basic**

```
Public Sub Set(ByRef _p1 As IRobotGeoPoint3D, ByRef _p2 As IRobotGeoPoint3D)
```

**Description**

The function sets the initial arc point P1 (see page 1611) and a point along the arc P2 (see page 1614).

**I.11 IRobotGeoObject****Class Hierarchy****C++**

```
interface IRobotGeoObject : IDispatch;
```

**C#**

```
public interface IRobotGeoObject;
```

**Visual Basic**

```
Public Interface IRobotGeoObject
```

**Description**

The basic interface describing a geometrical object .

**I.11.1 IRobotGeoObject Members**

The following tables list the members exposed by IRobotGeoObject.

**Public Fields**

	Name	Description
✳	Type (see page 1615)	Object type.

## Public Methods

	Name	Description
Initialize (see page 1616)		The function initializes internal data on the basis of the data defining the object. .

## I.11.2 IRobotGeoObject Fields

The fields of the IRobotGeoObject class are listed here.

### Public Fields

	Name	Description
Type (see page 1615)		Object type.

### I.11.2.1 Type

#### C++

```
HRESULT get_Type( IRobotGeoObjectType* );
```

#### C#

```
public IRobotGeoObjectType Type { get; }
```

#### Visual Basic

```
Public ReadOnly Type As IRobotGeoObjectType
```

#### Description

Object type.

## I.11.3 IRobotGeoObject Methods

The methods of the IRobotGeoObject class are listed here.

### Public Methods

	Name	Description
Initialize (see page 1616)		The function initializes internal data on the basis of the data defining the object. .

### I.11.3.1 Initialize

#### C++

```
HRESULT Initialize();
```

#### C#

```
public void Initialize();
```

#### Visual Basic

```
Public Sub Initialize()
```

#### Description

The function initializes internal data on the basis of the data defining the object. .

## I.12 IRobotGeoPolyline

### Class Hierarchy

**C++**

```
interface IRobotGeoPolyline : IRobotGeoObject;
```

**C#**

```
public interface IRobotGeoPolyline : IRobotGeoObject;
```

**Visual Basic**

```
Public Interface IRobotGeoPolyline
```

**Description**

Interface describing a polyline.

**I.12.1 IRobotGeoPolyline Members**

The following tables list the members exposed by IRobotGeoPolyline.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Segments (see page 1617)	Collection of segments describing a polyline.
◆	Type (see page 1615)	Object type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add (see page 1617)	The function adds a new segment to the polyline. .
◆	Clear (see page 1617)	The function clears a collection of segments of a polyline.
◆	Initialize (see page 1616)	The function initializes internal data on the basis of the data defining the object. .

**I.12.2 IRobotGeoPolyline Fields**

The fields of the IRobotGeoPolyline class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Segments (see page 1617)	Collection of segments describing a polyline.

**I.12.2.1 Segments****C++**

```
HRESULT get_Segments(IRobotGeoSegmentCollection**);
```

**C#**

```
public IRobotGeoSegmentCollection Segments { get; }
```

**Visual Basic**

```
Public ReadOnly Segments As IRobotGeoSegmentCollection
```

**Description**

Collection of segments describing a polyline.

**I.12.3 IRobotGeoPolyline Methods**

The methods of the IRobotGeoPolyline class are listed here.

## Public Methods

	Name	Description
>Add (see page 1617)	Add (see page 1617)	The function adds a new segment to the polyline. .
Clear (see page 1617)	Clear (see page 1617)	The function clears a collection of segments of a polyline.

### I.12.3.1 Add

#### C++

```
HRESULT Add( IRobotGeoSegment* _segment );
```

#### C#

```
public void Add( IRobotGeoSegment _segment );
```

#### Visual Basic

```
Public Sub Add( ByRef _segment As IRobotGeoSegment )
```

#### Description

The function adds a new segment to the polyline. .

### I.12.3.2 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

The function clears a collection of segments of a polyline.

## I.13 IRobotGeoSegmentCollection

#### Class Hierarchy

#### C++

```
interface IRobotGeoSegmentCollection : IRobotCollection;
```

#### C#

```
public interface IRobotGeoSegmentCollection : IRobotCollection;
```

#### Visual Basic

```
Public Interface IRobotGeoSegmentCollection
```

#### Description

Collection of segments of geometrical objects.

## I.14 IRobotGeoContour

#### Class Hierarchy

**C++**

```
interface IRobotGeoContour : IRobotGeoObject;
```

**C#**

```
public interface IRobotGeoContour : IRobotGeoObject;
```

**Visual Basic**

```
Public Interface IRobotGeoContour
```

**Description**

Interface describing a contour..

**I.14.1 IRobotGeoContour Members**

The following tables list the members exposed by IRobotGeoContour.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Segments (see page 1619)	Collection of segments describing a contour (the contour is closed automatically; one does not have to define the last segment).
◆	Type (see page 1615)	Object type.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Add (see page 1619)	The function adds a new segment. The segment may be a line or an arc.
◆	Clear (see page 1619)	The function clears a segment collection .
◆	Initialize (see page 1616)	The function initializes internal data on the basis of the data defining the object. .

**I.14.2 IRobotGeoContour Fields**

The fields of the IRobotGeoContour class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Segments (see page 1619)	Collection of segments describing a contour (the contour is closed automatically; one does not have to define the last segment).

**I.14.2.1 Segments****C++**

```
HRESULT get_Segments(IRobotGeoSegmentCollection**);
```

**C#**

```
public IRobotGeoSegmentCollection Segments { get; }
```

**Visual Basic**

```
Public ReadOnly Segments As IRobotGeoSegmentCollection
```

**Description**

Collection of segments describing a contour (the contour is closed automatically; one does not have to define the last segment).

### I.14.3 IRobotGeoContour Methods

The methods of the IRobotGeoContour class are listed here.

#### Public Methods

	Name	Description
>Add ( <a href="#">see page 1619</a> )	Add ( <a href="#">see page 1619</a> )	The function adds a new segment. The segment may be a line or an arc.
Clear ( <a href="#">see page 1619</a> )	Clear ( <a href="#">see page 1619</a> )	The function clears a segment collection .

#### I.14.3.1 Add

##### C++

```
HRESULT Add( IRobotGeoSegment* _segment );
```

##### C#

```
public void Add( IRobotGeoSegment _segment );
```

##### Visual Basic

```
Public Sub Add(ByRef _segment As IRobotGeoSegment)
```

##### Description

The function adds a new segment. The segment may be a line or an arc.

#### I.14.3.2 Clear

##### C++

```
HRESULT Clear();
```

##### C#

```
public void Clear();
```

##### Visual Basic

```
Public Sub Clear()
```

##### Description

The function clears a segment collection .

### I.15 IRobotGeoArc

#### Class Hierarchy

##### C++

```
interface IRobotGeoArc : IRobotGeoObject;
```

##### C#

```
public interface IRobotGeoArc : IRobotGeoObject;
```

##### Visual Basic

```
Public Interface IRobotGeoArc
```

##### Description

Interface describing an arc.

## I.15.1 IRobotGeoArc Members

The following tables list the members exposed by IRobotGeoArc.

### Public Fields

	Name	Description
◆	P1 (see page 1621)	Initial point of the arc.
◆	P2 (see page 1621)	Final point of the arc.
◆	P3 (see page 1621)	A point along the arc.
◆	Type (see page 1615)	Object type.

### Public Methods

	Name	Description
◆	Initialize (see page 1616)	The function initializes internal data on the basis of the data defining the object. .

## I.15.2 IRobotGeoArc Fields

The fields of the IRobotGeoArc class are listed here.

### Public Fields

	Name	Description
◆	P1 (see page 1621)	Initial point of the arc.
◆	P2 (see page 1621)	Final point of the arc.
◆	P3 (see page 1621)	A point along the arc.

### I.15.2.1 P1

#### C++

```
HRESULT get_P1(IRobotGeoPoint3D**);
HRESULT put_P1(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D P1 { get; set; }
```

#### Visual Basic

```
Public P1 As IRobotGeoPoint3D
```

#### Description

Initial point of the arc.

### I.15.2.2 P2

#### C++

```
HRESULT get_P2(IRobotGeoPoint3D**);
HRESULT put_P2(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D P2 { get; set; }
```

#### Visual Basic

```
Public P2 As IRobotGeoPoint3D
```

#### Description

Final point of the arc.

### I.15.2.3 P3

#### C++

```
HRESULT get_P3(IRobotGeoPoint3D**);
HRESULT put_P3(IRobotGeoPoint3D*);
```

#### C#

```
public IRobotGeoPoint3D P3 { get; set; }
```

#### Visual Basic

```
Public P3 As IRobotGeoPoint3D
```

#### Description

A point along the arc.

## I.16 IRobotGeoCircle

#### Class Hierarchy

#### C++

```
interface IRobotGeoCircle : IRobotGeoObject;
```

#### C#

```
public interface IRobotGeoCircle : IRobotGeoObject;
```

#### Visual Basic

```
Public Interface IRobotGeoCircle
```

#### Description

Interface describing a circle.

### I.16.1 IRobotGeoCircle Members

The following tables list the members exposed by IRobotGeoCircle.

#### Public Fields

	Name	Description
◆	P1 (see page 1622)	A point belonging to a circle.
◆	P2 (see page 1622)	A point belonging to a circle.
◆	P3 (see page 1623)	A point belonging to a circle.
◆	PC (see page 1623)	The circle center.
◆	Type (see page 1615)	Object type.

#### Public Methods

	Name	Description
≡◆	Initialize (see page 1616)	The function initializes internal data on the basis of the data defining the object. .

### I.16.2 IRobotGeoCircle Fields

The fields of the IRobotGeoCircle class are listed here.

#### Public Fields

	Name	Description
◆	P1 (see page 1622)	A point belonging to a circle.

◆	P2 (see page 1622)	A point belonging to a circle.
◆	P3 (see page 1623)	A point belonging to a circle.
◆	PC (see page 1623)	The circle center.

### I.16.2.1 P1

**C++**

```
HRESULT get_P1(IRobotGeoPoint3D**);
HRESULT put_P1(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D P1 { get; set; }
```

**Visual Basic**

```
Public P1 As IRobotGeoPoint3D
```

**Description**

A point belonging to a circle.

### I.16.2.2 P2

**C++**

```
HRESULT get_P2(IRobotGeoPoint3D**);
HRESULT put_P2(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D P2 { get; set; }
```

**Visual Basic**

```
Public P2 As IRobotGeoPoint3D
```

**Description**

A point belonging to a circle.

### I.16.2.3 P3

**C++**

```
HRESULT get_P3(IRobotGeoPoint3D**);
HRESULT put_P3(IRobotGeoPoint3D*);
```

**C#**

```
public IRobotGeoPoint3D P3 { get; set; }
```

**Visual Basic**

```
Public P3 As IRobotGeoPoint3D
```

**Description**

A point belonging to a circle.

### I.16.2.4 PC

**C++**

```
HRESULT get_PC(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D PC { get; }
```

**Visual Basic**

```
Public ReadOnly PC As IRobotGeoPoint3D
```

**Description**

The circle center.

**I.17 IRobotGeoPoint3DCollection****Class Hierarchy****C++**

```
interface IRobotGeoPoint3DCollection : IRobotCollection;
```

**C#**

```
public interface IRobotGeoPoint3DCollection : IRobotCollection;
```

**Visual Basic**

```
Public Interface IRobotGeoPoint3DCollection
```

**Description**

Collection of points.

**I.18 IRobotGeoLayer****Class Hierarchy****C++**

```
interface IRobotGeoLayer : IDispatch;
```

**C#**

```
public interface IRobotGeoLayer;
```

**Visual Basic**

```
Public Interface IRobotGeoLayer
```

**Description**

Definition of a layer as a space sector. Layer is defined by specifying a cutting plane (by means of three points: P1 (see page 1625), P2 (see page 1625) and P3 (see page 1625)) and the fourth PDir (see page 1625) point determining a half-space direction. Additionally, layer thickness may be determined by specifying the Thickness (see page 1626) value. If the layer thickness is not specified, the layer describes the entire half-space.

**I.18.1 IRobotGeoLayer Members**

The following tables list the members exposed by IRobotGeoLayer.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	IsThickDefined (see page 1624)	Auxiliary flag indicating if the layer thickness has been defined Available since version 2.0.
◆	P1 (see page 1625)	First point (P1) defining the cutting plane Available since version 2.0.
◆	P2 (see page 1625)	Second point (P2) defining the cutting plane Available since version 2.0.
◆	P3 (see page 1625)	Third point (P3) defining the cutting plane Available since version 2.0.
◆	PDir (see page 1625)	Point defining direction (i.e. it indicates one of the half-spaces determined by the cutting plane) Available since version 2.0.

	Thickness ( <a href="#">see page 1626</a> )	Layer thickness Available since version 2.0.
-----------------------------------------------------------------------------------	---------------------------------------------	----------------------------------------------

## I.18.2 IRobotGeoLayer Fields

The fields of the IRobotGeoLayer class are listed here.

### Public Fields

	Name	Description
	IsThickDefined ( <a href="#">see page 1624</a> )	Auxiliary flag indicating if the layer thickness has been defined Available since version 2.0.
	P1 ( <a href="#">see page 1625</a> )	First point (P1) defining the cutting plane Available since version 2.0.
	P2 ( <a href="#">see page 1625</a> )	Second point (P2) defining the cutting plane Available since version 2.0.
	P3 ( <a href="#">see page 1625</a> )	Third point (P3) defining the cutting plane Available since version 2.0.
	PDir ( <a href="#">see page 1625</a> )	Point defining direction (i.e. it indicates one of the half-spaces determined by the cutting plane) Available since version 2.0.
	Thickness ( <a href="#">see page 1626</a> )	Layer thickness Available since version 2.0.

### I.18.2.1 IsThickDefined

#### C++

```
HRESULT get_IsThickDefined(VARIANT_BOOL* );
HRESULT put_IsThickDefined(VARIANT_BOOL);
```

#### C#

```
public bool IsThickDefined { get; set; }
```

#### Visual Basic

```
Public IsThickDefined As Boolean
```

#### Description

Auxiliary flag indicating if the layer thickness has been defined Available since version 2.0.

### I.18.2.2 P1

#### C++

```
HRESULT get_P1(IRobotGeoPoint3D** );
```

#### C#

```
public IRobotGeoPoint3D P1 { get; }
```

#### Visual Basic

```
Public ReadOnly P1 As IRobotGeoPoint3D
```

#### Description

First point (P1) defining the cutting plane Available since version 2.0.

### I.18.2.3 P2

#### C++

```
HRESULT get_P2(IRobotGeoPoint3D** );
```

#### C#

```
public IRobotGeoPoint3D P2 { get; }
```

#### Visual Basic

```
Public ReadOnly P2 As IRobotGeoPoint3D
```

**Description**

Second point (P2) defining the cutting plane Available since version 2.0.

**I.18.2.4 P3****C++**

```
HRESULT get_P3(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D P3 { get; }
```

**Visual Basic**

```
Public ReadOnly P3 As IRobotGeoPoint3D
```

**Description**

Third point (P3) defining the cutting plane Available since version 2.0.

**I.18.2.5 PDir****C++**

```
HRESULT get_PDir(IRobotGeoPoint3D**);
```

**C#**

```
public IRobotGeoPoint3D PDir { get; }
```

**Visual Basic**

```
Public ReadOnly PDir As IRobotGeoPoint3D
```

**Description**

Point defining direction (i.e. it indicates one of the half-spaces determined by the cutting plane) Available since version 2.0.

**I.18.2.6 Thickness****C++**

```
HRESULT get_Thickness(double*);  
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thickness As Double
```

**Description**

Layer thickness Available since version 2.0.

**I.19 IRobotGeoArcDefinitionMethod****C++**

```
enum IRobotGeoArcDefinitionMethod;
```

**C#**

```
public enum IRobotGeoArcDefinitionMethod;
```

**Visual Basic**

```
Public Enum IRobotGeoArcDefinitionMethod
```

**Members**

Members	Description
I_GADM_CENTER_BEGIN_END = 0	Available since version 3.
I_GADM_BEGIN_MIDDLE_END = 1	Available since version 3.

**Description**

Available methods of arc definition.

**Version**

Available since version 3.

**I.20 IRobotGeoCoordinateAxisSense****C++**

```
enum IRobotGeoCoordinateAxisSense;
```

**C#**

```
public enum IRobotGeoCoordinateAxisSense;
```

**Visual Basic**

```
Public Enum IRobotGeoCoordinateAxisSense
```

**Members**

Members	Description
I_GCAS_OX_PLUS = 1	Available since version 4.5.
I_GCAS_OY_PLUS = 2	Available since version 4.5.
I_GCAS_OZ_PLUS = 3	Available since version 4.5.
I_GCAS_OX_MINUS = -1	Available since version 4.5.
I_GCAS_OY_MINUS = -2	Available since version 4.5.
I_GCAS_OZ_MINUS = -3	Available since version 4.5.

**Description**

Senses of the coordinate system axes.

**Version**

Available since version 4.5.

**I.21 IRobotGeoPlane****C++**

```
enum IRobotGeoPlane;
```

**C#**

```
public enum IRobotGeoPlane;
```

**Visual Basic**

```
Public Enum IRobotGeoPlane
```

## Members

Members	Description
I_GP_XY = 0	Available since version 11.
I_GP_XZ = 1	Available since version 11.
I_GP_YZ = 2	Available since version 11.

## Version

Available since version 11.

# II IRobotValuesArray

## Class Hierarchy

### C++

```
interface IRobotValuesArray : IDispatch;
```

### C#

```
public interface IRobotValuesArray;
```

## Visual Basic

```
Public Interface IRobotValuesArray
```

## Description

Interface representing a one-dimensional table of real numbers.

## II.1 IRobotValuesArray Members

The following tables list the members exposed by IRobotValuesArray.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1628</a> )	Number of elements in the table (at the same time, it is the index of the last element in the table).

### Public Methods

	Name	Description
◆	Get ( <a href="#">see page 1629</a> )	The function returns the value of table element with the given index. Table elements are indexed from 1 to Count ( <a href="#">see page 1628</a> ).
◆	Set ( <a href="#">see page 1629</a> )	The function sets the value of table element with the given index. Table elements are indexed from 1 to Count ( <a href="#">see page 1628</a> ).
◆	SetSize ( <a href="#">see page 1629</a> )	The function sets the table size. Table elements are indexed from 1 to table size.

## II.2 IRobotValuesArray Fields

The fields of the IRobotValuesArray class are listed here.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1628</a> )	Number of elements in the table (at the same time, it is the index of the last element in the table).

## II.2.1 Count

### C++

```
HRESULT get_Count(long*);
```

### C#

```
public long Count { get; }
```

### Visual Basic

```
Public ReadOnly Count As long
```

### Description

Number of elements in the table (at the same time, it is the index of the last element in the table).

## II.3 IRobotValuesArray Methods

The methods of the IRobotValuesArray class are listed here.

### Public Methods

	Name	Description
💡	Get (see page 1629)	The function returns the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1628).
💡	Set (see page 1629)	The function sets the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1628).
💡	SetSize (see page 1629)	The function sets the table size. Table elements are indexed from 1 to table size.

### II.3.1 Get

#### C++

```
HRESULT Get(long _idx, double* ret);
```

#### C#

```
public double Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As double
```

#### Description

The function returns the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1628).

### II.3.2 Set

#### C++

```
HRESULT Set(long _idx, double _val);
```

#### C#

```
public void Set(long _idx, double _val);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _val As double)
```

## Description

The function sets the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1628).

### II.3.3 SetSize

#### C++

```
HRESULT SetSize(long _count);
```

#### C#

```
public void SetSize(long _count);
```

#### Visual Basic

```
Public Sub SetSize(_count As long)
```

## Description

The function sets the table size. Table elements are indexed from 1 to table size.

## III IRobotNamesArray

### Class Hierarchy

#### C++

```
interface IRobotNamesArray : IDispatch;
```

#### C#

```
public interface IRobotNamesArray;
```

#### Visual Basic

```
Public Interface IRobotNamesArray
```

## Description

The interface describing a one-dimensional table of names represented by character strings.

### III.1 IRobotNamesArray Members

The following tables list the members exposed by IRobotNamesArray.

#### Public Fields

	Name	Description
◆	Count (see page 1630)	Number of elements in the table (it is also the index of the last element in the table).

#### Public Methods

	Name	Description
◆	Find (see page 1631)	Function returns index of the name specified in the table. Table elements are searched beginning with the determined start index. If the specified name is not found, function returns value -1..
◆	Get (see page 1631)	The function returns table element with the given index. Table element are indexed from 1 to Count (see page 1630)..
◆	Set (see page 1631)	The function sets the value of the given table element. The table elements are indexed from 1 to Count (see page 1630).

	SetSize (see page 1632)	The function sets a new table size. Table elements are indexed from 1 to Count (see page 1630). .
-----------------------------------------------------------------------------------	-------------------------	---------------------------------------------------------------------------------------------------

## III.2 IRobotNamesArray Fields

The fields of the IRobotNamesArray class are listed here.

### Public Fields

	Name	Description
	Count (see page 1630)	Number of elements in the table (it is also the index of the last element in the table).

### III.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of elements in the table (it is also the index of the last element in the table).

## III.3 IRobotNamesArray Methods

The methods of the IRobotNamesArray class are listed here.

### Public Methods

	Name	Description
	Find (see page 1631)	Function returns index of the name specified in the table. Table elements are searched beginning with the determined start index. If the specified name is not found, function returns value -1. .
	Get (see page 1631)	The function returns table element with the given index. Table element are indexed from 1 to Count (see page 1630). .
	Set (see page 1631)	The function sets the value of the given table element. The table elements are indexed from 1 to Count (see page 1630).
	SetSize (see page 1632)	The function sets a new table size. Table elements are indexed from 1 to Count (see page 1630). .

### III.3.1 Find

#### C++

```
HRESULT Find(BSTR _to_find, long _start_idx, long* ret);
```

#### C#

```
public long Find(String _to_find, long _start_idx);
```

#### Visual Basic

```
Public Function Find(_to_find As String, _start_idx As long) As long
```

#### Description

Function returns index of the name specified in the table. Table elements are searched beginning with the determined start

index. If the specified name is not found, function returns value -1..

#### Version

Available since version 3.5.

### III.3.2 Get

#### C++

```
HRESULT Get(long _idx, BSTR* ret);
```

#### C#

```
public String Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As String
```

#### Description

The function returns table element with the given index. Table element are indexed from 1 to Count (see page 1630)..

### III.3.3 Set

#### C++

```
HRESULT Set(long _idx, BSTR _text);
```

#### C#

```
public void Set(long _idx, String _text);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _text As String)
```

#### Description

The function sets the value of the given table element. The table elements are indexed from 1 to Count (see page 1630).

### III.3.4 SetSize

#### C++

```
HRESULT SetSize(long _count);
```

#### C#

```
public void SetSize(long _count);
```

#### Visual Basic

```
Public Sub SetSize(_count As long)
```

#### Description

The function sets a new table size. Table elements are indexed from 1 to Count (see page 1630)..

## IV IRobotNumbersArray

#### Class Hierarchy

#### C++

```
interface IRobotNumbersArray : IDispatch;
```

**C#**

```
public interface IRobotNumbersArray;
```

**Visual Basic**

```
Public Interface IRobotNumbersArray
```

**Description**

The interface representing a table of integers. Table elements are indexed from 1 to Count (see page 1633). .

## IV.1 IRobotNumbersArray Members

The following tables list the members exposed by IRobotNumbersArray.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1633)	Table size (at the same time, this is the index of the last element in the table) .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Get (see page 1633)	The function returns a table element with the indicated index. Table elements are indexed from 1 to Count (see page 1633). .
◆	Set (see page 1634)	The function sets the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1633).
◆	SetSize (see page 1634)	The function sets a new table size. Table elements will be indexed from 1 to table size. .

## IV.2 IRobotNumbersArray Fields

The fields of the IRobotNumbersArray class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count (see page 1633)	Table size (at the same time, this is the index of the last element in the table) .

### IV.2.1 Count

**C++**

```
HRESULT get_Count( long* );
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Table size (at the same time, this is the index of the last element in the table) .

## IV.3 IRobotNumbersArray Methods

The methods of the IRobotNumbersArray class are listed here.

## Public Methods

	Name	Description
💡	Get (see page 1633)	The function returns a table element with the indicated index. Table elements are indexed from 1 to Count (see page 1633). .
💡	Set (see page 1634)	The function sets the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1633).
💡	SetSize (see page 1634)	The function sets a new table size. Table elements will be indexed from 1 to table size. .

### IV.3.1 Get

#### C++

```
HRESULT Get(long _idx, long* ret);
```

#### C#

```
public long Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As long
```

#### Description

The function returns a table element with the indicated index. Table elements are indexed from 1 to Count (see page 1633). .

### IV.3.2 Set

#### C++

```
HRESULT Set(long _idx, long _val);
```

#### C#

```
public void Set(long _idx, long _val);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _val As long)
```

#### Description

The function sets the value of table element with the given index. Table elements are indexed from 1 to Count (see page 1633).

### IV.3.3 SetSize

#### C++

```
HRESULT SetSize(long _count);
```

#### C#

```
public void SetSize(long _count);
```

#### Visual Basic

```
Public Sub SetSize(_count As long)
```

#### Description

The function sets a new table size. Table elements will be indexed from 1 to table size. .

## V IRobotObjectsArray

### Class Hierarchy

#### C++

```
interface IRobotObjectsArray : IDispatch;
```

#### C#

```
public interface IRobotObjectsArray;
```

### Visual Basic

```
Public Interface IRobotObjectsArray
```

### Description

The interface providing access to the table of objects .

## V.1 IRobotObjectsArray Members

The following tables list the members exposed by IRobotObjectsArray.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1635</a> )	Number of objects in the table .

### Public Methods

	Name	Description
◆	Get ( <a href="#">see page 1635</a> )	The function returns the indicated table element. .
◆	Set ( <a href="#">see page 1636</a> )	The function sets the indicated table element. .
◆	SetSize ( <a href="#">see page 1636</a> )	The function sets the table size. Table elements are indexed from 1 to table size. The table size is also the number of its elements. .

## V.2 IRobotObjectsArray Fields

The fields of the IRobotObjectsArray class are listed here.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1635</a> )	Number of objects in the table .

## V.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

### Visual Basic

```
Public ReadOnly Count As long
```

### Description

Number of objects in the table .

## V.3 IRobotObjectsArray Methods

The methods of the IRobotObjectsArray class are listed here.

### Public Methods

	Name	Description
➊	Get (⠁ see page 1635)	The function returns the indicated table element. .
➋	Set (⠁ see page 1636)	The function sets the indicated table element. .
➌	SetSize (⠁ see page 1636)	The function sets the table size. Table elements are indexed from 1 to table size. The table size is also the number of its elements. .

### V.3.1 Get

#### C++

```
HRESULT Get(long _idx, IDispatch* ret);
```

#### C#

```
public IDispatch Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As IDispatch
```

#### Description

The function returns the indicated table element. .

### V.3.2 Set

#### C++

```
HRESULT Set(long _idx, IDispatch _object);
```

#### C#

```
public void Set(long _idx, IDispatch _object);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _object As IDispatch)
```

#### Description

The function sets the indicated table element. .

### V.3.3 SetSize

#### C++

```
HRESULT SetSize(long _count);
```

#### C#

```
public void SetSize(long _count);
```

#### Visual Basic

```
Public Sub SetSize(_count As long)
```

#### Description

The function sets the table size. Table elements are indexed from 1 to table size. The table size is also the number of its elements. .

## VI IRobotNumbersCollection

### Class Hierarchy

#### C++

```
interface IRobotNumbersCollection : IDispatch;
```

#### C#

```
public interface IRobotNumbersCollection;
```

### Visual Basic

```
Public Interface IRobotNumbersCollection
```

### Description

Collection of numbers (integers).

## VI.1 IRobotNumbersCollection Members

The following tables list the members exposed by IRobotNumbersCollection.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1637</a> )	Number of elements in the collection .

### Public Methods

	Name	Description
✿	Get ( <a href="#">see page 1637</a> )	The function returns the required collection element. The elements are indexed from 1 to Count ( <a href="#">see page 1637</a> ).

## VI.2 IRobotNumbersCollection Fields

The fields of the IRobotNumbersCollection class are listed here.

### Public Fields

	Name	Description
◆	Count ( <a href="#">see page 1637</a> )	Number of elements in the collection .

### VI.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

### Visual Basic

```
Public ReadOnly Count As long
```

### Description

Number of elements in the collection .

## VI.3 IRobotNumbersCollection Methods

The methods of the IRobotNumbersCollection class are listed here.

### Public Methods

	Name	Description
💡	Get (see page 1637)	The function returns the required collection element. The elements are indexed from 1 to Count (see page 1637).

### VI.3.1 Get

#### C++

```
HRESULT Get(long _idx, long* ret);
```

#### C#

```
public long Get(long _idx);
```

#### Visual Basic

```
Public Function Get(_idx As long) As long
```

#### Description

The function returns the required collection element. The elements are indexed from 1 to Count (see page 1637).

## VII IRobotCodeRegistrar

### Class Hierarchy

#### C++

```
interface IRobotCodeRegistrar : IDispatch;
```

#### C#

```
public interface IRobotCodeRegistrar;
```

#### Visual Basic

```
Public Interface IRobotCodeRegistrar
```

#### Description

In order to simplify the process of registering external codes extending the Robot program, RobotCodeRegistrar has been defined.

## VII.1 IRobotCodeRegistrar Members

The following tables list the members exposed by IRobotCodeRegistrar.

### Public Fields

	Name	Description
💡	CodeName (see page 1639)	Code name .
💡	CodeType (see page 1639)	Code type.
💡	Country (see page 1639)	Name of a country where the code is implemented.
💡	Guid (see page 1640)	The text containing CLSID of the main external code implementation component (the field may be filled alternatively from ProgId (see page 1640)).

◆	Manufacturer (see page 1640)	Name of the producer/dealer of the given code implementation.
◆	ProgId (see page 1640)	The text containing ProgID of the main external code implementation component (the field may be set alternatively with the Guid (see page 1640) field).

**Public Methods**

	Name	Description
+=	Register (see page 1641)	The function registers code implementation on the basis of the set values. .
+=	Unregister (see page 1641)	The function unregisters the code implementation on the basis of the set values.

**VII.2 IRobotCodeRegistrar Fields**

The fields of the IRobotCodeRegistrar class are listed here.

**Public Fields**

	Name	Description
◆	CodeName (see page 1639)	Code name .
◆	CodeType (see page 1639)	Code type.
◆	Country (see page 1639)	Name of a country where the code is implemented.
◆	Guid (see page 1640)	The text containing CLSID of the main external code implementation component (the field may be filled alternatively from ProgId (see page 1640)).
◆	Manufacturer (see page 1640)	Name of the producer/dealer of the given code implementation.
◆	ProgId (see page 1640)	The text containing ProgID of the main external code implementation component (the field may be set alternatively with the Guid (see page 1640) field).

**VII.2.1 CodeName****C++**

```
HRESULT get_CodeName(BSTR* );
HRESULT put_CodeName(BSTR);
```

**C#**

```
public String CodeName { get; set; }
```

**Visual Basic**

```
Public CodeName As String
```

**Description**

Code name .

**VII.2.2 CodeType****C++**

```
HRESULT get_CodeType(IRobotCodeType* );
HRESULT put_CodeType(IRobotCodeType);
```

**C#**

```
public IRobotCodeType CodeType { get; set; }
```

**Visual Basic**

```
Public CodeType As IRobotCodeType
```

**Description**

Code type.

**VII.2.3 Country****C++**

```
HRESULT get_Country(BSTR* );
HRESULT put_Country(BSTR);
```

**C#**

```
public String Country { get; set; }
```

**Visual Basic**

```
Public Country As String
```

**Description**

Name of a country where the code is implemented.

**Version**

Available since version 12.

**VII.2.4 Guid****C++**

```
HRESULT get_Guid(BSTR* );
HRESULT put_Guid(BSTR);
```

**C#**

```
public String Guid { get; set; }
```

**Visual Basic**

```
Public Guid As String
```

**Description**

The text containing CLSID of the main external code implementation component (the field may be filled alternatively from ProgId (see page 1640)).

**VII.2.5 Manufacturer****C++**

```
HRESULT get_Manufacturer(BSTR* );
HRESULT put_Manufacturer(BSTR);
```

**C#**

```
public String Manufacturer { get; set; }
```

**Visual Basic**

```
Public Manufacturer As String
```

**Description**

Name of the producer/dealer of the given code implementation.

**VII.2.6 ProgId****C++**

```
HRESULT get_ProgId(BSTR* );
```

```

HRESULT put_ProgId(BSTR);
C#
public String ProgId { get; set; }

```

**Visual Basic**

```
Public ProgId As String
```

**Description**

The text containing ProgID of the main external code implementation component (the field may be set alternatively with the Guid (see page 1640) field).

## VII.3 IRobotCodeRegistrar Methods

The methods of the IRobotCodeRegistrar class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	Register (see page 1641)	The function registers code implementation on the basis of the set values..
❖	Unregister (see page 1641)	The function unregisters the code implementation on the basis of the set values.

### VII.3.1 Register

**C++**

```
HRESULT Register();
```

**C#**

```
public void Register();
```

**Visual Basic**

```
Public Sub Register()
```

**Description**

The function registers code implementation on the basis of the set values..

### VII.3.2 Unregister

**C++**

```
HRESULT Unregister();
```

**C#**

```
public void Unregister();
```

**Visual Basic**

```
Public Sub Unregister()
```

**Description**

The function unregisters the code implementation on the basis of the set values.

## VIII IRobotWindow

**Class Hierarchy**

**C++**

```
interface IRobotWindow : IDispatch;
```

**C#**

```
public interface IRobotWindow;
```

**Visual Basic**

```
Public Interface IRobotWindow
```

**Description**

General interface describing a window in the Robot program. It describes both the main application window and windows which comprise individual views. .

## VIII.1 IRobotWindow Members

The following tables list the members exposed by IRobotWindow.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Caption ( [ see page 1642)	
❖	Handle ( [ see page 1642)	System window handle Available since version 1.7.
❖	IsActive ( [ see page 1643)	Indicator of window activity Available since version 1.7.
❖	State ( [ see page 1643)	State of the window Available since version 1.7.

**Public Methods**

	<b>Name</b>	<b>Description</b>
☞	Activate ( [ see page 1643)	Function results in activating the window. Available since version 1.7.
☞	SendMessage ( [ see page 1644)	The function allows to send a message to the window.

## VIII.2 IRobotWindow Fields

The fields of the IRobotWindow class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Caption ( [ see page 1642)	
❖	Handle ( [ see page 1642)	System window handle Available since version 1.7.
❖	IsActive ( [ see page 1643)	Indicator of window activity Available since version 1.7.
❖	State ( [ see page 1643)	State of the window Available since version 1.7.

### VIII.2.1 Caption

**C++**

```
HRESULT get_Caption(BSTR* );
HRESULT put_Caption(BSTR);
```

**C#**

```
public String Caption { get; set; }
```

**Visual Basic**

```
Public Caption As String
```

**Version**

Available since version 3.

**VIII.2.2 Handle****C++**

```
HRESULT get_Handle(long*);
```

**C#**

```
public long Handle { get; }
```

**Visual Basic**

```
Public ReadOnly Handle As long
```

**Description**

System window handle Available since version 1.7.

**VIII.2.3 IsActive****C++**

```
HRESULT get_IsActive(VARIANT_BOOL*);
```

**C#**

```
public bool IsActive { get; }
```

**Visual Basic**

```
Public ReadOnly IsActive As Boolean
```

**Description**

Indicator of window activity Available since version 1.7.

**VIII.2.4 State****C++**

```
HRESULT get_State(IRobotWindowState*);  
HRESULT put_State(IRobotWindowState*);
```

**C#**

```
public IRobotWindowState State { get; set; }
```

**Visual Basic**

```
Public State As IRobotWindowState
```

**Description**

State of the window Available since version 1.7.

**VIII.3 IRobotWindow Methods**

The methods of the IRobotWindow class are listed here.

**Public Methods**

	Name	Description
💡	Activate (↗ see page 1643)	Function results in activating the window. Available since version 1.7.
💡	SendMessage (↗ see page 1644)	The function allows to send a message to the window.

### VIII.3.1 Activate

**C++**

```
HRESULT Activate();
```

**C#**

```
public void Activate();
```

**Visual Basic**

```
Public Sub Activate()
```

**Description**

Function results in activating the window. Available since version 1.7.

### VIII.3.2 SendMessage

**C++**

```
HRESULT SendMessage(long _msg_id, long _w_param, long _l_param);
```

**C#**

```
public void SendMessage(long _msg_id, long _w_param, long _l_param);
```

**Visual Basic**

```
Public Sub SendMessage(_msg_id As Long, _w_param As Long, _l_param As Long)
```

**Description**

The function allows to send a message to the window.

**Version**

Available since version 3.

## IX IRobotWindowState

**C++**

```
enum IRobotWindowState;
```

**C#**

```
public enum IRobotWindowState;
```

**Visual Basic**

```
Public Enum IRobotWindowState
```

**Members**

Members	Description
I_WS_NORMAL = 0	Available since version 1.7.
I_WS_MAXIMIZE = 1	Available since version 1.7.
I_WS_MINIMIZE = 2	Available since version 1.7.

**Description**

Available window states.

## X IRobotProtectionInfo

### Class Hierarchy

#### C++

```
interface IRobotProtectionInfo : IDispatch;
```

#### C#

```
public interface IRobotProtectionInfo;
```

### Visual Basic

```
Public Interface IRobotProtectionInfo
```

### Description

Object provides access to information about protection of the Robot program and its extensions.

## X.1 IRobotProtectionInfo Members

The following tables list the members exposed by IRobotProtectionInfo.

### Public Methods

	Name	Description
💡	IsExtensionEnabled (see page 1645)	Function returns value different from zero (True), if the external module (application) - Robot program extension - has been activated in the Robot program protection system. Available since version 2.0.

## X.2 IRobotProtectionInfo Methods

The methods of the IRobotProtectionInfo class are listed here.

### Public Methods

	Name	Description
💡	IsExtensionEnabled (see page 1645)	Function returns value different from zero (True), if the external module (application) - Robot program extension - has been activated in the Robot program protection system. Available since version 2.0.

### X.2.1 IsExtensionEnabled

#### C++

```
HRESULT IsExtensionEnabled(long _ext_num, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsExtensionEnabled(long _ext_num);
```

### Visual Basic

```
Public Function IsExtensionEnabled(_ext_num As long) As Boolean
```

### Description

Function returns value different from zero (True), if the external module (application) - Robot program extension - has been activated in the Robot program protection system. Available since version 2.0.

## XI IRobotComponentFactory

### Class Hierarchy

#### C++

```
interface IRobotComponentFactory : IDispatch;
```

#### C#

```
public interface IRobotComponentFactory;
```

### Visual Basic

```
Public Interface IRobotComponentFactory
```

### Description

Auxiliary component factory. It should be applied instead of the standard New operator for creation of "free" components in case when the application is to cooperate with both the Robot program and Kernel, depending on the user selection. .

### Version

Available since version 2.5.

## XI.1 IRobotComponentFactory Members

The following tables list the members exposed by IRobotComponentFactory.

### Public Methods

	Name	Description
≡	Create (see page 1646)	Function creates a new component of the indicated type.
≡	CreateExt (see page 1646)	Function creates a new component supported by the indicated extension of the Robot program kernel.

## XI.2 IRobotComponentFactory Methods

The methods of the IRobotComponentFactory class are listed here.

### Public Methods

	Name	Description
≡	Create (see page 1646)	Function creates a new component of the indicated type.
≡	CreateExt (see page 1646)	Function creates a new component supported by the indicated extension of the Robot program kernel.

### XI.2.1 Create

#### C++

```
HRESULT Create( IRobotComponentType _cmpnt_type, IDispatch* ret);
```

#### C#

```
public IDispatch Create( IRobotComponentType _cmpnt_type);
```

### Visual Basic

```
Public Function Create(_cmpnt_type As IRobotComponentType) As IDispatch
```

### Description

Function creates a new component of the indicated type.

**Version**

Available since version 2.5.

**XI.2.2 CreateExt****C++**

```
HRESULT CreateExt(BSTR _extmdl_name, long _cmpnt_type, IDispatch* ret);
```

**C#**

```
public IDispatch CreateExt(String _extmdl_name, long _cmpnt_type);
```

**Visual Basic**

```
Public Function CreateExt(_extmdl_name As String, _cmpnt_type As Long) As IDispatch
```

**Description**

Function creates a new component supported by the indicated extension of the Robot program kernel.

**Version**

Available since version 2.5.

**XII IRobotComponentType****C++**

```
enum IRobotComponentType;
```

**C#**

```
public enum IRobotComponentType;
```

**Visual Basic**

```
Public Enum IRobotComponentType
```

**Members**

Members	Description
I_CT_VALUES_ARRAY = 1	Available since version 2.5.
I_CT_NUMBERS_ARRAY = 2	Available since version 2.5.
I_CT_NAMES_ARRAY = 3	Available since version 2.5.
I_CT_OBJECTS_ARRAY = 4	Available since version 2.5.
I_CT_GEO_POINT_2D = 5	Available since version 2.5.
I_CT_GEO_POINT_3D = 6	Available since version 2.5.
I_CT_GEO_CURVE_DIV = 7	Available since version 2.5.
I_CT_GEO_SEGMENT = 8	Available since version 2.5.
I_CT_GEO_SEGMENT_LINE = 9	Available since version 2.5.
I_CT_GEO_SEGMENT_ARC = 10	Available since version 2.5.
I_CT_GEO_OBJECT = 11	Available since version 2.5.
I_CT_GEO_POLYLINE = 12	Available since version 2.5.
I_CT_GEO_CONTOUR = 13	Available since version 2.5.
I_CT_GEO_ARC = 14	Available since version 2.5.
I_CT_GEO_CIRCLE = 15	Available since version 2.5.
I_CT_GEO_SEGMENT_COLLECTION = 16	Available since version 2.5.
I_CT_GEO_POINT_3D_COLLECTION = 17	Available since version 2.5.

I_CT_GEO_LAYER = 18	Available since version 2.5.
I_CT_EMITTER = 19	Available since version 2.5.
I_CT_JOINT_LOAD = 201	Available since version 2.5.
I_CT_JOINT_KNEE_LOAD = 202	Available since version 2.5.
I_CT_JOINT_ANGLE_LOAD = 203	Available since version 2.5.
I_CT_JOINT_TUBE_LOAD = 204	Available since version 2.5.
I_CT_JOINT_PINNED_LOAD = 205	Available since version 2.5.
I_CT_JOINT_FIXED_LOAD = 206	Available since version 2.5.
I_CT_JOINT_GUSSET_SIMPLE_LOAD = 207	Available since version 2.5.
I_CT_JOINT_GUSSET_CROSS_LOAD = 208	Available since version 2.5.
I_CT_JOINT_GUSSET_FLANGE_LOAD = 209	Available since version 2.5.
I_CT_MODIF_EXTRUSION = 29	Available since version 2.5.
I_CT_MODIF_LATHE = 30	Available since version 2.5.
I_CT_MODIF_PYRAMID = 31	Available since version 2.5.
I_CT_OPER_TRANSLATION = 32	Available since version 2.5.
I_CT_OPER_SCALING = 33	Available since version 2.5.
I_CT_OPER_ROTATION = 34	Available since version 2.5.
I_CT_OPER_MESHING = 35	Available since version 2.5.
I_CT_SPECTRAL_ANALYSIS_POINTS_COLLECTION = 36	Available since version 2.5.
I_CT_SPECTRAL_ANALYSIS_SPECTRUM = 37	Available since version 2.5.
I_CT_CASE_ANALYSIS_MODES_FILTER = 38	Available since version 2.5.
I_CT_FE_RESULT_PARAMS = 39	Object steering the parameters of getting results for finite elements. Available since version 2.5.
I_CT_RTF_VIEW = 40	View for RTF files. Available since version 2.5.
I_CT_POINTS_ARRAY = 41	Table of points in the 3D space (IRobotPointsArray (see page 1649)). Available since version 3.
I_CT_EXTREME_PARAMS = 42	Parameters of taking extreme values . Available since version 3.5.
I_CT_STRUCTURE_GEO_ANALYSER = 43	Structure geometry analyzer. Available since version 3.5.
I_CT_FE_EXTREME_PARAMS = 44	Available since version 4.
I_CT_FE_MULTI_RESULT_TYPE = 45	Available since version 4.1.
I_CT_SW_STRUCT3D_GEN_PARAMS = 46	Parameters of generation of a 3D structure model for snow/wind loads (IRobotSWStruct3DGenParams (see page 87) object type).
I_CT_SW_STRUCT3D_PURLIN_GEN_PARAMS = 47	IRobotSWStruct3DPurlinGenParams (see page 90) object type.
I_CT_SW_STRUCT3D = 48	3D structure model for generation of snow/wind loads (IRobotSWStruct3D (see page 82) object type).
I_CT_SW_STRUCT3D_FRAME = 49	IRobotSWStruct3DFrame (see page 85) object type.
I_CT_SW_STRUCT3D_ELEMENT = 50	IRobotSWStruct3DFrame (see page 85) object type.
I_CT_HTML_VIEW = 52	View for the HTML format files. Available since version 7.5.
I_CT_STRUCTURE_MERGE_DATA = 53	Available since version 8.
I_CT_NUMBERS_DICTIONARY = 54	Available since version 8.1.
I_CT_TABLE_SCREEN_CAPTURE_PARAMS = 55	Available since version 8.2.
I_CT_PARAM_DEF = 56	Definition of an external parameter. Available since version 8.2.

I_CTR_RESULT_QUERY_PARAMS = 57	Parameters of a query to the result server. Available since version 8.2.
--------------------------------	-----------------------------------------------------------------------------

**Description**

Available component types that can be created by means of the component factory.

**Version**

Available since version 2.5.

## XIII IRobotDegreeOfFreedom

**C++**

```
enum IRobotDegreeOfFreedom;
```

**C#**

```
public enum IRobotDegreeOfFreedom;
```

**Visual Basic**

```
Public Enum IRobotDegreeOfFreedom
```

**Members**

Members	Description
I_DOF_UX = 0	Available since version 2.5.
I_DOF_UY = 1	Available since version 2.5.
I_DOF_UZ = 2	Available since version 2.5.
I_DOF_RX = 3	Available since version 2.5.
I_DOF_RY = 4	Available since version 2.5.
I_DOF_RZ = 5	Available since version 2.5.

**Description**

Available degrees of freedom.

**Version**

Available since version 2.5.

## XIV IRobotPointsArray

**Class Hierarchy****C++**

```
interface IRobotPointsArray : IDispatch;
```

**C#**

```
public interface IRobotPointsArray;
```

**Visual Basic**

```
Public Interface IRobotPointsArray
```

**Description**

Table of points in the 3D space. .

**Version**

Available since version 3.

## XIV.1 IRobotPointsArray Members

The following tables list the members exposed by IRobotPointsArray.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1650</a> )	Number of elements (at the same time, it is the index of the last table element).

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Get ( <a href="#">see page 1650</a> )	Function takes point coordinates saved in the table under the indicated index. Table elements are indexed from 1 to Count ( <a href="#">see page 1650</a> ). .
◆	Set ( <a href="#">see page 1651</a> )	Function saves the specified point coordinates under the indicated index to the table. Table elements are indexed from 1 to Count ( <a href="#">see page 1650</a> ). .
◆	SetSize ( <a href="#">see page 1651</a> )	Function sets a table size.

## XIV.2 IRobotPointsArray Fields

The fields of the IRobotPointsArray class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 1650</a> )	Number of elements (at the same time, it is the index of the last table element).

### XIV.2.1 Count

**C++**

```
HRESULT get_Count(long*);
```

**C#**

```
public long Count { get; }
```

**Visual Basic**

```
Public ReadOnly Count As long
```

**Description**

Number of elements (at the same time, it is the index of the last table element).

**Version**

Available since version 3.

## XIV.3 IRobotPointsArray Methods

The methods of the IRobotPointsArray class are listed here.

## Public Methods

	Name	Description
💡	Get (see page 1650)	Function takes point coordinates saved in the table under the indicated index. Table elements are indexed from 1 to Count (see page 1650)..
💡	Set (see page 1651)	Function saves the specified point coordinates under the indicated index to the table. Table elements are indexed from 1 to Count (see page 1650).
💡	SetSize (see page 1651)	Function sets a table size.

### XIV.3.1 Get

#### C++

```
HRESULT Get(long _idx, double* _x, double* _y, double* _z);
```

#### C#

```
public void Get(long _idx, double* _x, double* _y, double* _z);
```

#### Visual Basic

```
Public Sub Get(_idx As long, ByRef _x As double*, ByRef _y As double*, ByRef _z As double*)
```

#### Description

Function takes point coordinates saved in the table under the indicated index. Table elements are indexed from 1 to Count (see page 1650). .

#### Version

Available since version 3.

### XIV.3.2 Set

#### C++

```
HRESULT Set(long _idx, double _x, double _y, double _z);
```

#### C#

```
public void Set(long _idx, double _x, double _y, double _z);
```

#### Visual Basic

```
Public Sub Set(_idx As long, _x As double, _y As double, _z As double)
```

#### Description

Function saves the specified point coordinates under the indicated index to the table. Table elements are indexed from 1 to Count (see page 1650).

#### Version

Available since version 3.

### XIV.3.3 SetSize

#### C++

```
HRESULT SetSize(long _size);
```

#### C#

```
public void SetSize(long _size);
```

**Visual Basic**

```
Public Sub SetSize(_size As long)
```

**Description**

Function sets a table size.

**Version**

Available since version 3.

## XV IRobotSerializable

**Class Hierarchy****C++**

```
interface IRobotSerializable : IDispatch;
```

**C#**

```
public interface IRobotSerializable;
```

**Visual Basic**

```
Public Interface IRobotSerializable
```

**Description**

Interface of the object whose data can be saved in/read from the stream.

**Version**

Available since version 8.

### XV.1 IRobotSerializable Members

The following tables list the members exposed by IRobotSerializable.

**Public Methods**

	Name	Description
≡	Read (↗ see page 1652)	Function reads object data from the specified stream.
≡	Write (↗ see page 1653)	Function saves object data in the specified stream.

### XV.2 IRobotSerializable Methods

The methods of the IRobotSerializable class are listed here.

**Public Methods**

	Name	Description
≡	Read (↗ see page 1652)	Function reads object data from the specified stream.
≡	Write (↗ see page 1653)	Function saves object data in the specified stream.

#### XV.2.1 Read

**C++**

```
HRESULT Read(IUnknown _source_stream, VARIANT_BOOL* ret);
```

**C#**

```
public bool Read(IUnknown _source_stream);
```

**Visual Basic**

```
Public Function Read(_source_stream As IUnknown) As Boolean
```

**Description**

Function reads object data from the specified stream.

**Version**

Available since version 8.

**XV.2.2 Write****C++**

```
HRESULT Write(IUnknown _dest_stream, VARIANT_BOOL* ret);
```

**C#**

```
public bool Write(IUnknown _dest_stream);
```

**Visual Basic**

```
Public Function Write(_dest_stream As IUnknown) As Boolean
```

**Description**

Function saves object data in the specified stream.

**Version**

Available since version 8.

**XVI IRobotNumbersDictionary****Class Hierarchy****C++**

```
interface IRobotNumbersDictionary : IDispatch;
```

**C#**

```
public interface IRobotNumbersDictionary;
```

**Visual Basic**

```
Public Interface IRobotNumbersDictionary
```

**Description**

Dictionary of pairs (integer, integer).

**Version**

Available since version 8.1.

**XVI.1 IRobotNumbersDictionary Members**

The following tables list the members exposed by IRobotNumbersDictionary.

**Public Fields**

	Name	Description
◆	Count (see page 1654)	Number of pairs defined in the dictionary.

## Public Methods

	Name	Description
✖	Delete (see page 1654)	Function deletes a pair with the specified key.
✖	Find (see page 1655)	Function returns an index of the pair with the specified key. If the pair is not found, function returns zero value.
✖	FindGet (see page 1655)	Function returns an index of the pair with the specified key and a value assigned to the key. If the pair is not found, function returns zero value.
✖	Get (see page 1655)	Function returns a pair of the specified index.
✖	Set (see page 1656)	Function adds a new pair to the dictionary. If a pair with the specified key already exists in the dictionary, its value will be replaced.

## XVI.2 IRobotNumbersDictionary Fields

The fields of the IRobotNumbersDictionary class are listed here.

### Public Fields

	Name	Description
❖	Count (see page 1654)	Number of pairs defined in the dictionary.

### XVI.2.1 Count

#### C++

```
HRESULT get_Count(long*);
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of pairs defined in the dictionary.

#### Version

Available since version 8.1.

## XVI.3 IRobotNumbersDictionary Methods

The methods of the IRobotNumbersDictionary class are listed here.

### Public Methods

	Name	Description
✖	Delete (see page 1654)	Function deletes a pair with the specified key.
✖	Find (see page 1655)	Function returns an index of the pair with the specified key. If the pair is not found, function returns zero value.
✖	FindGet (see page 1655)	Function returns an index of the pair with the specified key and a value assigned to the key. If the pair is not found, function returns zero value.
✖	Get (see page 1655)	Function returns a pair of the specified index.
✖	Set (see page 1656)	Function adds a new pair to the dictionary. If a pair with the specified key already exists in the dictionary, its value will be replaced.

### XVI.3.1 Delete

**C++**

```
HRESULT Delete(long _key);
```

**C#**

```
public void Delete(long _key);
```

**Visual Basic**

```
Public Sub Delete(_key As long)
```

**Description**

Function deletes a pair with the specified key.

**Version**

Available since version 8.1.

### XVI.3.2 Find

**C++**

```
HRESULT Find(long _key, long* ret);
```

**C#**

```
public long Find(long _key);
```

**Visual Basic**

```
Public Function Find(_key As long) As long
```

**Description**

Function returns an index of the pair with the specified key. If the pair is not found, function returns zero value.

**Version**

Available since version 8.1.

### XVI.3.3 FindGet

**C++**

```
HRESULT FindGet(long _key, long* _value, long* ret);
```

**C#**

```
public long FindGet(long _key, long* _value);
```

**Visual Basic**

```
Public Function FindGet(_key As long, ByRef _value As long*) As long
```

**Description**

Function returns an index of the pair with the specified key and a value assigned to the key. If the pair is not found, function returns zero value.

**Version**

Available since version 8.1.

## XVI.3.4 Get

### C++

```
HRESULT Get(long _idx, long* _key, long* _value);
```

### C#

```
public void Get(long _idx, long* _key, long* _value);
```

### Visual Basic

```
Public Sub Get(_idx As long, ByRef _key As long*, ByRef _value As long*)
```

### Description

Function returns a pair of the specified index.

### Version

Available since version 8.1.

## XVI.3.5 Set

### C++

```
HRESULT Set(long _key, long _value);
```

### C#

```
public void Set(long _key, long _value);
```

### Visual Basic

```
Public Sub Set(_key As long, _value As long)
```

### Description

Function adds a new pair to the dictionary. If a pair with the specified key already exists in the dictionary, its value will be replaced.

### Version

Available since version 8.1.

# Concrete

### Enumerations

	Name	Description
	IRConcrBarSectionGeometryType ( <a href="#">see page 1790</a> )	Types of geometry of an RC member section .

### Interfaces

	Name	Description
	IRConcrBarSectionData ( <a href="#">see page 1786</a> )	A ( <a href="#">see page 1787</a> ) simplified interface describing an RC member section.
	IRConcrCalcEngine ( <a href="#">see page 1790</a> )	RC calculation module.

# Supplementing the required reinforcement module with new codes

## Enumerations

	Name	Description
☞	IRBestCalcParamsDataDoubleValue (☞ see page 1669)	A set of parameter identifiers of real number type for the RBestCalcParamsData interface. The values of the set may be used as arguments defining the parameter type for the following functions of RBestCalcParamsData interface: SetDouble and GetDouble.. .
☞	IRBestCalcParamsDataIntegerValue (☞ see page 1670)	An identifier set for integer type parameters for the RBestCalcParamsData interface.
☞	IRBestCalcParamsDataStringValue (☞ see page 1671)	The identifier set for text values transferred for the RBestCalcParamsData interface.
☞	IRBestMemberDataDoubleValue (☞ see page 1674)	A set of identifiers of real number parameters for the RBestMemberData interface.
☞	IRBestMemberDataIntegerValue (☞ see page 1675)	An identifier set of integer type parameters for the BestMemberData interface.. .
☞	IRBestResultsDoubleValue (☞ see page 1676)	A set of identifiers of real-number parameters for the RBestResults interface.. .
☞	IRBestResultsIntegerValue (☞ see page 1677)	A set of identifiers of real-number parameters for the interface RBestResults.
☞	IRBestCalcErrors (☞ see page 1677)	The set of identifiers describing errors in calculations of theoretical reinforcement. If there occurred an error during calculations, the information about this fact should be written in the object of the RBestResults type, transferred as a parameter of the GetResults function of the RBestCodeCalcEngine interface. To write the information, one should use the SetInteger function that introduces integral values into an object. As an argument determining the type of integral value, one should provide the relevant identifier of error type, while as the error value, one should provide the relevant value from the set RBestCalcErrors. .
☞	IRBestCalcWarnings (☞ see page 1678)	Warning bit masks for calculations of theoretical reinforcement. .
☞	IRBestResultsStringValue (☞ see page 1678)	Set of identifiers of text parameters for the interface RBestResults.
☞	IRBestDimParamsIntegerValue (☞ see page 1681)	Available identifiers of parameters of an integer type for the RBestDimParams interface.
☞	IRBestMemberDataStringValue (☞ see page 1685)	The set of identifiers of text parameters for the RBestMemberData interface.. .
☞	IRBestMemberType (☞ see page 1685)	A set of identifiers for a type of RC member or plate.
☞	IRBestCodeCalculationType (☞ see page 1687)	
☞	IRBestCalculationType (☞ see page 1688)	
☞	IRBestDimParamsDoubleValue (☞ see page 1689)	
☞	IRBestBendType (☞ see page 1689)	
☞	IRBestDirection (☞ see page 1689)	
☞	IRBestLevel (☞ see page 1689)	
☞	IRBestForceDataDoubleValue (☞ see page 1690)	
☞	IRBestCoordSystem (☞ see page 1690)	

	IRBestForceDataIntegerValue (see page 1690)	
	IRBestForceDataSLSCombType (see page 1691)	Available since version 11.0.

## Interfaces

	Name	Description
	IRBestCodeService (see page 1658)	RBestCodeService makes code-dependent definitions and calculations for theoretical reinforcement of RC members available.
	IRBestCalcParamsDlg (see page 1660)	The interface describes the dialog box that allows the parameter definition for theoretical reinforcement calculations for RC members. .
	IRBestMemberDlg (see page 1661)	The interface describes the dialog box that makes a code-dependent definition of a RC member type possible.
	IRBestParamSet (see page 1662)	The interface describes a set of parameters that can be divided into two categories: values with a well defined meaning independent from code and these ones that meaning is connected with the given code. All values are available by the function set of the type Get/Set. The identifier of the taken or set quantity is their parameter. Code-dependent parameters have identifiers beginning from USER_VALUE upwards.
	IRBestCalcParamsData (see page 1668)	The interface defines calculation parameters of theoretical reinforcement for an RC member. Available parameters are described by the following types: RBestCalcParamsDataDoubleValue, RBestCalcParamsDataIntegerValue and RBestCalcParamsDataStringValue.
	IRBestCalcParamsDataList (see page 1671)	An object list of the BestCalcParamsData type. Each list element is associated with a label name. All the names are saved in the LabNames (see page 1672) table.
	IRBestMemberData (see page 1674)	The interface describes a data set for a type of RC member. Accessible parameters are described by the following types: RBestMemberDataDoubleValue, RBestMemberDataIntegerValue and RBestMemberDataStringValue. .
	IRBestResults (see page 1675)	The interface describes calculation results of theoretical reinforcement for single section of RC member. Available values are described by the types: RBestResultsDoubleValue, RBestResultsIntegerValue and RBestResultsStringValue.
	IRBestForceData (see page 1679)	A set of forces acting in a bar section. .
	IRBestDimParams (see page 1681)	A set of additional values parametrizing code (design) calculations. The RBestDimParamsIntegerValue type describes the set of available parameters. .
	IRBestCodeCalcEngine (see page 1682)	The interface describes the object performing code calculations of theoretical reinforcement for RC members.
	IRBestPlateCalcParamsDlg (see page 1686)	
	IRBestCodeServiceExt (see page 1687)	Available since version 2.0.
	IRBestCodeService2 (see page 1691)	

## I.1 IRBestCodeService

### Class Hierarchy

#### C++

```
interface IRBestCodeService : IDispatch;
```

**C#**

```
public interface IRBestCodeService;
```

**Visual Basic**

```
Public Interface IRBestCodeService
```

**Description**

RBESTCODESERVICE makes code-dependent definitions and calculations for theoretical reinforcement of RC members available.

**I.1.1 IRBestCodeService Members**

The following tables list the members exposed by IRBestCodeService.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CalcEngine ( [ see page 1659) )	The object performing code calculations.
◆	CalcParamsDlg ( [ see page 1659) )	A dialog box that allows to edit calculation parameters.
◆	MemberDlg ( [ see page 1659) )	A dialog box that allows to edit RC member parameters .

**I.1.2 IRBestCodeService Fields**

The fields of the IRBestCodeService class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CalcEngine ( [ see page 1659) )	The object performing code calculations.
◆	CalcParamsDlg ( [ see page 1659) )	A dialog box that allows to edit calculation parameters.
◆	MemberDlg ( [ see page 1659) )	A dialog box that allows to edit RC member parameters .

**I.1.2.1 CalcEngine****C++**

```
HRESULT get_CalcEngine( IRBestCodeCalcEngine** );
```

**C#**

```
public IRBestCodeCalcEngine CalcEngine { get; }
```

**Visual Basic**

```
Public ReadOnly CalcEngine As IRBestCodeCalcEngine
```

**Description**

The object performing code calculations.

**I.1.2.2 CalcParamsDlg****C++**

```
HRESULT get_CalcParamsDlg( IRBestCalcParamsDlg** );
```

**C#**

```
public IRBestCalcParamsDlg CalcParamsDlg { get; }
```

**Visual Basic**

```
Public ReadOnly CalcParamsDlg As IRBestCalcParamsDlg
```

## Description

A dialog box that allows to edit calculation parameters.

### I.1.2.3 MemberDlg

#### C++

```
HRESULT get_MemberDlg( IRBestMemberDlg** );
```

#### C#

```
public IRBestMemberDlg MemberDlg { get; }
```

#### Visual Basic

```
Public ReadOnly MemberDlg As IRBestMemberDlg
```

#### Description

A dialog box that allows to edit RC member parameters .

## I.2 IRBestCalcParamsDlg

#### Class Hierarchy

#### C++

```
interface IRBestCalcParamsDlg : IDispatch;
```

#### C#

```
public interface IRBestCalcParamsDlg;
```

#### Visual Basic

```
Public Interface IRBestCalcParamsDlg
```

#### Description

The interface describes the dialog box that allows the parameter definition for theoretical reinforcement calculations for RC members. .

### I.2.1 IRBestCalcParamsDlg Members

The following tables list the members exposed by IRBestCalcParamsDlg.

#### Public Methods

	Name	Description
≡♥	DoModal ( <a href="#">see page 1660</a> )	The function creates and displays a new modal dialog box that allows to edit the list of objects describing calculation parameters.
≡♥	SetStandard ( <a href="#">see page 1661</a> )	The function sets standard (default) parameter values for the object. .

### I.2.2 IRBestCalcParamsDlg Methods

The methods of the IRBestCalcParamsDlg class are listed here.

#### Public Methods

	Name	Description
≡♥	DoModal ( <a href="#">see page 1660</a> )	The function creates and displays a new modal dialog box that allows to edit the list of objects describing calculation parameters.
≡♥	SetStandard ( <a href="#">see page 1661</a> )	The function sets standard (default) parameter values for the object. .

### I.2.2.1 DoModal

**C++**

```
HRESULT DoModal(IRBestCalcParamsDataList* _param_list);
```

**C#**

```
public void DoModal(IRBestCalcParamsDataList _param_list);
```

**Visual Basic**

```
Public Sub DoModal(ByRef _param_list As IRBestCalcParamsDataList)
```

#### Description

The function creates and displays a new modal dialog box that allows to edit the list of objects describing calculation parameters.

### I.2.2.2 SetStandard

**C++**

```
HRESULT SetStandard(IRBestCalcParamsData* _params);
```

**C#**

```
public void SetStandard(IRBestCalcParamsData _params);
```

**Visual Basic**

```
Public Sub SetStandard(ByRef _params As IRBestCalcParamsData)
```

#### Description

The function sets standard (default) parameter values for the object. .

## I.3 IRBestMemberDlg

### Class Hierarchy

**C++**

```
interface IRBestMemberDlg : IDispatch;
```

**C#**

```
public interface IRBestMemberDlg;
```

**Visual Basic**

```
Public Interface IRBestMemberDlg
```

#### Description

The interface describes the dialog box that makes a code-dependent definition of a RC member type possible.

### I.3.1 IRBestMemberDlg Members

The following tables list the members exposed by IRBestMemberDlg.

#### Public Methods

	Name	Description
	DoModal (see page 1662)	The function creates and displays a modal dialog box that allows to edit the bar code parameters saved in a given object.

	SetStandard ( <a href="#">see page 1662</a> )	The function sets standard (default) values in the indicated object for the code-dependent parameters.
-----------------------------------------------------------------------------------	-----------------------------------------------	--------------------------------------------------------------------------------------------------------

## I.3.2 IRBestMemberDlg Methods

The methods of the IRBestMemberDlg class are listed here.

### Public Methods

	Name	Description
	DoModal ( <a href="#">see page 1662</a> )	The function creates and displays a modal dialog box that allows to edit the bar code parameters saved in a given object.
	SetStandard ( <a href="#">see page 1662</a> )	The function sets standard (default) values in the indicated object for the code-dependent parameters.

### I.3.2.1 DoModal

#### C++

```
HRESULT DoModal( IRBestMemberData* __params );
```

#### C#

```
public void DoModal( IRBestMemberData __params );
```

#### Visual Basic

```
Public Sub DoModal( ByRef __params As IRBestMemberData )
```

#### Description

The function creates and displays a modal dialog box that allows to edit the bar code parameters saved in a given object.

### I.3.2.2 SetStandard

#### C++

```
HRESULT SetStandard( IRBestMemberData* __params );
```

#### C#

```
public void SetStandard( IRBestMemberData __params );
```

#### Visual Basic

```
Public Sub SetStandard( ByRef __params As IRBestMemberData )
```

#### Description

The function sets standard (default) values in the indicated object for the code-dependent parameters.

## I.4 IRBestParamSet

### Class Hierarchy

#### C++

```
interface IRBestParamSet : IDispatch;
```

#### C#

```
public interface IRBestParamSet;
```

#### Visual Basic

```
Public Interface IRBestParamSet
```

## Description

The interface describes a set of parameters that can be divided into two categories: values with a well defined meaning independent from code and these ones that meaning is connected with the given code. All values are available by the function set of the type Get/Set. The identifier of the taken or set quantity is their parameter. Code-dependent parameters have identifiers beginning from USER\_VALUE upwards.

### I.4.1 IRBestParamSet Members

The following tables list the members exposed by IRBestParamSet.

#### Public Methods

	Name	Description
≡	Clear (see page 1664)	The function deletes all settings of parameters carried out by means of the group of functions Set*..
≡	ClearDouble (see page 1664)	The function deletes the setting of the real-number type parameter with the given identifier.
≡	ClearInteger (see page 1665)	The function deletes the setting of the integer type parameter with the given identifier.
≡	ClearString (see page 1665)	The function deletes the setting of the text type parameter with the given identifier.
≡	GetDouble (see page 1665)	The function takes a real number, which is a parameter value with a given identifier. The function returns zero (False) if the parameter value with the indicated identifier was not set.
≡	GetInteger (see page 1665)	The function takes an integer number, which is a parameter value with a given identifier. It gives back zero (False) if the value of the parameter with the indicated identifier was not set.
≡	GetString (see page 1666)	The function takes a text (character string) which is a parameter value with the given identifier. It gives zero (False) back if the value of the parameter with the indicated identifier was not set. .
≡	IsValidDouble (see page 1666)	The function checks if the indicated real number is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
≡	IsValidInteger (see page 1666)	The function checks if the indicated integer is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
≡	IsValidString (see page 1666)	The function checks if the indicated text (character string) is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
≡	SetDouble (see page 1667)	The function sets a value of the given parameter if it is correct (according to IsValidDouble (see page 1666)). If the parameter value is incorrect, the function returns zero (False).
≡	SetInteger (see page 1667)	The function sets a value of the given parameter if it is correct (according to IsValidInteger (see page 1666)). If the parameter value is incorrect, the function returns zero (False).
≡	SetString (see page 1667)	The function sets the value of the given parameter, if correct (according to IsValidString (see page 1666)). If the parameter value is not correct, the function returns zero (False). .

### I.4.2 IRBestParamSet Methods

The methods of the IRBestParamSet class are listed here.

#### Public Methods

	Name	Description
≡	Clear (see page 1664)	The function deletes all settings of parameters carried out by means of the group of functions Set*..

	ClearDouble ( <a href="#">see page 1664</a> )	The function deletes the setting of the real-number type parameter with the given identifier.
	ClearInteger ( <a href="#">see page 1665</a> )	The function deletes the setting of the integer type parameter with the given identifier.
	ClearString ( <a href="#">see page 1665</a> )	The function deletes the setting of the text type parameter with the given identifier.
	GetDouble ( <a href="#">see page 1665</a> )	The function takes a real number, which is a parameter value with a given identifier. The function returns zero (False) if the parameter value with the indicated identifier was not set.
	GetInteger ( <a href="#">see page 1665</a> )	The function takes an integer number, which is a parameter value with a given identifier. It gives back zero (False) if the value of the parameter with the indicated identifier was not set.
	GetString ( <a href="#">see page 1666</a> )	The function takes a text (character string) which is a parameter value with the given identifier. It gives zero (False) back if the value of the parameter with the indicated identifier was not set. .
	IsValidDouble ( <a href="#">see page 1666</a> )	The function checks if the indicated real number is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	IsValidInteger ( <a href="#">see page 1666</a> )	The function checks if the indicated integer is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	IsValidString ( <a href="#">see page 1666</a> )	The function checks if the indicated text (character string) is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	SetDouble ( <a href="#">see page 1667</a> )	The function sets a value of the given parameter if it is correct (according to IsValidDouble ( <a href="#">see page 1666</a> )). If the parameter value is incorrect, the function returns zero (False).
	SetInteger ( <a href="#">see page 1667</a> )	The function sets a value of the given parameter if it is correct (according to IsValidInteger ( <a href="#">see page 1666</a> )). If the parameter value is incorrect, the function returns zero (False).
	SetString ( <a href="#">see page 1667</a> )	The function sets the value of the given parameter, if correct (according to IsValidString ( <a href="#">see page 1666</a> )). If the parameter value is not correct, the function returns zero (False). .

#### I.4.2.1 Clear

C++

```
HRESULT Clear();
```

C#

```
public void Clear();
```

Visual Basic

```
Public Sub Clear()
```

Description

The function deletes all settings of parameters carried out by means of the group of functions Set\*. .

#### I.4.2.2 ClearDouble

C++

```
HRESULT ClearDouble(int _param_id);
```

C#

```
public void ClearDouble(int _param_id);
```

**Visual Basic**

```
Public Sub ClearDouble(_param_id As int)
```

**Description**

The function deletes the setting of the real-number type parameter with the given identifier.

**I.4.2.3 ClearInteger****C++**

```
HRESULT ClearInteger(int _param_id);
```

**C#**

```
public void ClearInteger(int _param_id);
```

**Visual Basic**

```
Public Sub ClearInteger(_param_id As int)
```

**Description**

The function deletes the setting of the integer type parameter with the given identifier.

**I.4.2.4 ClearString****C++**

```
HRESULT ClearString(int _param_id);
```

**C#**

```
public void ClearString(int _param_id);
```

**Visual Basic**

```
Public Sub ClearString(_param_id As int)
```

**Description**

The function deletes the setting of the text type parameter with the given identifier.

**I.4.2.5 GetDouble****C++**

```
HRESULT GetDouble(int _param_id, double* _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetDouble(int _param_id, double* _value);
```

**Visual Basic**

```
Public Function GetDouble(_param_id As int, ByRef _value As double*) As Boolean
```

**Description**

The function takes a real number, which is a parameter value with a given identifier. The function returns zero (False) if the parameter value with the indicated identifier was not set.

**I.4.2.6 GetInteger****C++**

```
HRESULT GetInteger(int _param_id, int* _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetInteger(int _param_id, int* _value);
```

**Visual Basic**

```
Public Function GetInteger(_param_id As int, ByRef _value As int*) As Boolean
```

**Description**

The function takes an integer number, which is a parameter value with a given identifier. It gives back zero (False) if the value of the parameter with the indicated identifier was not set.

**I.4.2.7 GetString****C++**

```
HRESULT GetString(int _param_id, BSTR* _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetString(int _param_id, BSTR* _value);
```

**Visual Basic**

```
Public Function GetString(_param_id As int, ByRef _value As BSTR*) As Boolean
```

**Description**

The function takes a text (character string) which is a parameter value with the given identifier. It gives zero (False) back if the value of the parameter with the indicated identifier was not set. .

**I.4.2.8 IsValidDouble****C++**

```
HRESULT IsValidDouble(int _param_id, double _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsValidDouble(int _param_id, double _value);
```

**Visual Basic**

```
Public Function IsValidDouble(_param_id As int, _value As double) As Boolean
```

**Description**

The function checks if the indicated real number is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .

**I.4.2.9 IsValidInteger****C++**

```
HRESULT IsValidInteger(int _param_id, int _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsValidInteger(int _param_id, int _value);
```

**Visual Basic**

```
Public Function IsValidInteger(_param_id As int, _value As int) As Boolean
```

**Description**

The function checks if the indicated integer is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .

#### I.4.2.10 IsValidString

**C++**

```
HRESULT IsValidString(int _param_id, BSTR _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsValidString(int _param_id, String _value);
```

**Visual Basic**

```
Public Function IsValidString(_param_id As int, _value As String) As Boolean
```

**Description**

The function checks if the indicated text (character string) is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .

#### I.4.2.11 SetDouble

**C++**

```
HRESULT SetDouble(int _param_id, double _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetDouble(int _param_id, double _value);
```

**Visual Basic**

```
Public Function SetDouble(_param_id As int, _value As double) As Boolean
```

**Description**

The function sets a value of the given parameter if it is correct (according to IsValidDouble ( see page 1666)). If the parameter value is incorrect, the function returns zero (False).

#### I.4.2.12 SetInteger

**C++**

```
HRESULT SetInteger(int _param_id, int _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetInteger(int _param_id, int _value);
```

**Visual Basic**

```
Public Function SetInteger(_param_id As int, _value As int) As Boolean
```

**Description**

The function sets a value of the given parameter if it is correct (according to IsValidInteger ( see page 1666)). If the parameter value is incorrect, the function returns zero (False).

#### I.4.2.13 SetString

**C++**

```
HRESULT SetString(int _param_id, BSTR _value, VARIANT_BOOL* ret);
```

**C#**

```
public bool SetString(int _param_id, BSTR _value);
```

## Visual Basic

```
Public Function SetString(_param_id As int, _value As BSTR) As Boolean
```

### Description

The function sets the value of the given parameter, if correct (according to IsValidString (see page 1666)). If the parameter value is not correct, the function returns zero (False). .

## I.5 IRBestCalcParamsData

### Class Hierarchy

#### C++

```
interface IRBestCalcParamsData : IRBestParamSet;
```

#### C#

```
public interface IRBestCalcParamsData : IRBestParamSet;
```

## Visual Basic

```
Public Interface IRBestCalcParamsData
```

### Description

The interface defines calculation parameters of theoretical reinforcement for an RC member. Available parameters are described by the following types: RBestCalcParamsDataDoubleValue, RBestCalcParamsDataIntegerValue and RBestCalcParamsDataSetStringValue.

### I.5.1 IRBestCalcParamsData Members

The following tables list the members exposed by IRBestCalcParamsData.

#### Public Fields

	Name	Description
◆	ModularityList (see page 1669)	A list of modularity values.

#### Public Methods

	Name	Description
◆	Clear (see page 1664)	The function deletes all settings of parameters carried out by means of the group of functions Set*. .
◆	ClearDouble (see page 1664)	The function deletes the setting of the real-number type parameter with the given identifier.
◆	ClearInteger (see page 1665)	The function deletes the setting of the integer type parameter with the given identifier.
◆	ClearString (see page 1665)	The function deletes the setting of the text type parameter with the given identifier.
◆	GetDouble (see page 1665)	The function takes a real number, which is a parameter value with a given identifier. The function returns zero (False) if the parameter value with the indicated identifier was not set.
◆	GetInteger (see page 1665)	The function takes an integer number, which is a parameter value with a given identifier. It gives back zero (False) if the value of the parameter with the indicated identifier was not set.
◆	GetString (see page 1666)	The function takes a text (character string) which is a parameter value with the given identifier. It gives zero (False) back if the value of the parameter with the indicated identifier was not set. .
◆	IsValidDouble (see page 1666)	The function checks if the indicated real number is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .

	IsValidInteger (see page 1666)	The function checks if the indicated integer is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	IsValidString (see page 1666)	The function checks if the indicated text (character string) is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	SetDouble (see page 1667)	The function sets a value of the given parameter if it is correct (according to IsValidDouble (see page 1666)). If the parameter value is incorrect, the function returns zero (False).
	SetInteger (see page 1667)	The function sets a value of the given parameter if it is correct (according to IsValidInteger (see page 1666)). If the parameter value is incorrect, the function returns zero (False).
	SetString (see page 1667)	The function sets the value of the given parameter, if correct (according to IsValidString (see page 1666)). If the parameter value is not correct, the function returns zero (False). .

## I.5.2 IRBestCalcParamsData Fields

The fields of the IRBestCalcParamsData class are listed here.

### Public Fields

	Name	Description
	ModularityList (see page 1669)	A list of modularity values.

### I.5.2.1 ModularityList

#### C++

```
HRESULT get_ModularityList(IRobotValuesArray**);
```

#### C#

```
public IRobotValuesArray ModularityList { get; }
```

#### Visual Basic

```
Public ReadOnly ModularityList As IRobotValuesArray
```

#### Description

A list of modularity values.

## I.6 IRBestCalcParamsDataDoubleValue

#### C++

```
enum IRBestCalcParamsDataDoubleValue;
```

#### C#

```
public enum IRBestCalcParamsDataDoubleValue;
```

#### Visual Basic

```
Public Enum IRBestCalcParamsDataDoubleValue
```

#### Members

Members	Description
I_BCPDDV_LONG_TENSION_BAR_DIAM = 0	Diameter of top reinforcing bars.
I_BCPDDV_LONG_COMPRESSION_BAR_DIAM = 1	Diameter of bottom reinforcing bars.
I_BCPDDV_TRAN_BAR_DIAM = 2	Diameter of transversal reinforcement.
I_BCPDDV_TRAN_INCLINATION = 3	A transversal reinforcement inclination angle (in radians).

I_BCPDDV_SAND_CONTENT = 4	Sand percentage [0..1] (for lightweight concrete).
I_BCPDDV_LONG_STEEL_FE = 5	Steel resistance of longitudinal reinforcement.
I_BCPDDV_TRAN_STEEL_FE = 6	Steel resistance of transversal reinforcement .
I_BCPDDV_CONCRETE_FC = 7	Concrete resistance.
I_BCPDDV_COVER_SIZE = 8	Cover (calculated to bar axis).
I_BCPDDV_CONCRETE_CREEP = 9	Concrete creep coefficient.
I_BCPDDV_REDISTRIBUTION = 10	Redistribution coefficient.
I_BCPDDV_MAX_BREAK = 11	Maximum cracking width.
I_BCPDDV_USER_VALUE = 1000	The smallest identifier reserved for the user-defined values.
I_BCPDDV_MAX_DEFLECTION = 12	Available since version 2.0.
I_BCPDDV_MAX_CRACK = 13	Available since version 2.0.
I_BCPDDV_MAIN_RNF_COORD_X = 14	Available since version 2.0.
I_BCPDDV_MAIN_RNF_COORD_Y = 15	Available since version 2.0.
I_BCPDDV_MAIN_RNF_COORD_Z = 16	Available since version 2.0.
I_BCPDDV_FLT2F = 17	Available since version 2.0.

#### Description

A set of parameter identifiers of real number type for the RBestCalcParamsData interface. The values of the set may be used as arguments defining the parameter type for the following functions of RBestCalcParamsData interface: SetDouble and GetDouble. .

## I.7 IRBestCalcParamsDataIntegerValue

#### C++

```
enum IRBestCalcParamsDataIntegerValue;
```

#### C#

```
public enum IRBestCalcParamsDataIntegerValue;
```

#### Visual Basic

```
Public Enum IRBestCalcParamsDataIntegerValue
```

#### Members

Members	Description
I_BCPDIV_SEISMICS = 0	Seismic risk.
I_BCPDIV_CRACKING = 1	An exposure type .
I_BCPDIV_TRAN_TYPE = 2	Type of transversal reinforcement (0 - stirrups, 1 - spiral).
I_BCPDIV_LEGS_NUM = 3	Number of legs in stirrups.
I_BCPDIV_CONCRETE_LIGHT = 4	Lightweight concrete used (0 - no, 1 - yes).
I_BCPDIV_TRAN_ZONES_NUM = 5	Number of transversal reinforcement sections in final results (refers to beams) .
I_BCPDIV_TRAN_ZONES_OPTIMIZATION = 6	Optimization of spacing of transversal reinforcement sections (0 - switched off, 1 - switched on).
I_BCPDIV_LONG_STEEL_BY_CLASS = 7	Steel resistance of longitudinal reinforcement defined on the basis of steel grade (0 - no, 1 - yes) .
I_BCPDIV_TRAN_STEEL_BY_CLASS = 8	Steel resistance of transversal reinforcement defined on the basis of steel grade (0 - no, 1 - yes).
I_BCPDIV_CONCRETE_BY_CLASS = 9	Concrete resistance defined on the basis of a concrete class (0 - no, 1 - yes).
I_BCPDIV_CONCRETE_BY_CONSTRUCTION = 10	Concrete resistance taken from the structure (0 - no, 1 - yes).
I_BCPDIV_LONG_BAR_DIAM_THE_SAME = 11	Equal bar diameters of a top and bottom longitudinal reinforcement (0 - no, 1 - yes).

I_BCPDIV_USER_VALUE = 1000	The smallest identifier reserved for the user-defined values .
I_BCPDIV_CALC_DEFLECTION = 12	Available since version 2.0.
I_BCPDIV_CALC_CRACK = 13	Available since version 2.0.
I_BCPDIV_ADJUST_DEFLECTION = 14	Available since version 2.0.
I_BCPDIV_FLT2F = 15	Available since version 2.0.
I_BCPDIV_MAIN_RNF_COORD_SYSTEM = 16	Available since version 2.0.

**Description**

An identifier set for integer type parameters for the IRBestCalcParamsData interface.

## I.8 IRBestCalcParamsDataStringValue

**C++**

```
enum IRBestCalcParamsDataStringValue;
```

**C#**

```
public enum IRBestCalcParamsDataStringValue;
```

**Visual Basic**

```
Public Enum IRBestCalcParamsDataStringValue
```

**Members**

Members	Description
I_BCPDSV_NAME = 0	Name of parameter set.
I_BCPDSV_LONG_REINF_CLASS = 1	Name of steel grade of longitudinal reinforcement.
I_BCPDSV_TRAN_REINF_CLASS = 2	Name of steel grade of transversal reinforcement.
I_BCPDSV_CONCRETE_CLASS = 3	Name of concrete class.
I_BCPDSV_USER_VALUE = 1000	The smallest identifier reserved for the user-defined values .

**Description**

The identifier set for text values transferred for the IRBestCalcParamsData interface.

## I.9 IRBestCalcParamsDataList

**Class Hierarchy****C++**

```
interface IRBestCalcParamsDataList : IDispatch;
```

**C#**

```
public interface IRBestCalcParamsDataList;
```

**Visual Basic**

```
Public Interface IRBestCalcParamsDataList
```

**Description**

An object list of the BestCalcParamsData type. Each list element is associated with a label name. All the names are saved in the LabNames (see page 1672) table.

### I.9.1 IRBestCalcParamsDataList Members

The following tables list the members exposed by IRBestCalcParamsDataList.

## Public Fields

	Name	Description
◆	LabNames ( [ see page 1672 )	Table with label names associated with particular sets of parameters .
◆	Selected ( [ see page 1672 )	Name of the label associated with the indicated element from the list (if the name is empty, no element from the list is indicated).

## Public Methods

	Name	Description
◆	Create ( [ see page 1673 )	The function creates and returns a new empty set of parameters.
◆	Delete ( [ see page 1673 )	The function deletes from the list an element associated with the given label .
◆	Get ( [ see page 1673 )	The function gives a list element associated with the given label .
◆	Store ( [ see page 1674 )	The function writes the defined set of parameters as the data for the label of the defined name. Definition of a new name results in the creation of a new label associated with the defined set of parameters. If an existing name is selected, the data associated with it will be overwritten..

## I.9.2 IRBestCalcParamsDataList Fields

The fields of the IRBestCalcParamsDataList class are listed here.

### Public Fields

	Name	Description
◆	LabNames ( [ see page 1672 )	Table with label names associated with particular sets of parameters .
◆	Selected ( [ see page 1672 )	Name of the label associated with the indicated element from the list (if the name is empty, no element from the list is indicated).

### I.9.2.1 LabNames

#### C++

```
HRESULT get_LabNames( IRobotNamesArray** );
```

#### C#

```
public IRobotNamesArray LabNames { get; }
```

#### Visual Basic

```
Public ReadOnly LabNames As IRobotNamesArray
```

#### Description

Table with label names associated with particular sets of parameters .

### I.9.2.2 Selected

#### C++

```
HRESULT get_Selected(BSTR* );
HRESULT put_Selected(BSTR);
```

#### C#

```
public String Selected { get; set; }
```

#### Visual Basic

```
Public Selected As String
```

#### Description

Name of the label associated with the indicated element from the list (if the name is empty, no element from the list is indicated).

### I.9.3 IRBestCalcParamsDataList Methods

The methods of the IRBestCalcParamsDataList class are listed here.

#### Public Methods

	Name	Description
⊕	Create ( [ see page 1673) )	The function creates and returns a new empty set of parameters.
⊕	Delete ( [ see page 1673)	The function deletes from the list an element associated with the given label .
⊕	Get ( [ see page 1673)	The function gives a list element associated with the given label .
⊕	Store ( [ see page 1674)	The function writes the defined set of parameters as the data for the label of the defined name. Definition of a new name results in the creation of a new label associated with the defined set of parameters. If an existing name is selected, the data associated with it will be overwritten..

#### I.9.3.1 Create

##### C++

```
HRESULT Create( IRBestCalcParamsData** ret);
```

##### C#

```
public IRBestCalcParamsData Create();
```

##### Visual Basic

```
Public Function Create() As IRBestCalcParamsData
```

#### Description

The function creates and returns a new empty set of parameters.

#### I.9.3.2 Delete

##### C++

```
HRESULT Delete(BSTR _lab_name, VARIANT_BOOL* ret);
```

##### C#

```
public bool Delete(String _lab_name);
```

##### Visual Basic

```
Public Function Delete(_lab_name As String) As Boolean
```

#### Description

The function deletes from the list an element associated with the given label .

#### I.9.3.3 Get

##### C++

```
HRESULT Get(BSTR _lab_name, IRBestCalcParamsData** ret);
```

##### C#

```
public IRBestCalcParamsData Get(String _lab_name);
```

##### Visual Basic

```
Public Function Get(_lab_name As String) As IRBestCalcParamsData
```

## Description

The function gives a list element associated with the given label .

### I.9.3.4 Store

#### C++

```
HRESULT Store(BSTR _lab_name, IRBestCalcParamsData* _params, VARIANT_BOOL* ret);
```

#### C#

```
public bool Store(String _lab_name, IRBestCalcParamsData _params);
```

#### Visual Basic

```
Public Function Store(_lab_name As String, ByRef _params As IRBestCalcParamsData) As Boolean
```

#### Description

The function writes the defined set of parameters as the data for the label of the defined name. Definition of a new name results in the creation of a new label associated with the defined set of parameters. If an existing name is selected, the data associated with it will be overwritten. .

## I.10 IRBestMemberData

#### Class Hierarchy

#### C++

```
interface IRBestMemberData : IRBestParamSet;
```

#### C#

```
public interface IRBestMemberData : IRBestParamSet;
```

#### Visual Basic

```
Public Interface IRBestMemberData
```

#### Description

The interface describes a data set for a type of RC member. Accessible parameters are described by the following types: RBestMemberDataDoubleValue, RBestMemberDataIntegerValue and RBestMemberDataStringValue. .

## I.11 IRBestMemberDataDoubleValue

#### C++

```
enum IRBestMemberDataDoubleValue;
```

#### C#

```
public enum IRBestMemberDataDoubleValue;
```

#### Visual Basic

```
Public Enum IRBestMemberDataDoubleValue
```

#### Members

Members	Description
I_BMDDV_LENGTH_Y = 0	Length in Y direction.
I_BMDDV_LENGTH_Z = 1	Length in Z direction.
I_BMDDV_BUCK_COEF_Y = 2	Buckling coefficient Y.

I_BMDDV_BUCK_COEF_Z = 3	Buckling coefficient Z.
I_BMDDV_SUPPORT_WIDTH_1 = 4	Length of the first support (calculated to axis).
I_BMDDV_SUPPORT_WIDTH_2 = 5	Length of the second support (calculated to axis).
I_BMDDV_USER_VALUE = 1000	The smallest identifier reserved for the user-defined values.

**Description**

A set of identifiers of real number parameters for the RBestMemberData interface.

**I.12 IRBestMemberDataIntegerValue****C++**

```
enum IRBestMemberDataIntegerValue;
```

**C#**

```
public enum IRBestMemberDataIntegerValue;
```

**Visual Basic**

```
Public Enum IRBestMemberDataIntegerValue
```

**Members**

Members	Description
I_BMDIV_CALC_AS_SLENDER_Y = 1	Calculations of slender columns in direction Y (0 - no, 1 - yes).
I_BMDIV_SWAY_Y = 3	Sway structure in direction Y (0 - no, 1 - yes).
I_BMDIV_SWAY_Z = 4	Sway structure in direction Z (0 - no, 1 - yes).
I_BMDIV_USER_VALUE = 1000	The smallest identifier reserved for a user-defined value.
I_BMDIV_TYPE = 0	Assumes one of the values of the calculable type IRBestMemberType ( see page 1685).
I_BMDIV_CALC_AS_SLENDER_Z = 2	Calculations of slender columns in direction Z (0 - no, 1 - yes).

**Description**

An identifier set of integer type parameters for the BestMemberData interface. .

**I.13 IRBestResults****Class Hierarchy****C++**

```
interface IRBestResults : IRBestParamSet;
```

**C#**

```
public interface IRBestResults : IRBestParamSet;
```

**Visual Basic**

```
Public Interface IRBestResults
```

**Description**

The interface describes calculation results of theoretical reinforcement for single section of RC member. Available values are described by the types: RBestResultsDoubleValue, RBestResultsIntegerValue and RBestResultsStringValue.

## I.14 IRBestResultsDoubleValue

### C++

```
enum IRBestResultsDoubleValue;
```

### C#

```
public enum IRBestResultsDoubleValue;
```

### Visual Basic

```
Public Enum IRBestResultsDoubleValue
```

### Members

Members	Description
I_BRDV_AS_TOP = 0	Top reinforcement area.
I_BRDV_AS_BOTTOM = 1	Bottom reinforcement area.
I_BRDV_AS_LEFT = 2	Left reinforcement area.
I_BRDV_AS_RIGHT = 3	Right reinforcement area.
I_BRDV_STIRR_SPACE = 9	Stirrup spacing.
I_BRDV_STIRR_SPACE_MOD = 10	Stirrup spacing with modularity taken into consideration.
I_BRDV_LAMBDA_Y = 13	Lambda coefficient for Y direction.
I_BRDV_LAMBDA_Z = 14	Lambda coefficient for Z direction.
I_BRDV_CRACK_WIDTH = 15	Cracking width.
I_BRDV_USER_VALUE = 1000	The smallest identifier reserved for user-defined values. .
I_BRDV_AS_MIN = 4	The minimum code-defined reinforcement area.
I_BRDV_AS_MAX = 5	The maximum code-defined reinforcement area.
I_BRDV_RO = 6	Reinforcement density.
I_BRDV_RO_MIN = 7	Minimum code-defined reinforcement density.
I_BRDV_RO_MAX = 8	Maximum code-defined reinforcement density.
I_BRDV_STIRR_SPACE_MIN = 11	Minimum code-defined stirrup spacing.
I_BRDV_STIRR_SPACE_MAX = 12	Maximum code-defined stirrup spacing.
I_BRDV_RIGIDITY = 16	Available since version 2.0.
I_BRDV_BAR_SPACING = 17	Available since version 2.0.
I_BRDV_AS_PROVIDED_TOP_MY = 18	Provided top reinforcement (My). Available since version 11.
I_BRDV_AS_PROVIDED_TOP_MZ = 19	Provided top reinforcement (Mz). Available since version 11.
I_BRDV_AS_PROVIDED_BOTTOM_MY = 20	Provided bottom reinforcement (My). Available since version 11.
I_BRDV_AS_PROVIDED_BOTTOM_MZ = 21	Provided bottom reinforcement (Mz). Available since version 11.
I_BRDV_CRACK_WIDTH_LT_TOP = 22	Available since version 12.5.
I_BRDV_CRACK_WIDTH_LT_BOTTOM = 23	Available since version 12.5.
I_BRDV_CRACK_WIDTH_ST_TOP = 24	Available since version 12.5.
I_BRDV_CRACK_WIDTH_ST_BOTTOM = 25	Available since version 12.5.
I_BRDV_STIFFNESS_TOP = 26	Available since version 12.5.
I_BRDV_STIFFNESS_BOTTOM = 27	Available since version 12.5.
I_BRDV_AS_MIN_TOP = 28	Available since version 12.5.
I_BRDV_AS_MIN_BOTTOM = 29	Available since version 12.5.
I_BRDV_STIRR_DENSITY = 30	Density of transverse reinforcement [m <sup>2</sup> /m]. Available since version 16.

**Description**

A set of identifiers of real-number parameters for the RBestResults interface. .

**I.15 IRBestResultsIntegerValue****C++**

```
enum IRBestResultsIntegerValue;
```

**C#**

```
public enum IRBestResultsIntegerValue;
```

**Visual Basic**

```
Public Enum IRBestResultsIntegerValue
```

**Members**

Members	Description
I_BRIV_BEND_CASE_IDX = 0	Index of the designing load case in the force vector for the calculations carried out for bending.
I_BRIV_BEND_ERROR_NUM = 3	The error number (RBestCalcErrors determines available error numbers) for calculations for bending.
I_BRIV_SHEAR_CASE_IDX = 1	Index of the designing load case in the force vector for the calculations carried out for shearing.
I_BRIV_TORSION_CASE_IDX = 2	Index of the designing load case in the force vector for the calculations carried out for torsion.
I_BRIV_USER_VALUE = 1000	The smallest identifier reserved for user-defined values .
I_BRIV_BEND_WARNING_NUM = 6	Warning bit mask for calculations carried out for bending (the bit values of the available warnings are determined by the interface RBestCalcWarnings).
I_BRIV_SHEAR_WARNING_NUM = 7	Warning bit mask for calculations carried out for shearing.
I_BRIV_TORSION_WARNING_NUM = 8	Warning bit mask for calculations carried out for torsion.
I_BRIV_SHEAR_ERROR_NUM = 4	Error number for calculations carried out for shearing.
I_BRIV_TORSION_ERROR_NUM = 5	Error number for calculations carried out for torsion.

**Description**

A set of identifiers of real-number parameters for the interface RBestResults.

**I.16 IRBestCalcErrors****C++**

```
enum IRBestCalcErrors;
```

**C#**

```
public enum IRBestCalcErrors;
```

**Visual Basic**

```
Public Enum IRBestCalcErrors
```

**Members**

Members	Description
I_BCE_NO_ERRORS = 0	Correct calculations.
I_BCE_NO_CALC = 1	No calculations.
I_BCE_SECTION_TOO_SMALL = 2	Section values are too small for a defined moment.

I_BCE_SECTION_SHEAR = 3	Section values are too small for a given shear force.
I_BCE_BCKL_COLUMN = 4	Axial force reaches a critical value .
I_BCE_USER_ERROR = -1	An error (defined meaning).

**Description**

The set of identifiers describing errors in calculations of theoretical reinforcement. If there occurred an error during calculations, the information about this fact should be written in the object of the RBestResults type, transferred as a parameter of the GetResults function of the RBestCodeCalcEngine interface. To write the information, one should use the SetInteger function that introduces integral values into an object. As an argument determining the type of integral value, one should provide the relevant identifier of error type, while as the error value, one should provide the relevant value from the set RBestCalcErrors. .

## I.17 IRBestCalcWarnings

**C++**

```
enum IRBestCalcWarnings;
```

**C#**

```
public enum IRBestCalcWarnings;
```

**Visual Basic**

```
Public Enum IRBestCalcWarnings
```

**Members**

Members	Description
I_BCW_NO_WARNINGS = 0	No warnings.
I_BCW_RO_MIN = 0x1	A reinforcement ratio too low.
I_BCW_RO_MAX = 0x2	A reinforcement ratio too high.
I_BCW_SPACE_BY_CODE = 0x4	Stirrup spacing calculated from a maximum stirrup spacing given in the code.
I_BCW_LAMBDA_MAX = 0x8	Slenderness exceeds a maximum value given in the code.
I_BCW_BREAK_MAX = 0x10	Cracking width exceeds a maximum value given in the code.
I_BCW_USER_WARNING = 0x20	Warning (defined meaning).
I_BCW_STEEL_EXC_RAR = 0x40	Admissible steel stress is exceeded (rare combination). Available since version 12.
I_BCW_CONCRETE_EXC_RAR = 0x80	Admissible concrete stress is exceeded (rare combination). Available since version 12.
I_BCW_CONCRETE_EXC_QPR = 0x100	Admissible concrete stress is exceeded (quasi-permanent combination). Available since version 12.
I_BCW_COT_THETA_CHANGED = 0x200	CotTHETA value is changed. Available since version 12.

**Description**

Warning bit masks for calculations of theoretical reinforcement. .

## I.18 IRBestResultsStringValue

**C++**

```
enum IRBestResultsStringValue;
```

**C#**

```
public enum IRBestResultsStringValue;
```

**Visual Basic**

```
Public Enum IRBestResultsStringValue
```

**Members**

Members	Description
I_BRSV_BEND_ERROR_NAME = 0	A message text about a definable error in calculations for bending.
I_BRSV_BEND_WARNING_NAME = 3	A message text about a definable warning in calculations for bending.
I_BRSV_SHEAR_WARNING_NAME = 4	A message text about a definable warning in calculations for shearing.
I_BRSV_TORSION_WARNING_NAME = 5	A message text about a definable warning in calculations for torsion.
I_BRSV_SHEAR_ERROR_NAME = 1	A message text about a definable error in calculations for shearing.
I_BRSV_TORSION_ERROR_NAME = 2	A message text about a definable error in calculations for torsion.
I_BRSV_USER_VALUE = 1000	The smallest identifier reserved for a user-defined value .

**Description**

Set of identifiers of text parameters for the interface RBestResults.

## I.19 IRBestForceData

**Class Hierarchy****C++**

```
interface IRBestForceData : IRBestParamSet;
```

**C#**

```
public interface IRBestForceData : IRBestParamSet;
```

**Visual Basic**

```
Public Interface IRBestForceData
```

**Description**

A set of forces acting in a bar section. .

### I.19.1 IRBestForceData Members

The following tables list the members exposed by IRBestForceData.

**Public Fields**

	Name	Description
❖	LimitState (☞ see page 1680)	Type of analyzed limit state.
❖	Values (☞ see page 1681)	Values of forces acting in a bar section.

**Public Methods**

	Name	Description
❖	Clear (☞ see page 1664)	The function deletes all settings of parameters carried out by means of the group of functions Set*. .

	ClearDouble ( <a href="#">see page 1664</a> )	The function deletes the setting of the real-number type parameter with the given identifier.
	ClearInteger ( <a href="#">see page 1665</a> )	The function deletes the setting of the integer type parameter with the given identifier.
	ClearString ( <a href="#">see page 1665</a> )	The function deletes the setting of the text type parameter with the given identifier.
	GetDouble ( <a href="#">see page 1665</a> )	The function takes a real number, which is a parameter value with a given identifier. The function returns zero (False) if the parameter value with the indicated identifier was not set.
	GetInteger ( <a href="#">see page 1665</a> )	The function takes an integer number, which is a parameter value with a given identifier. It gives back zero (False) if the value of the parameter with the indicated identifier was not set.
	GetString ( <a href="#">see page 1666</a> )	The function takes a text (character string) which is a parameter value with the given identifier. It gives zero (False) back if the value of the parameter with the indicated identifier was not set. .
	IsValidDouble ( <a href="#">see page 1666</a> )	The function checks if the indicated real number is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	IsValidInteger ( <a href="#">see page 1666</a> )	The function checks if the indicated integer is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	IsValidString ( <a href="#">see page 1666</a> )	The function checks if the indicated text (character string) is a correct value of the parameter with the indicated identifier. It returns zero (False) if the value is not correct. .
	SetDouble ( <a href="#">see page 1667</a> )	The function sets a value of the given parameter if it is correct (according to IsValidDouble ( <a href="#">see page 1666</a> )). If the parameter value is incorrect, the function returns zero (False).
	SetInteger ( <a href="#">see page 1667</a> )	The function sets a value of the given parameter if it is correct (according to IsValidInteger ( <a href="#">see page 1666</a> )). If the parameter value is incorrect, the function returns zero (False).
	SetString ( <a href="#">see page 1667</a> )	The function sets the value of the given parameter, if correct (according to IsValidString ( <a href="#">see page 1666</a> )). If the parameter value is not correct, the function returns zero (False). .

## I.19.2 IRBestForceData Fields

The fields of the IRBestForceData class are listed here.

### Public Fields

	Name	Description
	LimitState ( <a href="#">see page 1680</a> )	Type of analyzed limit state.
	Values ( <a href="#">see page 1681</a> )	Values of forces acting in a bar section.

### I.19.2.1 LimitState

#### C++

```
HRESULT get_LimitState(IRobotLimitState* );
```

#### C#

```
public IRobotLimitState LimitState { get; }
```

#### Visual Basic

```
Public ReadOnly LimitState As IRobotLimitState
```

#### Description

Type of analyzed limit state.

### I.19.2.2 Values

**C++**

```
HRESULT get_Values(IRobotBarForceData**);
```

**C#**

```
public IRobotBarForceData Values { get; }
```

**Visual Basic**

```
Public ReadOnly Values As IRobotBarForceData
```

**Description**

Values of forces acting in a bar section.

## I.20 IRBestDimParams

**Class Hierarchy**

**C++**

```
interface IRBestDimParams : IRBestParamSet;
```

**C#**

```
public interface IRBestDimParams : IRBestParamSet;
```

**Visual Basic**

```
Public Interface IRBestDimParams
```

**Description**

A set of additional values parametrizing code (design) calculations. The RBESTDIMPARAMSINTEGERVALUE type describes the set of available parameters..

## I.21 IRBestDimParamsIntegerValue

**C++**

```
enum IRBestDimParamsIntegerValue;
```

**C#**

```
public enum IRBestDimParamsIntegerValue;
```

**Visual Basic**

```
Public Enum IRBestDimParamsIntegerValue
```

**Members**

Members	Description
I_BDPIV_BEND_SIMPLE = 0	Calculations for simple bending (1 - on, 0 - off).
I_BDPIV_BEND_UNIAXIAL = 1	Calculations for offset compression and bending (1 - on, 0 - off).
I_BDPIV_BEND_BIAXIAL = 2	Calculations for offset biaxial compression (1 - on, 0 - off).
I_BDPIV_SHEAR = 3	Calculations for shearing (1 - on, 0 - off).
I_BDPIV_TORSION = 4	Calculations for torsion (1 - on, 0 - off).
I_BDPIV_USER_VALUE = 1000	The smallest identifier reserved for the user-defined values .
I_BDPIV_CALCULATION_TYPE = 7	Available since version 2.0.
I_BDPIV_DIR_Y = 5	Available since version 2.0.

I_BDPIV_DIR_Z = 6	Available since version 2.0.
I_BDPIV_DIRECTION = 8	Available since version 2.0.
I_BDPIV_BEND = 9	Available since version 2.0.
I_BDPIV_LEVEL = 10	Available since version 2.0.

## Description

Available identifiers of parameters of an integer type for the RBestDimParams interface.

## I.22 IRBestCodeCalcEngine

### Class Hierarchy

#### C++

```
interface IRBestCodeCalcEngine : IDispatch;
```

#### C#

```
public interface IRBestCodeCalcEngine;
```

#### Visual Basic

```
Public Interface IRBestCodeCalcEngine
```

### Description

The interface describes the object performing code calculations of theoretical reinforcement for RC members.

### I.22.1 IRBestCodeCalcEngine Members

The following tables list the members exposed by IRBestCodeCalcEngine.

#### Public Methods

	Name	Description
💡	Calculate (see page 1683)	The function performs calculations on a base of set parameters.
💡	GetResults (see page 1683)	The function returns the results of recently performed calculations to the indicated object.
💡	SetForces (see page 1684)	The function parametrizes the calculation process in the case of short beams and columns. It provides information about forces acting in a section (forces), as well as the position of the section along the member (pos [0 .. member length]).
💡	SetForcesArray (see page 1684)	
💡	SetForcesSlender (see page 1684)	The function parametrizes the calculation process in the case of slender columns. It provides information about forces acting in column cross-sections: at column ends (forces_begin, forces_end) and at the intermediate cross-section (forces_middle).
💡	SetGeometry (see page 1684)	The function parametrizes the calculation process. It provides information about the geometry of a bar section.
💡	SetParams (see page 1685)	The function parametrizes the calculation process. It transfers information concerning parameters for calculation of theoretical reinforcement (calc_params) as well as RC member parameters (memb_def).

### I.22.2 IRBestCodeCalcEngine Methods

The methods of the IRBestCodeCalcEngine class are listed here.

#### Public Methods

	Name	Description
💡	Calculate (see page 1683)	The function performs calculations on a base of set parameters.

	GetResults (see page 1683)	The function returns the results of recently performed calculations to the indicated object.
	SetForces (see page 1684)	The function parametrizes the calculation process in the case of short beams and columns. It provides information about forces acting in a section (forces), as well as the position of the section along the member (pos [0 .. member length]).
	SetForcesArray (see page 1684)	
	SetForcesSlender (see page 1684)	The function parametrizes the calculation process in the case of slender columns. It provides information about forces acting in column cross-sections: at column ends (forces_begin, forces_end) and at the intermediate cross-section (forces_middle).
	SetGeometry (see page 1684)	The function parametrizes the calculation process. It provides information about the geometry of a bar section.
	SetParams (see page 1685)	The function parametrizes the calculation process. It transfers information concerning parameters for calculation of theoretical reinforcement (calc_params) as well as RC member parameters (memb_def).

### I.22.2.1 Calculate

C++

```
HRESULT Calculate(IRBestDimParams* _dim_params);
```

C#

```
public void Calculate(IRBestDimParams _dim_params);
```

Visual Basic

```
Public Sub Calculate(ByRef _dim_params As IRBestDimParams)
```

Description

The function performs calculations on a base of set parameters.

### I.22.2.2 GetResults

C++

```
HRESULT GetResults(IRBestResults* _results);
```

C#

```
public void GetResults(IRBestResults _results);
```

Visual Basic

```
Public Sub GetResults(ByRef _results As IRBestResults)
```

Description

The function returns the results of recently performed calculations to the indicated object.

### I.22.2.3 SetForces

C++

```
HRESULT SetForces(IRobotObjectsArray* _forces, double _pos);
```

C#

```
public void SetForces(IRobotObjectsArray _forces, double _pos);
```

Visual Basic

```
Public Sub SetForces(ByRef _forces As IRobotObjectsArray, _pos As double)
```

## Description

The function parametrizes the calculation process in the case of short beams and columns. It provides information about forces acting in a section (forces), as well as the position of the section along the member (pos [0 .. member length]).

### I.22.2.4 SetForcesArray

#### C++

```
HRESULT SetForcesArray(SAFEARRAY* _forces, SAFEARRAY* _params, double _pos);
```

#### C#

```
public void SetForcesArray(Array _forces, Array _params, double _pos);
```

#### Visual Basic

```
Public Sub SetForcesArray(ByRef _forces As Double)(), ByRef _params As Integer)(), _pos As Double)
```

#### Version

Available since version 12.

### I.22.2.5 SetForcesSlender

#### C++

```
HRESULT SetForcesSlender(IRobotObjectsArray* _forces_begin, IRobotObjectsArray* _forces_midle, IRobotObjectsArray* _forces_end);
```

#### C#

```
public void SetForcesSlender(IRobotObjectsArray _forces_begin, IRobotObjectsArray _forces_midle, IRobotObjectsArray _forces_end);
```

#### Visual Basic

```
Public Sub SetForcesSlender(ByRef _forces_begin As IRobotObjectsArray, ByRef _forces_midle As IRobotObjectsArray, ByRef _forces_end As IRobotObjectsArray)
```

#### Description

The function parametrizes the calculation process in the case of slender columns. It provides information about forces acting in column cross-sections: at column ends (forces\_begin, forces\_end) and at the intermediate cross-section (forces\_middle).

### I.22.2.6 SetGeometry

#### C++

```
HRESULT SetGeometry(IRConcrBarSectionData* _sect_geo);
```

#### C#

```
public void SetGeometry(IRConcrBarSectionData _sect_geo);
```

#### Visual Basic

```
Public Sub SetGeometry(ByRef _sect_geo As IRConcrBarSectionData)
```

#### Description

The function parametrizes the calculation process. It provides information about the geometry of a bar section.

### I.22.2.7 SetParams

#### C++

```
HRESULT SetParams(IRBestCalcParamsData* _calc_params, IRBestMemberData* _memb_def);
```

**C#**

```
public void SetParams(IRBestCalcParamsData _calc_params, IRBestMemberData _memb_def);
```

**Visual Basic**

```
Public Sub SetParams(ByRef _calc_params As IRBestCalcParamsData, ByRef _memb_def As IRBestMemberData)
```

**Description**

The function parametrizes the calculation process. It transfers information concerning parameters for calculation of theoretical reinforcement (`calc_params`) as well as RC member parameters (`memb_def`).

## I.23 IRBestMemberDataStringValue

**C++**

```
enum IRBestMemberDataStringValue;
```

**C#**

```
public enum IRBestMemberDataStringValue;
```

**Visual Basic**

```
Public Enum IRBestMemberDataStringValue
```

**Members**

Members	Description
I_BMDSV_USER_VALUE = 1000	The smallest identifier reserved for the user .

**Description**

The set of identifiers of text parameters for the RBestMemberData interface. .

## I.24 IRBestMemberType

**C++**

```
enum IRBestMemberType;
```

**C#**

```
public enum IRBestMemberType;
```

**Visual Basic**

```
Public Enum IRBestMemberType
```

**Description**

A set of identifiers for a type of RC member or plate.

## I.25 IRBestPlateCalcParamsDlg

**Class Hierarchy****C++**

```
interface IRBestPlateCalcParamsDlg : IDispatch;
```

**C#**

```
public interface IRBestPlateCalcParamsDlg;
```

**Visual Basic**

```
Public Interface IRBestPlateCalcParamsDlg
```

**I.25.1 IRBestPlateCalcParamsDlg Members**

The following tables list the members exposed by IRBestPlateCalcParamsDlg.

**Public Methods**

	Name	Description
➊	DoModal ( see page 1686)	Available since version 1.7.
➋	SetStandard ( see page 1687)	Available since version 1.7.

**I.25.2 IRBestPlateCalcParamsDlg Methods**

The methods of the IRBestPlateCalcParamsDlg class are listed here.

**Public Methods**

	Name	Description
➊	DoModal ( see page 1686)	Available since version 1.7.
➋	SetStandard ( see page 1687)	Available since version 1.7.

**I.25.2.1 DoModal****C++**

```
HRESULT DoModal(IRBestCalcParamsData* __params, BSTR __label_name);
```

**C#**

```
public void DoModal(IRBestCalcParamsData __params, String __label_name);
```

**Visual Basic**

```
Public Sub DoModal(ByRef __params As IRBestCalcParamsData, __label_name As String)
```

**Description**

Available since version 1.7.

**I.25.2.2 SetStandard****C++**

```
HRESULT SetStandard(IRBestCalcParamsData* __params);
```

**C#**

```
public void SetStandard(IRBestCalcParamsData __params);
```

**Visual Basic**

```
Public Sub SetStandard(ByRef __params As IRBestCalcParamsData)
```

**Description**

Available since version 1.7.

**I.26 IRBestCodeCalculationType****C++**

```
enum IRBestCodeCalculationType;
```

**C#**

```
public enum IRBestCodeCalculationType;
```

**Visual Basic**

```
Public Enum IRBestCodeCalculationType
```

**Members**

Members	Description
I_BCCT_PLATE = 0	Available since version 1.7.
I_BCCT_MEMBER = 1	Available since version 1.7.

## I.27 IRBestCodeServiceExt

**Class Hierarchy****C++**

```
interface IRBestCodeServiceExt : IRBestCodeService;
```

**C#**

```
public interface IRBestCodeServiceExt : IRBestCodeService;
```

**Visual Basic**

```
Public Interface IRBestCodeServiceExt
```

**Description**

Available since version 2.0.

### I.27.1 IRBestCodeServiceExt Members

The following tables list the members exposed by IRBestCodeServiceExt.

**Public Fields**

	Name	Description
◆	CalcEngine (see page 1659)	The object performing code calculations.
◆	CalcParamsDlg (see page 1659)	A dialog box that allows to edit calculation parameters.
◆	MemberDlg (see page 1659)	A dialog box that allows to edit RC member parameters .

**Public Methods**

	Name	Description
◆	GetCalcParamsDlg (see page 1688)	Function returns the interface to the appropriate dialog box for calculation parameter definition. Available since version 2.0.
◆	IsServed (see page 1688)	Available since version 2.0.

### I.27.2 IRBestCodeServiceExt Methods

The methods of the IRBestCodeServiceExt class are listed here.

**Public Methods**

	Name	Description
◆	GetCalcParamsDlg (see page 1688)	Function returns the interface to the appropriate dialog box for calculation parameter definition. Available since version 2.0.
◆	IsServed (see page 1688)	Available since version 2.0.

### I.27.2.1 GetCalcParamsDlg

**C++**

```
HRESULT GetCalcParamsDlg(IRBestCodeCalculationType _calc_type, IUnknown* ret);
```

**C#**

```
public IUnknown GetCalcParamsDlg(IRBestCodeCalculationType _calc_type);
```

**Visual Basic**

```
Public Function GetCalcParamsDlg(_calc_type As IRBestCodeCalculationType) As IUnknown
```

**Description**

Function returns the interface to the appropriate dialog box for calculation parameter definition. Available since version 2.0.

### I.27.2.2 IsServed

**C++**

```
HRESULT IsServed(IRBestCodeCalculationType _calc_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsServed(IRBestCodeCalculationType _calc_type);
```

**Visual Basic**

```
Public Function IsServed(_calc_type As IRBestCodeCalculationType) As Boolean
```

**Description**

Available since version 2.0.

## I.28 IRBestCalculationType

**C++**

```
enum IRBestCalculationType;
```

**C#**

```
public enum IRBestCalculationType;
```

**Visual Basic**

```
Public Enum IRBestCalculationType
```

## I.29 IRBestDimParamsDoubleValue

**C++**

```
enum IRBestDimParamsDoubleValue;
```

**C#**

```
public enum IRBestDimParamsDoubleValue;
```

**Visual Basic**

```
Public Enum IRBestDimParamsDoubleValue
```

**Members**

Members	Description
I_BDPDV_CONCRETE_FC = 11	Available since version 2.0.

**I.30 IRBestBendType****C++**

```
enum IRBestBendType;
```

**C#**

```
public enum IRBestBendType;
```

**Visual Basic**

```
Public Enum IRBestBendType
```

**I.31 IRBestDirection****C++**

```
enum IRBestDirection;
```

**C#**

```
public enum IRBestDirection;
```

**Visual Basic**

```
Public Enum IRBestDirection
```

**I.32 IRBestLevel****C++**

```
enum IRBestLevel;
```

**C#**

```
public enum IRBestLevel;
```

**Visual Basic**

```
Public Enum IRBestLevel
```

**Members**

Members	Description
I_BL_DOWN = 1	Available since version 2.0.

**I.33 IRBestForceDataDoubleValue****C++**

```
enum IRBestForceDataDoubleValue;
```

**C#**

```
public enum IRBestForceDataDoubleValue;
```

**Visual Basic**

```
Public Enum IRBestForceDataDoubleValue
```

**Members**

<b>Members</b>	<b>Description</b>
I_BFDDV_COEF_LONG = 0	Available since version 2.0.
I_BFDDV_COEF_LONG2TOT = 1	Available since version 12.5.

## I.34 IRBestCoordSystem

**C++**

```
enum IRBestCoordSystem;
```

**C#**

```
public enum IRBestCoordSystem;
```

**Visual Basic**

```
Public Enum IRBestCoordSystem
```

**Members**

<b>Members</b>	<b>Description</b>
I_BCS_CARTESIAN = 1	Available since version 2.0.
I_BCS_POLAR = 2	Available since version 2.0.

## I.35 IRBestForceDataIntegerValue

**C++**

```
enum IRBestForceDataIntegerValue;
```

**C#**

```
public enum IRBestForceDataIntegerValue;
```

**Visual Basic**

```
Public Enum IRBestForceDataIntegerValue
```

**Members**

<b>Members</b>	<b>Description</b>
I_BFDIV_SLS_COMB_TYPE = 0	Available from version 11.0. Available since version 11.

**Version**

Available since version 11.

## I.36 IRBestForceDataSLSCombType

**C++**

```
enum IRBestForceDataSLSCombType;
```

**C#**

```
public enum IRBestForceDataSLSCombType;
```

**Visual Basic**

```
Public Enum IRBestForceDataSLSCombType
```

## Members

Members	Description
I_BFDLSS_COMB_TYPE_NONE = 0	Available since version 11.
I_BFDLSS_COMB_TYPE_RAR = 1	Available since version 11.
I_BFDLSS_COMB_TYPE_FRE = 2	Available since version 11.
I_BFDLSS_COMB_TYPE_QPR = 3	Available since version 11.

## Description

Available since version 11.0.

## Version

Available since version 11.

## I.37 IRBestCodeService2

### Class Hierarchy

#### C++

```
interface IRBestCodeService2 : IDispatch;
```

#### C#

```
public interface IRBestCodeService2;
```

### Visual Basic

```
Public Interface IRBestCodeService2
```

## Version

Available since version 12.5.

### I.37.1 IRBestCodeService2 Members

The following tables list the members exposed by IRBestCodeService2.

#### Public Fields

	Name	Description
❖	RobotVersion ( <a href="#">see page 1692</a> )	
❖	ShowBucklingButton ( <a href="#">see page 1692</a> )	

### I.37.2 IRBestCodeService2 Fields

The fields of the IRBestCodeService2 class are listed here.

#### Public Fields

	Name	Description
❖	RobotVersion ( <a href="#">see page 1692</a> )	
❖	ShowBucklingButton ( <a href="#">see page 1692</a> )	

#### I.37.2.1 RobotVersion

##### C++

```
HRESULT get_RobotVersion( BSTR* );
```

**C#**

```
public BSTR RobotVersion { get; }
```

**Visual Basic**

```
Public ReadOnly RobotVersion As BSTR
```

**Version**

Available since version 12.5.

**I.37.2.2 ShowBucklingButton****C++**

```
HRESULT get_ShowBucklingButton(VARIANT_BOOL* );
```

**C#**

```
public bool ShowBucklingButton { get; }
```

**Visual Basic**

```
Public ReadOnly ShowBucklingButton As Boolean
```

**Version**

Available since version 12.5.

**II Plate and shell reinforcement****Enumerations**

	<b>Name</b>	<b>Description</b>
	IRConcrReinforceDirection ( <a href="#">see page 1699</a> )	Available reinforcement directions for plates/shells.
	IRConcr_PN_SteelGrades ( <a href="#">see page 1711</a> )	Reinforcing steel grades for Polish codes: PN-84 and PN-99.
	IRConcr_PN84_ConcreteGrades ( <a href="#">see page 1712</a> )	Concrete class for Polish code PN-84.
	IRConcr_PN84_ExposureRatings ( <a href="#">see page 1712</a> )	Types of exposure rating defined for the Polish code PN-84. .
	IRConcr_PN99_ExposureRatings ( <a href="#">see page 1713</a> )	Types of exposure rating defined for the Polish code PN-99. .
	IRConcr_PN99_ConcreteGrades ( <a href="#">see page 1721</a> )	Concrete classes for Polish code PN-99.
	IRConcr_BAEL_CrackingType ( <a href="#">see page 1730</a> )	
	IRConcr_BAEL_EnvironmentType ( <a href="#">see page 1731</a> )	
	IRConcr_BAEL_WaterLevel ( <a href="#">see page 1731</a> )	
	IRConcr_BAEL_ConcreteGrades ( <a href="#">see page 1731</a> )	
	IRConcr_BAEL_SteelGrades ( <a href="#">see page 1732</a> )	
	IRConcr_PN84_HumidityType ( <a href="#">see page 1732</a> )	Types of humidity defined for the PN-84 code .
	IRConcr_ACI318_LoadActionPeriodType ( <a href="#">see page 1740</a> )	Available duration times of variable long-term loads. .
	IRConcr_ACI318_SteelGrades ( <a href="#">see page 1740</a> )	

	IRConcr_ACI318_BarDim (see page 1742)	Available bar diameters for the code ACI 318/99.
	IRConcr_ACI318_MetricBarDim (see page 1743)	Available bar diameters for the code ACI 318/99 metric.
	IRConcrReinforceBarType (see page 1743)	Types (designations) of reinforcing bars singled out during reinforcement definition. .
	IRConcr_BS8110_ExposureRatings (see page 1750)	Exposure types defined for the code BS 8110. .
	IRConcr_BS8110_ConcreteAge (see page 1750)	Predicted values of concrete age.
	IRConcr_BS8110_ConcreteGrades (see page 1751)	Concrete classes according to the code BS 8110. .
	IRConcr_BS8110_PartialSafetyFactors (see page 1751)	Safety factor types.
	IRConcrReinforceCalcType (see page 1757)	
	IRConcr_SNIP_SteelGrades (see page 1763)	
	IRConcr_SNIP_ConcreteGrades (see page 1763)	
	IRConcr_SNIP_ConcreteTypes (see page 1763)	
	IRConcr_SNIP_CuringMethods (see page 1763)	
	IRConcr_SNIP_Exposure (see page 1765)	
	IRConcr_EC2_ConcreteGrades (see page 1766)	Concrete classes according to Eurocode 2.
	IRConcr_EC2_ITALIAN_SteelGrades (see page 1766)	
	IRConcr_EC2_ExposureRatings (see page 1766)	Exposure types according to Eurocode 2.
	IRConcr_EC2_NAD (see page 1771)	
	IRConcrMinimumReinforcementType (see page 1771)	
	IRConcr_IS2000_EnvironmentType (see page 1771)	Exposure types defined for the IS 456:2000 code.

## Interfaces

	Name	Description
	IRConcrReinforceData (see page 1694)	Definition of plate and shell reinforcement type includes parameters that are code-independent.
	IRConcrConcreteParams (see page 1700)	Data characterizing concrete. .
	IRConcrSteelParams (see page 1703)	Set of parameters describing reinforcing steel properties.
	IRConcr_PN84_ReinforceData (see page 1705)	Parameters of plate and shell reinforcement for the PN-84 code. Steel grades are described by the identifier set RConcr_PN_SteeGrades, concrete classes by RConcr_PN84_ConcreteGrades, whereas types of exposure rating RConcr_PN84_ExposureRatings.
	IRConcr_PN99_ReinforceData (see page 1713)	Reinforcement parameters for the Polish code PN-99. Steel grades are described by the identifier set RConcr_PN_SteeGrades, concrete classes by RConcr_PN99_ConcreteGrades, whereas types of exposure rating by RConcr_PN99_ExposureRatings. .

	IRConcr_BAEL_ReinforceData (see page 1721)	Parameters of plate and shell reinforcement according to the BAEL code (91 and 99). Steel grades are determined by the identifiers from the set RConcr_BAEL_SteelGrades whereas concrete classes by RConcr_BAEL_ConcreteGrades. .
	IRConcr_ACI318_ReinforceData (see page 1733)	Interface of reinforcement parameters for the codes: "ACI 318/99" and "ACI 318/99 metric". .
	IRConcr_ACI318_ConcreteParams (see page 1741)	Parameters describing concrete for the code ACI 318/99. .
	IRConcr_BS8110_ReinforceData (see page 1744)	Interface of reinforcement parameters for the code BS8110.
	IRConcrSlabRequiredReinfCalcParams (see page 1752)	Calculation parameters for the required (theoretical) reinforcement for plates and shells. .
	IRConcrSlabRequiredReinfEngine (see page 1755)	Calculation module for required (theoretical) reinforcement of plates and shells. .
	IRConcrReinforceData2 (see page 1757)	
	IRConcrReinforceDataMain (see page 1758)	
	IRConcr_SNIP_ReinforceData (see page 1761)	Interface of plate/shell reinforcement parameters for the code "SNIP 2.03.01-84".
	IRConcr_SNIP_ConcreteParams (see page 1763)	
	IRConcr_EC2_ReinforceData (see page 1766)	
	IRConcr_IS2000_ReinforceData (see page 1772)	Reinforcement parameters for the Indian code IS 456:2000.
	IRConcrPlateCodeService2 (see page 1778)	

## II.1 IRConcrReinforceData

### Class Hierarchy

#### C++

```
interface IRConcrReinforceData : IDispatch;
```

#### C#

```
public interface IRConcrReinforceData;
```

#### Visual Basic

```
Public Interface IRConcrReinforceData
```

#### Description

Definition of plate and shell reinforcement type includes parameters that are code-independent.

### II.1.1 IRConcrReinforceData Members

The following tables list the members exposed by IRConcrReinforceData.

#### Public Fields

	Name	Description
	BarDim_D1_Bot (see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
	BarDim_D1_Up (see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.

◆	BarDim_D2_Bot (see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (see page 1697)	Code name Available since version 1.7.
◆	Concrete (see page 1697)	Available since version 1.7.
◆	Cover_Bot (see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	ReinforcingSteel (see page 1698)	Available since version 1.7.

#### Public Methods

	Name	Description
◆	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

### II.1.2 IRConcrReinforceData Fields

The fields of the IRConcrReinforceData class are listed here.

#### Public Fields

	Name	Description
◆	BarDim_D1_Bot (see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (see page 1697)	Code name Available since version 1.7.
◆	Concrete (see page 1697)	Available since version 1.7.
◆	Cover_Bot (see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	ReinforcingSteel (see page 1698)	Available since version 1.7.

#### II.1.2.1 BarDim\_D1\_Bot

##### C++

```
HRESULT get_BarDim_D1_Bot(double* );
HRESULT put_BarDim_D1_Bot(double);
```

##### C#

```
public double BarDim_D1_Bot { get; set; }
```

##### Visual Basic

```
Public BarDim_D1_Bot As Double
```

**Description**

Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.

**II.1.2.2 BarDim\_D1\_Up****C++**

```
HRESULT get_BarDown_D1_Up(double*);  
HRESULT put_BarDown_D1_Up(double);
```

**C#**

```
public double BarDim_D1_Up { get; set; }
```

**Visual Basic**

```
Public BarDim_D1_Up As double
```

**Description**

Diameter of reinforcing bars of top main reinforcement Available since version 1.7.

**II.1.2.3 BarDim\_D2\_Bot****C++**

```
HRESULT get_BarDown_D2_Bot(double*);  
HRESULT put_BarDown_D2_Bot(double);
```

**C#**

```
public double BarDim_D2_Bot { get; set; }
```

**Visual Basic**

```
Public BarDim_D2_Bot As double
```

**Description**

Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.

**II.1.2.4 BarDim\_D2\_Up****C++**

```
HRESULT get_BarDown_D2_Up(double*);  
HRESULT put_BarDown_D2_Up(double);
```

**C#**

```
public double BarDim_D2_Up { get; set; }
```

**Visual Basic**

```
Public BarDim_D2_Up As double
```

**Description**

Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.

**II.1.2.5 CodeName****C++**

```
HRESULT get_CodeName(BSTR*);
```

**C#**

```
public String CodeName { get; }
```

**Visual Basic**

```
Public ReadOnly CodeName As String
```

**Description**

Code name Available since version 1.7.

### II.1.2.6 Concrete

**C++**

```
HRESULT get_Concrete(IRConcrConcreteParams**);
```

**C#**

```
public IRConcrConcreteParams Concrete { get; }
```

**Visual Basic**

```
Public ReadOnly Concrete As IRConcrConcreteParams
```

**Description**

Available since version 1.7.

### II.1.2.7 Cover\_Bot

**C++**

```
HRESULT get_Cover_Bot(double*);  
HRESULT put_Cover_Bot(double);
```

**C#**

```
public double Cover_Bot { get; set; }
```

**Visual Basic**

```
Public Cover_Bot As Double
```

**Description**

Cover - bottom reinforcement Available since version 1.7.

### II.1.2.8 Cover\_Up

**C++**

```
HRESULT get_Cover_Up(double*);  
HRESULT put_Cover_Up(double);
```

**C#**

```
public double Cover_Up { get; set; }
```

**Visual Basic**

```
Public Cover_Up As Double
```

**Description**

Cover - top reinforcement Available since version 1.7.

### II.1.2.9 ReinforcingSteel

**C++**

```
HRESULT get_ReinforcingSteel(IRConcrSteelParams**);
```

**C#**

```
public IRConcrSteelParams ReinforcingSteel { get; }
```

**Visual Basic**

```
Public ReadOnly ReinforcingSteel As IRConcrSteelParams
```

**Description**

Available since version 1.7.

**II.1.3 IRConcrReinforceData Methods**

The methods of the IRConcrReinforceData class are listed here.

**Public Methods**

	Name	Description
💡	GetMainDirection (🔗 see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
💡	SetMainDirection (🔗 see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

**II.1.3.1 GetMainDirection****C++**

```
HRESULT GetMainDirection(double* _x, double* _y, double* _z, IRConcrReinforceDirection* ret);
```

**C#**

```
public IRConcrReinforceDirection GetMainDirection(double* _x, double* _y, double* _z);
```

**Visual Basic**

```
Public Function GetMainDirection(ByRef _x As Double*, ByRef _y As Double*, ByRef _z As Double*) As IRConcrReinforceDirection
```

**Description**

Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG\_VECTOR, or the point coordinates in the polar system - POLAR\_POINT). Available since version 1.7.

**II.1.3.2 SetMainDirection****C++**

```
HRESULT SetMainDirection(IRConcrReinforceDirection _dir_type, double _x, double _y, double _z);
```

**C#**

```
public void SetMainDirection(IRConcrReinforceDirection _dir_type, double _x, double _y, double _z);
```

**Visual Basic**

```
Public Sub SetMainDirection(_dir_type As IRConcrReinforceDirection, _x As Double, _y As
```

```
double, _z As double)
```

#### Description

Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG\_VECTOR, or the point coordinates in the polar system - POLAR\_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

## II.2 IRConcrReinforceDirection

#### C++

```
enum IRConcrReinforceDirection;
```

#### C#

```
public enum IRConcrReinforceDirection;
```

#### Visual Basic

```
Public Enum IRConcrReinforceDirection
```

#### Members

Members	Description
I_CRD_ALONG_X = 0	Available since version 1.7.
I_CRD_ALONG_Y = 1	Available since version 1.7.
I_CRD_ALONG_Z = 2	Available since version 1.7.
I_CRD_CARTESIAN_ALONG_VECTOR = 3	Direction determined in the Cartesian system by means of the specified vector Available since version 1.7.
I_CRD_POLAR_POINT = 4	Direction determined in the polar system by means of the specified point Available since version 1.7.
I_CRD_AUTOMATIC = 5	Automatic direction compatible to local X axis of the panel. Available since version 7.5.

#### Description

Available reinforcement directions for plates/shells.

## II.3 IRConcrConcreteParams

#### Class Hierarchy

#### C++

```
interface IRConcrConcreteParams : IDispatch;
```

#### C#

```
public interface IRConcrConcreteParams;
```

#### Visual Basic

```
Public Interface IRConcrConcreteParams
```

#### Description

Data characterizing concrete. .

### II.3.1 IRConcrConcreteParams Members

The following tables list the members exposed by IRConcrConcreteParams.

## Public Fields

	Name	Description
◆	Ec (see page 1701)	Elasticity modulus for concrete; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fc (see page 1701)	Characteristic compressive strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fc_calc (see page 1702)	Design compressive strength ( $F_{c\_calc} = F_c$ (see page 1701) / gammaC, where gammaC denotes safety factor); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fcj (see page 1702)	Value that is used only in the French code and denotes Fc (see page 1701) (characteristic compressive strength); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Ft (see page 1702)	Characteristic tensile strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Grade (see page 1702)	Identifier denoting a predefined concrete class (according to the code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

## II.3.2 IRConcrConcreteParams Fields

The fields of the IRConcrConcreteParams class are listed here.

## Public Fields

	Name	Description
◆	Ec (see page 1701)	Elasticity modulus for concrete; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fc (see page 1701)	Characteristic compressive strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fc_calc (see page 1702)	Design compressive strength ( $F_{c\_calc} = F_c$ (see page 1701) / gammaC, where gammaC denotes safety factor); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Fcj (see page 1702)	Value that is used only in the French code and denotes Fc (see page 1701) (characteristic compressive strength); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Ft (see page 1702)	Characteristic tensile strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
◆	Grade (see page 1702)	Identifier denoting a predefined concrete class (according to the code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

## II.3.2.1 Ec

### C++

```
HRESULT get_Ec(double* );
HRESULT put_Ec(double );
```

**C#**

```
public double Ec { get; set; }
```

**Visual Basic**

```
Public Ec As Double
```

**Description**

Elasticity modulus for concrete; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations

Available since version 1.7.

**II.3.2.2 Fc****C++**

```
HRESULT get_Fc(double* );
HRESULT put_Fc(double);
```

**C#**

```
public double Fc { get; set; }
```

**Visual Basic**

```
Public Fc As Double
```

**Description**

Characteristic compressive strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.

**II.3.2.3 Fc\_calc****C++**

```
HRESULT get_Fc_calc(double* );
HRESULT put_Fc_calc(double);
```

**C#**

```
public double Fc_calc { get; set; }
```

**Visual Basic**

```
Public Fc_calc As Double
```

**Description**

Design compressive strength ( $F_{c\_calc} = F_c$  (see page 1701) / gammaC, where gammaC denotes safety factor); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.

**II.3.2.4 Fcj****C++**

```
HRESULT get_Fcj(double* );
HRESULT put_Fcj(double);
```

**C#**

```
public double Fcj { get; set; }
```

**Visual Basic**

```
Public Fcj As Double
```

## Description

Value that is used only in the French code and denotes Fc (see page 1701) (characteristic compressive strength); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.

### II.3.2.5 Ft

#### C++

```
HRESULT get_Ft(double* );
HRESULT put_Ft(double);
```

#### C#

```
public double Ft { get; set; }
```

#### Visual Basic

```
Public Ft As Double
```

#### Description

Characteristic tensile strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.

### II.3.2.6 Grade

#### C++

```
HRESULT get_Grade(short* );
HRESULT put_Grade(short);
```

#### C#

```
public short Grade { get; set; }
```

#### Visual Basic

```
Public Grade As Short
```

#### Description

Identifier denoting a predefined concrete class (according to the code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

## II.4 IRConcrSteelParams

### Class Hierarchy

#### C++

```
interface IRConcrSteelParams : IDispatch;
```

#### C#

```
public interface IRConcrSteelParams;
```

#### Visual Basic

```
Public Interface IRConcrSteelParams
```

#### Description

Set of parameters describing reinforcing steel properties.

### II.4.1 IRConcrSteelParams Members

The following tables list the members exposed by IRConcrSteelParams.

## Public Fields

	Name	Description
◆	Ey ( <a href="#">see page 1704</a> )	Value of limit elastic strain Available since version 1.7.
◆	Ft ( <a href="#">see page 1704</a> )	Characteristic tensile strength Available since version 1.7.
◆	Ft_calc ( <a href="#">see page 1704</a> )	Calculation tensile strength ( $Ft\_calc = Ft ( \text{see page 1704} ) / \gamma_s$ , where $\gamma_s$ is a safety factor) Available since version 1.7.
◆	Fy ( <a href="#">see page 1704</a> )	Characteristic yield point Available since version 1.7.
◆	Fy_calc ( <a href="#">see page 1705</a> )	Calculation yield point ( $Fy\_calc = Fy ( \text{see page 1704} ) / \gamma_s$ , where $\gamma_s$ is a safety factor) Available since version 1.7.
◆	Grade ( <a href="#">see page 1705</a> )	Identifier denoting a predefined steel grade (according to code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

## II.4.2 IRConcrSteelParams Fields

The fields of the IRConcrSteelParams class are listed here.

## Public Fields

	Name	Description
◆	Ey ( <a href="#">see page 1704</a> )	Value of limit elastic strain Available since version 1.7.
◆	Ft ( <a href="#">see page 1704</a> )	Characteristic tensile strength Available since version 1.7.
◆	Ft_calc ( <a href="#">see page 1704</a> )	Calculation tensile strength ( $Ft\_calc = Ft ( \text{see page 1704} ) / \gamma_s$ , where $\gamma_s$ is a safety factor) Available since version 1.7.
◆	Fy ( <a href="#">see page 1704</a> )	Characteristic yield point Available since version 1.7.
◆	Fy_calc ( <a href="#">see page 1705</a> )	Calculation yield point ( $Fy\_calc = Fy ( \text{see page 1704} ) / \gamma_s$ , where $\gamma_s$ is a safety factor) Available since version 1.7.
◆	Grade ( <a href="#">see page 1705</a> )	Identifier denoting a predefined steel grade (according to code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

### II.4.2.1 Ey

#### C++

```
HRESULT get_Ey(double* );
HRESULT put_Ey(double);
```

#### C#

```
public double Ey { get; set; }
```

#### Visual Basic

```
Public Ey As Double
```

#### Description

Value of limit elastic strain Available since version 1.7.

### II.4.2.2 Ft

#### C++

```
HRESULT get_Ft(double* );
HRESULT put_Ft(double);
```

#### C#

```
public double Ft { get; set; }
```

#### Visual Basic

```
Public Ft As Double
```

**Description**

Characteristic tensile strength Available since version 1.7.

**II.4.2.3 Ft\_calc****C++**

```
HRESULT get_Ft_calc(double*);  
HRESULT put_Ft_calc(double);
```

**C#**

```
public double Ft_calc { get; set; }
```

**Visual Basic**

```
Public Ft_calc As double
```

**Description**

Calculation tensile strength ( $Ft\_calc = F_t$  (see page 1704) / gammaS, where gammaS is a safety factor) Available since version 1.7.

**II.4.2.4 Fy****C++**

```
HRESULT get_Fy(double*);  
HRESULT put_Fy(double);
```

**C#**

```
public double Fy { get; set; }
```

**Visual Basic**

```
Public Fy As double
```

**Description**

Characteristic yield point Available since version 1.7.

**II.4.2.5 Fy\_calc****C++**

```
HRESULT get_Fy_calc(double*);  
HRESULT put_Fy_calc(double);
```

**C#**

```
public double Fy_calc { get; set; }
```

**Visual Basic**

```
Public Fy_calc As double
```

**Description**

Calculation yield point(  $Fy\_calc = F_y$  (see page 1704) / gammaS, where gammaS is a safety factor) Available since version 1.7.

**II.4.2.6 Grade****C++**

```
HRESULT get_Grade(short*);  
HRESULT put_Grade(short);
```

**C#**

```
public short Grade { get; set; }
```

**Visual Basic**

```
Public Grade As short
```

**Description**

Identifier denoting a predefined steel grade (according to code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

## II.5 IRConcr\_PN84\_ReinforceData

**Class Hierarchy****C++**

```
interface IRConcr_PN84_ReinforceData : IRConcrReinforceData;
```

**C#**

```
public interface IRConcr_PN84_ReinforceData : IRConcrReinforceData;
```

**Visual Basic**

```
Public Interface IRConcr_PN84_ReinforceData
```

**Description**

Parameters of plate and shell reinforcement for the PN-84 code. Steel grades are described by the identifier set RConcr\_PN\_SteeGrades, concrete classes by RConcr\_PN84\_ConcreteGrades, whereas types of exposure rating RConcr\_PN84\_ExposureRatings.

### II.5.1 IRConcr\_PN84\_ReinforceData Members

The following tables list the members exposed by IRConcr\_PN84\_ReinforceData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CheckConstructionStages (↗ see page 1707)	Structure check in the phases of structure construction Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (↗ see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	LongActionAbove100 (↗ see page 1708)	term action of the temperature over 100 degrees Available since version 1.7.
◆	Main (↗ see page 1708)	
◆	ReinforcingSteel (↗ see page 1698)	Available since version 1.7.
◆	SingleShortLoad (↗ see page 1708)	Single-time short-term load Available since version 1.7.

◆	SLS_ConcreteAge (see page 1708)	Concrete (see page 1697) age (days) Available since version 1.7.
◆	SLS_Cracking (see page 1709)	Flag indicating if cracking width is to be calculated Available since version 1.7.
◆	SLS_CrackingReinfCorrect (see page 1709)	Flag switching on automatic correction of cracking width by increasing reinforcement area Available since version 1.7.
◆	SLS_CreepingCoef (see page 1709)	Concrete (see page 1697) creeping coefficient Available since version 1.7.
◆	SLS_Deflection (see page 1709)	Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.
◆	SLS_DeflectionReinfCorrect (see page 1710)	Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.
◆	SLS_EnvHumidity (see page 1710)	Type of environment humidity Available since version 1.7.
◆	SLS_Exposure (see page 1710)	Exposure rating Available since version 1.7.
◆	SLS_LoadsRatio (see page 1711)	term loads to total loads ratio Available since version 1.7.
◆	SLS_MaxCracking (see page 1711)	Admissible cracking width Available since version 1.7.
◆	SLS_MaxDeflection (see page 1711)	Admissible deflection value Available since version 1.7.

## Public Methods

	Name	Description
◆	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

## II.5.2 IRConcr\_PN84\_ReinforceData Fields

The fields of the IRConcr\_PN84\_ReinforceData class are listed here.

### Public Fields

	Name	Description
◆	CheckConstructionStages (see page 1707)	Structure check in the phases of structure construction Available since version 1.7.
◆	LongActionAbove100 (see page 1708)	term action of the temperature over 100 degrees Available since version 1.7.
◆	Main (see page 1708)	
◆	SingleShortLoad (see page 1708)	Single-time short-term load Available since version 1.7.
◆	SLS_ConcreteAge (see page 1708)	Concrete (see page 1697) age (days) Available since version 1.7.
◆	SLS_Cracking (see page 1709)	Flag indicating if cracking width is to be calculated Available since version 1.7.
◆	SLS_CrackingReinfCorrect (see page 1709)	Flag switching on automatic correction of cracking width by increasing reinforcement area Available since version 1.7.
◆	SLS_CreepingCoef (see page 1709)	Concrete (see page 1697) creeping coefficient Available since version 1.7.

❖	SLS_Deflection (see page 1709)	Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.
❖	SLS_DeflectionReinfCorrect (see page 1710)	Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.
❖	SLS_EnvHumidity (see page 1710)	Type of environment humidity Available since version 1.7.
❖	SLS_Exposure (see page 1710)	Exposure rating Available since version 1.7.
❖	SLS_LoadsRatio (see page 1711)	term loads to total loads ratio Available since version 1.7.
❖	SLS_MaxCracking (see page 1711)	Admissible cracking width Available since version 1.7.
❖	SLS_MaxDeflection (see page 1711)	Admissible deflection value Available since version 1.7.

### II.5.2.1 CheckConstructionStages

C++

```
HRESULT get_CheckConstructionStages(VARIANT_BOOL* );
HRESULT put_CheckConstructionStages(VARIANT_BOOL);
```

C#

```
public bool CheckConstructionStages { get; set; }
```

Visual Basic

```
Public CheckConstructionStages As Boolean
```

Description

Structure check in the phases of structure construction Available since version 1.7.

### II.5.2.2 LongActionAbove100

C++

```
HRESULT get_LongActionAbove100(VARIANT_BOOL* );
HRESULT put_LongActionAbove100(VARIANT_BOOL);
```

C#

```
public bool LongActionAbove100 { get; set; }
```

Visual Basic

```
Public LongActionAbove100 As Boolean
```

Description

term action of the temperature over 100 degrees Available since version 1.7.

### II.5.2.3 Main

C++

```
HRESULT get_Main(IRConcrReinforceDataMain** );
```

C#

```
public IRConcrReinforceDataMain Main { get; }
```

Visual Basic

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

#### II.5.2.4 SingleShortLoad

**C++**

```
HRESULT get_SingleShortLoad(VARIANT_BOOL* );
HRESULT put_SingleShortLoad(VARIANT_BOOL);
```

**C#**

```
public bool SingleShortLoad { get; set; }
```

**Visual Basic**

```
Public SSingleShortLoad As Boolean
```

**Description**

Single-time short-term load Available since version 1.7.

#### II.5.2.5 SLS\_ConcreteAge

**C++**

```
HRESULT get_SLS_ConcreteAge(short* );
HRESULT put_SLS_ConcreteAge(short);
```

**C#**

```
public short SLS_ConcreteAge { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteAge As short
```

**Description**

Concrete (█ see page 1697) age (days) Available since version 1.7.

#### II.5.2.6 SLS\_Cracking

**C++**

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**Description**

Flag indicating if cracking width is to be calculated Available since version 1.7.

#### II.5.2.7 SLS\_CrackingReinfCorrect

**C++**

```
HRESULT get_SLS_CrackingReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorrect As Boolean
```

**Description**

Flag switching on automatic correction of cracking width by increasing reinforcement area Available since version 1.7.

**II.5.2.8 SLS\_CreepingCoef****C++**

```
HRESULT get_SLS_CreepingCoef(double* );
HRESULT put_SLS_CreepingCoef(double);
```

**C#**

```
public double SLS_CreepingCoef { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoef As Double
```

**Description**

Concrete (see page 1697) creeping coefficient Available since version 1.7.

**II.5.2.9 SLS\_Deflection****C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Description**

Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.

**II.5.2.10 SLS\_DeflectionReinfCorrect****C++**

```
HRESULT get_SLS_DeflectionReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrect As Boolean
```

**Description**

Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.

**II.5.2.11 SLS\_EnvHumidity****C++**

```
HRESULT get_SLS_EnvHumidity(IRConcr_PN84_HumidityType* );
HRESULT put_SLS_EnvHumidity(IRConcr_PN84_HumidityType);
```

**C#**

```
public IRConcr_PN84_HumidityType SLS_EnvHumidity { get; set; }
```

**Visual Basic**

```
Public SLS_EnvHumidity As IRConcr_PN84_HumidityType
```

**Description**

Type of environment humidity Available since version 1.7.

**II.5.2.12 SLS\_Exposure****C++**

```
HRESULT get_SLS_Exposure(short* );
HRESULT put_SLS_Exposure(short);
```

**C#**

```
public short SLS_Exposure { get; set; }
```

**Visual Basic**

```
Public SLS_Exposure As short
```

**Description**

Exposure rating Available since version 1.7.

**II.5.2.13 SLS\_LoadsRatio****C++**

```
HRESULT get_SLS_LoadsRatio(double* );
HRESULT put_SLS_LoadsRatio(double);
```

**C#**

```
public double SLS_LoadsRatio { get; set; }
```

**Visual Basic**

```
Public SLS_LoadsRatio As double
```

**Description**

term loads to total loads ratio Available since version 1.7.

**II.5.2.14 SLS\_MaxCracking****C++**

```
HRESULT get_SLS_MaxCracking(double* );
HRESULT put_SLS_MaxCracking(double);
```

**C#**

```
public double SLS_MaxCracking { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCracking As double
```

**Description**

Admissible cracking width Available since version 1.7.

**II.5.2.15 SLS\_MaxDeflection****C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As Double
```

**Description**

Admissible deflection value Available since version 1.7.

## II.6 IRConcr\_PN\_SteelGrades

**C++**

```
enum IRConcr_PN_SteelGrades;
```

**C#**

```
public enum IRConcr_PN_SteelGrades;
```

**Visual Basic**

```
Public Enum IRConcr_PN_SteelGrades
```

**Members**

Members	Description
I_CPNSG_A_0 = 2	Available since version 1.7.
I_CPNSG_A_I = 3	Available since version 1.7.
I_CPNSG_A_II = 4	Available since version 1.7.
I_CPNSG_A_III = 5	Available since version 1.7.
I_CPNSG_A_IIIN = 6	Available since version 1.7.

**Description**

Reinforcing steel grades for Polish codes: PN-84 and PN-99.

## II.7 IRConcr\_PN84\_ConcreteGrades

**C++**

```
enum IRConcr_PN84_ConcreteGrades;
```

**C#**

```
public enum IRConcr_PN84_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_PN84_ConcreteGrades
```

**Members**

Members	Description
I_CPN84CG_ANOTHER = -1	Other concrete type Available since version 1.7.
I_CPN84CG_B10 = 0	Available since version 1.7.
I_CPN84CG_B12_5 = 1	Available since version 1.7.
I_CPN84CG_B15 = 2	Available since version 1.7.
I_CPN84CG_B17_5 = 3	Available since version 1.7.
I_CPN84CG_B20 = 4	Available since version 1.7.
I_CPN84CG_B25 = 5	Available since version 1.7.

I_CPN84CG_B30 = 6	Available since version 1.7.
I_CPN84CG_B35 = 7	Available since version 1.7.
I_CPN84CG_B40 = 8	Available since version 1.7.
I_CPN84CG_B50 = 9	Available since version 1.7.
I_CPN84CG_B55 = 10	Available since version 1.7.
I_CPN84CG_B60 = 11	Available since version 1.7.
I_CPN84CG_B65 = 12	Available since version 1.7.

**Description**

Concrete class for Polish code PN-84.

## II.8 IRConcr\_PN84\_ExposureRatings

**C++**

```
enum IRConcr_PN84_ExposureRatings;
```

**C#**

```
public enum IRConcr_PN84_ExposureRatings;
```

**Visual Basic**

```
Public Enum IRConcr_PN84_ExposureRatings
```

**Members**

Members	Description
I_CPN84ER_MILD = 0	Available since version 1.7.
I_CPN84ER_MODERATE = 1	Available since version 1.7.
I_CPN84ER_SEVERE_OR_LEAKPROOF = 2	Available since version 1.7.

**Description**

Types of exposure rating defined for the Polish code PN-84..

## II.9 IRConcr\_PN99\_ExposureRatings

**C++**

```
enum IRConcr_PN99_ExposureRatings;
```

**C#**

```
public enum IRConcr_PN99_ExposureRatings;
```

**Visual Basic**

```
Public Enum IRConcr_PN99_ExposureRatings
```

**Members**

Members	Description
I_CPN99ER_X0 = 0	Available since version 8.5.
I_CPN99ER_XC1_XC2_XC3_XC4 = 1	Available since version 8.5.
I_CPN99ER_XF1_XF3 = 2	Available since version 8.5.
I_CPN99ER_XD1_XD2_XD3 = 3	Available since version 8.5.
I_CPN99ER_XS1_XS2_XS3 = 4	Available since version 8.5.
I_CPN99ER_XF2_XF4 = 5	Available since version 8.5.
I_CPN99ER_XA1_XA2_XA3 = 6	Available since version 8.5.

## Description

Types of exposure rating defined for the Polish code PN-99.. .

## II.10 IRConcr\_PN99\_ReinforceData

### Class Hierarchy

#### C++

```
interface IRConcr_PN99_ReinforceData : IRConcrReinforceData;
```

#### C#

```
public interface IRConcr_PN99_ReinforceData : IRConcrReinforceData;
```

### Visual Basic

```
Public Interface IRConcr_PN99_ReinforceData
```

### Description

Reinforcement parameters for the Polish code PN-99. Steel grades are described by the identifier set RConcr\_PN\_SteeGrades, concrete classes by RConcr\_PN99\_ConcreteGrades, whereas types of exposure rating by RConcr\_PN99\_ExposureRatings.. .

## II.10.1 IRConcr\_PN99\_ReinforceData Members

The following tables list the members exposed by IRConcr\_PN99\_ReinforceData.

### Public Fields

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (↗ see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	IsSpecialStructure (↗ see page 1716)	Structure of special importance.
◆	Main (↗ see page 1716)	
◆	ReinforcingSteel (↗ see page 1698)	Available since version 1.7.
◆	SLS_Concrete (↗ see page 1716)	Concrete (↗ see page 1697) age (years) Available since version 1.7.
◆	SLS_ConcreteAge (↗ see page 1716)	Concrete (↗ see page 1697) age expressed in days (it is significant to which range among the ranges determined by the code the day number belongs) Available since version 1.7.
◆	SLS_Cracking (↗ see page 1717)	Flag indicating if cracking width is to be calculated Available since version 1.7.
◆	SLS_CrackingReinfCorrect (↗ see page 1717)	Flag switching on automatic correction of cracking width by increasing area of reinforcement Available since version 1.7.
◆	SLS_CreepingCoef (↗ see page 1717)	

◆	SLS_CreepingCoefValue (see page 1717)	
◆	SLS_Deflection (see page 1718)	Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.
◆	SLS_DeflectionReinfCorrect (see page 1718)	Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.
◆	SLS_EnvHumidityVal (see page 1718)	Environment humidity value (in %) Available since version 1.7.
◆	SLS_ExposureBot (see page 1718)	Environment class for the bottom area.
◆	SLS_ExposureTop (see page 1719)	Environment class for the top area.
◆	SLS_LoadRatio (see page 1719)	
◆	SLS_MaxCrackingBot (see page 1719)	Maximum cracking for the bottom area.
◆	SLS_MaxCrackingBotEnabled (see page 1720)	
◆	SLS_MaxCrackingTop (see page 1720)	Maximum cracking for the top area.
◆	SLS_MaxCrackingTopEnabled (see page 1720)	
◆	SLS_MaxDeflection (see page 1720)	Admissible deflection value Available since version 1.7.

#### Public Methods

	Name	Description
◆	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

#### II.10.2 IRConcr\_PN99\_ReinforceData Fields

The fields of the IRConcr\_PN99\_ReinforceData class are listed here.

#### Public Fields

	Name	Description
◆	IsSpecialStructure (see page 1716)	Structure of special importance.
◆	Main (see page 1716)	
◆	SLS_Concrete (see page 1716)	Concrete (see page 1697) age (years) Available since version 1.7.
◆	SLS_ConcreteAge (see page 1716)	Concrete (see page 1697) age expressed in days (it is significant to which range among the ranges determined by the code the day number belongs) Available since version 1.7.
◆	SLS_Cracking (see page 1717)	Flag indicating if cracking width is to be calculated Available since version 1.7.
◆	SLS_CrackingReinfCorrect (see page 1717)	Flag switching on automatic correction of cracking width by increasing area of reinforcement Available since version 1.7.
◆	SLS_CreepingCoef (see page 1717)	

◆	SLS_CreepingCoefValue (see page 1717)	
◆	SLS_Deflection (see page 1718)	Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.
◆	SLS_DeflectionReinfCorrect (see page 1718)	Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.
◆	SLS_EnvHumidityVal (see page 1718)	Environment humidity value (in %) Available since version 1.7.
◆	SLS_ExposureBot (see page 1718)	Environment class for the bottom area.
◆	SLS_ExposureTop (see page 1719)	Environment class for the top area.
◆	SLS_LoadRatio (see page 1719)	
◆	SLS_MaxCrackingBot (see page 1719)	Maximum cracking for the bottom area.
◆	SLS_MaxCrackingBotEnabled (see page 1720)	
◆	SLS_MaxCrackingTop (see page 1720)	Maximum cracking for the top area.
◆	SLS_MaxCrackingTopEnabled (see page 1720)	
◆	SLS_MaxDeflection (see page 1720)	Admissible deflection value Available since version 1.7.

### II.10.2.1 IsSpecialStructure

#### C++

```
HRESULT get_IsSpecialStructure(VARIANT_BOOL* );
HRESULT put_IsSpecialStructure(VARIANT_BOOL);
```

#### C#

```
public bool IsSpecialStructure { get; set; }
```

#### Visual Basic

```
Public IsSpecialStructure As Boolean
```

#### Description

Structure of special importance.

#### Version

Available since version 8.5.

### II.10.2.2 Main

#### C++

```
HRESULT get_Main(IRConcrReinforceDataMain** );
```

#### C#

```
public IRConcrReinforceDataMain Main { get; }
```

#### Visual Basic

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

### II.10.2.3 SLS\_Concrete

#### C++

```
HRESULT get_SLS_Concrete(double* );
```

```

HRESULT put_SLS_Concrete(double);

C#
public double SLS_Concrete { get; set; }
```

**Visual Basic**

```
Public SLS_Concrete As double
```

**Description**

Concrete (see page 1697) age (years) Available since version 1.7.

**II.10.2.4 SLS\_ConcreteAge****C++**

```

HRESULT get_SLS_ConcreteAge(short*);
HRESULT put_SLS_ConcreteAge(short);
```

**C#**

```
public short SLS_ConcreteAge { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteAge As short
```

**Description**

Concrete (see page 1697) age expressed in days (it is significant to which range among the ranges determined by the code the day number belongs) Available since version 1.7.

**II.10.2.5 SLS\_Cracking****C++**

```

HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**Description**

Flag indicating if cracking width is to be calculated Available since version 1.7.

**II.10.2.6 SLS\_CrackingReinfCorrect****C++**

```

HRESULT get_SLS_CrackingReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorrect As Boolean
```

**Description**

Flag switching on automatic correction of cracking width by increasing area of reinforcement Available since version 1.7.

### II.10.2.7 SLS\_CreepingCoef

**C++**

```
HRESULT get_SLS_CreepingCoef(VARIANT_BOOL* );
HRESULT put_SLS_CreepingCoef(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CreepingCoef { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoef As Boolean
```

### II.10.2.8 SLS\_CreepingCoefValue

**C++**

```
HRESULT get_SLS_CreepingCoefValue(double* );
HRESULT put_SLS_CreepingCoefValue(double);
```

**C#**

```
public double SLS_CreepingCoefValue { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoefValue As Double
```

### II.10.2.9 SLS\_Deflection

**C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Description**

Flag indicating if a deflection value for RC slab taking cracking into account is to be calculated Available since version 1.7.

### II.10.2.10 SLS\_DeflectionReinfCorrect

**C++**

```
HRESULT get_SLS_DeflectionReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrect As Boolean
```

**Description**

Flag switching on automatic correction of deflections by increasing reinforcement area Available since version 1.7.

### II.10.2.11 SLS\_EnvHumidityVal

**C++**

```
HRESULT get_SLS_EnvHumidityVal(double* );
```

```
HRESULT put_SLS_EnvHumidityVal(double);
```

**C#**

```
public double SLS_EnvHumidityVal { get; set; }
```

**Visual Basic**

```
Public SLS_EnvHumidityVal As double
```

**Description**

Environment humidity value (in %) Available since version 1.7.

## II.10.2.12 SLS\_ExposureBot

**C++**

```
HRESULT get_SLS_ExposureBot(short*);  
HRESULT put_SLS_ExposureBot(short);
```

**C#**

```
public short SLS_ExposureBot { get; set; }
```

**Visual Basic**

```
Public SLS_ExposureBot As short
```

**Description**

Environment class for the bottom area.

**Version**

Available since version 8.5.

## II.10.2.13 SLS\_ExposureTop

**C++**

```
HRESULT get_SLS_ExposureTop(short*);  
HRESULT put_SLS_ExposureTop(short);
```

**C#**

```
public short SLS_ExposureTop { get; set; }
```

**Visual Basic**

```
Public SLS_ExposureTop As short
```

**Description**

Environment class for the top area.

**Version**

Available since version 8.5.

## II.10.2.14 SLS\_LoadRatio

**C++**

```
HRESULT get_SLS_LoadRatio(double*);  
HRESULT put_SLS_LoadRatio(double);
```

**C#**

```
public double SLS_LoadRatio { get; set; }
```

**Visual Basic**

```
Public SLS_LoadRatio As double
```

### II.10.2.15 SLS\_MaxCrackingBot

#### C++

```
HRESULT get_SLS_MaxCrackingBot(double*);  
HRESULT put_SLS_MaxCrackingBot(double);
```

#### C#

```
public double SLS_MaxCrackingBot { get; set; }
```

#### Visual Basic

```
Public SLS_MaxCrackingBot As Double
```

#### Description

Maximum cracking for the bottom area.

#### Version

Available since version 8.5.

### II.10.2.16 SLS\_MaxCrackingBotEnabled

#### C++

```
HRESULT get_SLS_MaxCrackingBotEnabled(VARIANT_BOOL*);  
HRESULT put_SLS_MaxCrackingBotEnabled(VARIANT_BOOL);
```

#### C#

```
public bool SLS_MaxCrackingBotEnabled { get; set; }
```

#### Visual Basic

```
Public SLS_MaxCrackingBotEnabled As Boolean
```

#### Version

Available since version 8.5.

### II.10.2.17 SLS\_MaxCrackingTop

#### C++

```
HRESULT get_SLS_MaxCrackingTop(double*);  
HRESULT put_SLS_MaxCrackingTop(double);
```

#### C#

```
public double SLS_MaxCrackingTop { get; set; }
```

#### Visual Basic

```
Public SLS_MaxCrackingTop As Double
```

#### Description

Maximum cracking for the top area.

#### Version

Available since version 8.5.

### II.10.2.18 SLS\_MaxCrackingTopEnabled

#### C++

```
HRESULT get_SLS_MaxCrackingTopEnabled(VARIANT_BOOL*);  
HRESULT put_SLS_MaxCrackingTopEnabled(VARIANT_BOOL);
```

**C#**

```
public bool SLS_MaxCrackingTopEnabled { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCrackingTopEnabled As Boolean
```

**Version**

Available since version 8.5.

**II.10.2.19 SLS\_MaxDeflection****C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As Double
```

**Description**

Admissible deflection value Available since version 1.7.

**II.11 IRConcr\_PN99\_ConcreteGrades****C++**

```
enum IRConcr_PN99_ConcreteGrades;
```

**C#**

```
public enum IRConcr_PN99_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_PN99_ConcreteGrades
```

**Members**

Members	Description
I_CPN99CG_ANOTHER = -1	Other concrete class Available since version 1.7.
I_CPN99CG_B15 = 0	Available since version 1.7.
I_CPN99CG_B20 = 1	Available since version 1.7.
I_CPN99CG_B25 = 2	Available since version 1.7.
I_CPN99CG_B30 = 3	Available since version 1.7.
I_CPN99CG_B37 = 4	Available since version 1.7.
I_CPN99CG_B45 = 5	Available since version 1.7.
I_CPN99CG_B50 = 6	Available since version 1.7.
I_CPN99CG_B55 = 7	Available since version 1.7.
I_CPN99CG_B60 = 8	Available since version 1.7.
I_CPN99CG_B65 = 9	Available since version 1.7.
I_CPN99CG_B70 = 10	Available since version 1.7.

**Description**

Concrete classes for Polish code PN-99.

## II.12 IRConcr\_BAEL\_ReinforceData

### Class Hierarchy

#### C++

```
interface IRConcr_BAEL_ReinforceData : IRConcrReinforceData;
```

#### C#

```
public interface IRConcr_BAEL_ReinforceData : IRConcrReinforceData;
```

### Visual Basic

```
Public Interface IRConcr_BAEL_ReinforceData
```

### Description

Parameters of plate and shell reinforcement according to the BAEL code (91 and 99). Steel grades are determined by the identifiers from the set RConcr\_BAEL\_SteelGrades whereas concrete classes by RConcr\_BAEL\_ConcreteGrades. .

### II.12.1 IRConcr\_BAEL\_ReinforceData Members

The following tables list the members exposed by IRConcr\_BAEL\_ReinforceData.

#### Public Fields

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (↗ see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	CrackAlpha (↗ see page 1724)	Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.
◆	CrackAlphaBot (↗ see page 1724)	
◆	CrackAlphaTop (↗ see page 1725)	
◆	CrackEnv (↗ see page 1725)	Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.
◆	CrackEnvBot (↗ see page 1725)	
◆	CrackEnvTop (↗ see page 1725)	
◆	CrackExtraParams (↗ see page 1726)	Flag indicating if additional parameters resulting from the requirements of DTU 14-1 (for limited cracking) or Fascicule 74 (for not permissible cracking) are defined Available since version 2.0.
◆	CrackExtraParamsBot (↗ see page 1726)	
◆	CrackExtraParamsTop (↗ see page 1726)	
◆	Cracking (↗ see page 1726)	Available since version 1.7.
◆	CrackingBot (↗ see page 1727)	
◆	CrackingTop (↗ see page 1727)	

◆	CrackWaterLevel ( <a href="#">see page 1727</a> )	Additional parameter resulting from the requirements of DTU 14-1 Available since version 1.7.
◆	CrackWaterLevelBot ( <a href="#">see page 1727</a> )	
◆	CrackWaterLevelTop ( <a href="#">see page 1728</a> )	
◆	Main ( <a href="#">see page 1728</a> )	
◆	ReinforcingSteel ( <a href="#">see page 1698</a> )	Available since version 1.7.
◆	SLS_ConcreteStressesBot ( <a href="#">see page 1728</a> )	Allowable SLS concrete stresses for bottom reinforcement.
◆	SLS_ConcreteStressesTop ( <a href="#">see page 1728</a> )	Allowable SLS concrete stresses for top reinforcement.
◆	SLS_Deflection ( <a href="#">see page 1729</a> )	Available since version 1.7.
◆	SLS_DeflectionReinfCorrect ( <a href="#">see page 1729</a> )	Available since version 1.7.
◆	SLS_MaxDeflection ( <a href="#">see page 1729</a> )	Available since version 1.7.
◆	SLS_SteelStressesBot ( <a href="#">see page 1730</a> )	Allowable SLS steel stresses for bottom reinforcement.
◆	SLS_SteelStressesTop ( <a href="#">see page 1730</a> )	Allowable SLS steel stresses for top reinforcement.
◆	SteelSymbol ( <a href="#">see page 1730</a> )	Available since version 1.7.

#### Public Methods

	Name	Description
◆	GetMainDirection ( <a href="#">see page 1698</a> )	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection ( <a href="#">see page 1699</a> )	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

## II.12.2 IRConcr\_BAEL\_ReinforceData Fields

The fields of the IRConcr\_BAEL\_ReinforceData class are listed here.

#### Public Fields

	Name	Description
◆	CrackAlpha ( <a href="#">see page 1724</a> )	Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.
◆	CrackAlphaBot ( <a href="#">see page 1724</a> )	
◆	CrackAlphaTop ( <a href="#">see page 1725</a> )	
◆	CrackEnv ( <a href="#">see page 1725</a> )	Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.
◆	CrackEnvBot ( <a href="#">see page 1725</a> )	
◆	CrackEnvTop ( <a href="#">see page 1725</a> )	
◆	CrackExtraParams ( <a href="#">see page 1726</a> )	Flag indicating if additional parameters resulting from the requirements of DTU 14-1 (for limited cracking) or Fascicule 74 (for not permissible cracking) are defined Available since version 2.0.
◆	CrackExtraParamsBot ( <a href="#">see page 1726</a> )	

◆	CrackExtraParamsTop (see page 1726)	
◆	Cracking (see page 1726)	Available since version 1.7.
◆	CrackingBot (see page 1727)	
◆	CrackingTop (see page 1727)	
◆	CrackWaterLevel (see page 1727)	Additional parameter resulting from the requirements of DTU 14-1 Available since version 1.7.
◆	CrackWaterLevelBot (see page 1727)	
◆	CrackWaterLevelTop (see page 1728)	
◆	Main (see page 1728)	
◆	SLS_ConcreteStressesBot (see page 1728)	Allowable SLS concrete stresses for bottom reinforcement.
◆	SLS_ConcreteStressesTop (see page 1728)	Allowable SLS concrete stresses for top reinforcement.
◆	SLS_Deflection (see page 1729)	Available since version 1.7.
◆	SLS_DeflectionReinfCorrect (see page 1729)	Available since version 1.7.
◆	SLS_MaxDeflection (see page 1729)	Available since version 1.7.
◆	SLS_SteelStressesBot (see page 1730)	Allowable SLS steel stresses for bottom reinforcement.
◆	SLS_SteelStressesTop (see page 1730)	Allowable SLS steel stresses for top reinforcement.
◆	SteelSymbol (see page 1730)	Available since version 1.7.

### II.12.2.1 CrackAlpha

#### C++

```
HRESULT get_CrackAlpha(double* );
HRESULT put_CrackAlpha(double);
```

#### C#

```
public double CrackAlpha { get; set; }
```

#### Visual Basic

```
Public CrackAlpha As Double
```

#### Description

Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.

### II.12.2.2 CrackAlphaBot

#### C++

```
HRESULT get_CrackAlphaBot(double* );
HRESULT put_CrackAlphaBot(double);
```

#### C#

```
public double CrackAlphaBot { get; set; }
```

#### Visual Basic

```
Public CrackAlphaBot As Double
```

#### Version

Available since version 8.

### II.12.2.3 CrackAlphaTop

#### C++

```
HRESULT get_CrackAlphaTop(double* );
HRESULT put_CrackAlphaTop(double);
```

#### C#

```
public double CrackAlphaTop { get; set; }
```

#### Visual Basic

```
Public CrackAlphaTop As Double
```

#### Version

Available since version 8.

### II.12.2.4 CrackEnv

#### C++

```
HRESULT get_CrackEnv(IRConcr_BAEL_EnvironmentType* );
HRESULT put_CrackEnv(IRConcr_BAEL_EnvironmentType);
```

#### C#

```
public IRConcr_BAEL_EnvironmentType CrackEnv { get; set; }
```

#### Visual Basic

```
Public CrackEnv As IRConcr_BAEL_EnvironmentType
```

#### Description

Additional parameter resulting from the requirements of DTU 14-1 or Fascicule 74 Available since version 1.7.

### II.12.2.5 CrackEnvBot

#### C++

```
HRESULT get_CrackEnvBot(IRConcr_BAEL_EnvironmentType* );
HRESULT put_CrackEnvBot(IRConcr_BAEL_EnvironmentType);
```

#### C#

```
public IRConcr_BAEL_EnvironmentType CrackEnvBot { get; set; }
```

#### Visual Basic

```
Public CrackEnvBot As IRConcr_BAEL_EnvironmentType
```

#### Version

Available since version 8.

### II.12.2.6 CrackEnvTop

#### C++

```
HRESULT get_CrackEnvTop(IRConcr_BAEL_EnvironmentType* );
HRESULT put_CrackEnvTop(IRConcr_BAEL_EnvironmentType);
```

#### C#

```
public IRConcr_BAEL_EnvironmentType CrackEnvTop { get; set; }
```

#### Visual Basic

```
Public CrackEnvTop As IRConcr_BAEL_EnvironmentType
```

**Version**

Available since version 8.

**II.12.2.7 CrackExtraParams****C++**

```
HRESULT get_CrackExtraParams(VARIANT_BOOL* );
HRESULT put_CrackExtraParams(VARIANT_BOOL);
```

**C#**

```
public bool CrackExtraParams { get; set; }
```

**Visual Basic**

```
Public CrackExtraParams As Boolean
```

**Description**

Flag indicating if additional parameters resulting from the requirements of DTU 14-1 (for limited cracking) or Fascicule 74 (for not permissible cracking) are defined Available since version 2.0.

**II.12.2.8 CrackExtraParamsBot****C++**

```
HRESULT get_CrackExtraParamsBot(VARIANT_BOOL* );
HRESULT put_CrackExtraParamsBot(VARIANT_BOOL);
```

**C#**

```
public bool CrackExtraParamsBot { get; set; }
```

**Visual Basic**

```
Public CrackExtraParamsBot As Boolean
```

**Version**

Available since version 8.

**II.12.2.9 CrackExtraParamsTop****C++**

```
HRESULT get_CrackExtraParamsTop(VARIANT_BOOL* );
HRESULT put_CrackExtraParamsTop(VARIANT_BOOL);
```

**C#**

```
public bool CrackExtraParamsTop { get; set; }
```

**Visual Basic**

```
Public CrackExtraParamsTop As Boolean
```

**Version**

Available since version 8.

**II.12.2.10 Cracking****C++**

```
HRESULT get_Cracking(IRConcr_BAEL_CrackingType* );
HRESULT put_Cracking(IRConcr_BAEL_CrackingType);
```

**C#**

```
public IRConcr_BAEL_CrackingType Cracking { get; set; }
```

**Visual Basic**

```
Public Cracking As IRConcr_BAEL_CrackingType
```

**Description**

Available since version 1.7.

**II.12.2.11 CrackingBot****C++**

```
HRESULT get_CrackingBot(IRConcr_BAEL_CrackingType* );
HRESULT put_CrackingBot(IRConcr_BAEL_CrackingType);
```

**C#**

```
public IRConcr_BAEL_CrackingType CrackingBot { get; set; }
```

**Visual Basic**

```
Public CrackingBot As IRConcr_BAEL_CrackingType
```

**Version**

Available since version 8.

**II.12.2.12 CrackingTop****C++**

```
HRESULT get_CrackingTop(IRConcr_BAEL_CrackingType* );
HRESULT put_CrackingTop(IRConcr_BAEL_CrackingType);
```

**C#**

```
public IRConcr_BAEL_CrackingType CrackingTop { get; set; }
```

**Visual Basic**

```
Public CrackingTop As IRConcr_BAEL_CrackingType
```

**Version**

Available since version 8.

**II.12.2.13 CrackWaterLevel****C++**

```
HRESULT get_CrackWaterLevel(IRConcr_BAEL_WaterLevel* );
HRESULT put_CrackWaterLevel(IRConcr_BAEL_WaterLevel);
```

**C#**

```
public IRConcr_BAEL_WaterLevel CrackWaterLevel { get; set; }
```

**Visual Basic**

```
Public CrackWaterLevel As IRConcr_BAEL_WaterLevel
```

**Description**

Additional parameter resulting from the requirements of DTU 14-1 Available since version 1.7.

**II.12.2.14 CrackWaterLevelBot****C++**

```
HRESULT get_CrackWaterLevelBot(IRConcr_BAEL_WaterLevel* );
HRESULT put_CrackWaterLevelBot(IRConcr_BAEL_WaterLevel);
```

**C#**

```
public IRConcr_BAEL_WaterLevel CrackWaterLevelBot { get; set; }
```

**Visual Basic**

```
Public CrackWaterLevelBot As IRConcr_BAEL_WaterLevel
```

**Version**

Available since version 8.

**II.12.2.15 CrackWaterLevelTop****C++**

```
HRESULT get_CrackWaterLevelTop(IRConcr_BAEL_WaterLevel* );
HRESULT put_CrackWaterLevelTop(IRConcr_BAEL_WaterLevel* );
```

**C#**

```
public IRConcr_BAEL_WaterLevel CrackWaterLevelTop { get; set; }
```

**Visual Basic**

```
Public CrackWaterLevelTop As IRConcr_BAEL_WaterLevel
```

**Version**

Available since version 8.

**II.12.2.16 Main****C++**

```
HRESULT get_Main(IRConcrReinforceDataMain** );
```

**C#**

```
public IRConcrReinforceDataMain Main { get; }
```

**Visual Basic**

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

**II.12.2.17 SLS\_ConcreteStressesBot****C++**

```
HRESULT get_SLS_ConcreteStressesBot(double* );
HRESULT put_SLS_ConcreteStressesBot(double );
```

**C#**

```
public double SLS_ConcreteStressesBot { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteStressesBot As Double
```

**Description**

Allowable SLS concrete stresses for bottom reinforcement.

**Version**

Available since version 15.2.

**II.12.2.18 SLS\_ConcreteStressesTop****C++**

```
HRESULT get_SLS_ConcreteStressesTop(double* );
HRESULT put_SLS_ConcreteStressesTop(double );
```

**C#**

```
public double SLS_ConcreteStressesTop { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteStressesTop As Double
```

**Description**

Allowable SLS concrete stresses for top reinforcement.

**Version**

Available since version 15.2.

## II.12.2.19 SLS\_Deflection

**C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Description**

Available since version 1.7.

## II.12.2.20 SLS\_DeflectionReinfCorrect

**C++**

```
HRESULT get_SLS_DeflectionReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrect As Boolean
```

**Description**

Available since version 1.7.

## II.12.2.21 SLS\_MaxDeflection

**C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As Double
```

**Description**

Available since version 1.7.

## II.12.2.22 SLS\_SteelStressesBot

### C++

```
HRESULT get_SLS_SteelStressesBot(double*);  
HRESULT put_SLS_SteelStressesBot(double);
```

### C#

```
public double SLS_SteelStressesBot { get; set; }
```

### Visual Basic

```
Public SLS_SteelStressesBot As Double
```

### Description

Allowable SLS steel stresses for bottom reinforcement.

### Version

Available since version 15.2.

## II.12.2.23 SLS\_SteelStressesTop

### C++

```
HRESULT get_SLS_SteelStressesTop(double*);  
HRESULT put_SLS_SteelStressesTop(double);
```

### C#

```
public double SLS_SteelStressesTop { get; set; }
```

### Visual Basic

```
Public SLS_SteelStressesTop As Double
```

### Description

Allowable SLS steel stresses for top reinforcement.

### Version

Available since version 15.2.

## II.12.2.24 SteelSymbol

### C++

```
HRESULT get_SteelSymbol(BSTR*);  
HRESULT put_SteelSymbol(BSTR);
```

### C#

```
public String SteelSymbol { get; set; }
```

### Visual Basic

```
Public SteelSymbol As String
```

### Description

Available since version 1.7.

## II.13 IRConcr\_BAEL\_CrackingType

### C++

```
enum IRConcr_BAEL_CrackingType;
```

**C#**

```
public enum IRConcr_BAEL_CrackingType;
```

**Visual Basic**

```
Public Enum IRConcr_BAEL_CrackingType
```

**Members**

Members	Description
I_CBAELOT_OTHER = 3	Available since version 1.7.
I_CBAELOT_NOT_SEVERE = 0	Available since version 1.7.
I_CBAELOT_SEVERE = 1	Available since version 1.7.
I_CBAELOT_VERY_SEVERE = 2	Available since version 1.7.

## II.14 IRConcr\_BAEL\_EnvironmentType

**C++**

```
enum IRConcr_BAEL_EnvironmentType;
```

**C#**

```
public enum IRConcr_BAEL_EnvironmentType;
```

**Visual Basic**

```
Public Enum IRConcr_BAEL_EnvironmentType
```

**Members**

Members	Description
I_CBAELET.Aggressive_Water = 1	Available since version 1.7.
I_CBAELET.Normal_Water = 0	Available since version 1.7.

## II.15 IRConcr\_BAEL\_WaterLevel

**C++**

```
enum IRConcr_BAEL_WaterLevel;
```

**C#**

```
public enum IRConcr_BAEL_WaterLevel;
```

**Visual Basic**

```
Public Enum IRConcr_BAEL_WaterLevel
```

**Members**

Members	Description
I_CBAELWL.Low = 0	Available since version 1.7.
I_CBAELWL.High = 1	Available since version 1.7.
I_CBAELWL.Exceptional = 2	Available since version 1.7.

## II.16 IRConcr\_BAEL\_ConcreteGrades

**C++**

```
enum IRConcr_BAEL_ConcreteGrades;
```

**C#**

```
public enum IRConcr_BAEL_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_BAEL_ConcreteGrades
```

**Members**

Members	Description
I_CBAELCG_OTHER = -1	Available since version 1.7.
I_CBAELCG_C12_15 = 0	Available since version 1.7.
I_CBAELCG_C16_20 = 1	Available since version 1.7.
I_CBAELCG_C20_25 = 2	Available since version 1.7.
I_CBAELCG_C25_30 = 3	Available since version 1.7.
I_CBAELCG_C30_37 = 4	Available since version 1.7.
I_CBAELCG_C35_45 = 5	Available since version 1.7.
I_CBAELCG_C40_50 = 6	Available since version 1.7.
I_CBAELCG_C45_55 = 7	Available since version 1.7.
I_CBAELCG_C50_60 = 8	Available since version 1.7.

**II.17 IRConcr\_BAEL\_SteelGrades****C++**

```
enum IRConcr_BAEL_SteelGrades;
```

**C#**

```
public enum IRConcr_BAEL_SteelGrades;
```

**Visual Basic**

```
Public Enum IRConcr_BAEL_SteelGrades
```

**Members**

Members	Description
I_CBAELSG_PLAIN = 0	Available since version 1.7.
I_CBAELSG_DEFORMED = 1	Available since version 1.7.

**II.18 IRConcr\_PN84\_HumidityType****C++**

```
enum IRConcr_PN84_HumidityType;
```

**C#**

```
public enum IRConcr_PN84_HumidityType;
```

**Visual Basic**

```
Public Enum IRConcr_PN84_HumidityType
```

**Members**

Members	Description
I_CPN84HT_40 = 0	Up to 40 percent Available since version 2.0.
I_CPN84HT_40_75 = 1	From 40 to 75 percent Available since version 2.0.
I_CPN84HT_75 = 2	Over 75 percent Available since version 2.0.

I_CPN84HT_WATER = 3	Water Available since version 2.0.
---------------------	------------------------------------

**Description**

Types of humidity defined for the PN-84 code .

## II.19 IRConcr\_ACI318\_ReinforceData

**Class Hierarchy****C++**

```
interface IRConcr_ACI318_ReinforceData : IDispatch;
```

**C#**

```
public interface IRConcr_ACI318_ReinforceData;
```

**Visual Basic**

```
Public Interface IRConcr_ACI318_ReinforceData
```

**Description**

Interface of reinforcement parameters for the codes: "ACI 318/99" and "ACI 318/99 metric". .

**Version**

Available since version 3.5.

### II.19.1 IRConcr\_ACI318\_ReinforceData Members

The following tables list the members exposed by IRConcr\_ACI318\_ReinforceData.

**Public Fields**

	Name	Description
◆	IsMetric ( <a href="#">see page 1734</a> )	Flag informing about the code variant: true value for "ACI 318/99 metric", false value for "ACI 318/99". .
◆	Main ( <a href="#">see page 1735</a> )	
◆	SLS_Cracking ( <a href="#">see page 1735</a> )	Flag determining calculation of cracking width.
◆	SLS_CreepingCoef ( <a href="#">see page 1735</a> )	
◆	SLS_CreepingCoefValue ( <a href="#">see page 1735</a> )	
◆	SLS_Deflection ( <a href="#">see page 1736</a> )	Flag determining calculation of a deflection value.
◆	SLS_DeflectionReinfCorrection ( <a href="#">see page 1736</a> )	Flag managing automatic correction of deflections by increasing the area of reinforcement .
◆	SLS_Factor ( <a href="#">see page 1736</a> )	Flag indicating if the concrete creep coefficient should be considered in reinforcement definition .
◆	SLS_FactorValue ( <a href="#">see page 1737</a> )	Value of concrete creep coefficient.
◆	SLS_LoadActionPeriod ( <a href="#">see page 1737</a> )	Duration of variable long-term loads.
◆	SLS_LoadRatio ( <a href="#">see page 1737</a> )	Ratio of long-term loads to total loads.
◆	SLS_MaxCracking ( <a href="#">see page 1738</a> )	Admissible value of cracking width.
◆	SLS_MaxDeflection ( <a href="#">see page 1738</a> )	Admissible value of deflection.

## Public Methods

	Name	Description
»	GetBarDim ( [ see page 1738) )	Function takes diameter of the specified bar. .
»	GetMetricBarDim ( [ see page 1739) )	Equivalent of the GetBarDim ( [ see page 1738) function for the metric variant of the code. .
»	SetBarDim ( [ see page 1739) )	Function sets a diameter of the specified bar.
»	SetMetricBarDim ( [ see page 1739) )	Equivalent of the SetBarDim ( [ see page 1739) function for the metric variant of the code. .

## II.19.2 IRConcr\_ACI318\_ReinforceData Fields

The fields of the IRConcr\_ACI318\_ReinforceData class are listed here.

### Public Fields

	Name	Description
◆	IsMetric ( [ see page 1734) )	Flag informing about the code variant: true value for "ACI 318/99 metric", false value for "ACI 318/99". .
◆	Main ( [ see page 1735) )	
◆	SLS_Cracking ( [ see page 1735) )	Flag determining calculation of cracking width.
◆	SLS_CreepingCoef ( [ see page 1735) )	
◆	SLS_CreepingCoefValue ( [ see page 1735) )	
◆	SLS_Deflection ( [ see page 1736) )	Flag determining calculation of a deflection value.
◆	SLS_DeflectionReinfCorrection ( [ see page 1736) )	Flag managing automatic correction of deflections by increasing the area of reinforcement .
◆	SLS_Factor ( [ see page 1736) )	Flag indicating if the concrete creep coefficient should be considered in reinforcement definition .
◆	SLS_FactorValue ( [ see page 1737) )	Value of concrete creep coefficient.
◆	SLS_LoadActionPeriod ( [ see page 1737) )	Duration of variable long-term loads.
◆	SLS_LoadRatio ( [ see page 1737) )	Ratio of long-term loads to total loads.
◆	SLS_MaxCracking ( [ see page 1738) )	Admissible value of cracking width.
◆	SLS_MaxDeflection ( [ see page 1738) )	Admissible value of deflection.

### II.19.2.1 IsMetric

#### C++

```
HRESULT get_IsMetric(VARIANT_BOOL* );
```

#### C#

```
public bool IsMetric { get; }
```

#### Visual Basic

```
Public ReadOnly IsMetric As Boolean
```

#### Description

Flag informing about the code variant: true value for "ACI 318/99 metric", false value for "ACI 318/99". .

#### Version

Available since version 3.5.

## II.19.2.2 Main

### C++

```
HRESULT get_Main( IRConcrReinforceDataMain** );
```

### C#

```
public IRConcrReinforceDataMain Main { get; }
```

### Visual Basic

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

## II.19.2.3 SLS\_Cracking

### C++

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

### C#

```
public bool SLS_Cracking { get; set; }
```

### Visual Basic

```
Public SLS_Cracking As Boolean
```

### Description

Flag determining calculation of cracking width.

### Version

Available since version 3.5.

## II.19.2.4 SLS\_CreepingCoef

### C++

```
HRESULT get_SLS_CreepingCoef(VARIANT_BOOL* );
HRESULT put_SLS_CreepingCoef(VARIANT_BOOL);
```

### C#

```
public bool SLS_CreepingCoef { get; set; }
```

### Visual Basic

```
Public SLS_CreepingCoef As Boolean
```

## II.19.2.5 SLS\_CreepingCoefValue

### C++

```
HRESULT get_SLS_CreepingCoefValue(double* );
HRESULT put_SLS_CreepingCoefValue(double);
```

### C#

```
public double SLS_CreepingCoefValue { get; set; }
```

### Visual Basic

```
Public SLS_CreepingCoefValue As double
```

## II.19.2.6 SLS\_Deflection

### C++

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Description**

Flag determining calculation of a deflection value.

**Version**

Available since version 3.5.

**II.19.2.7 SLS\_DeflectionReinfCorrection****C++**

```
HRESULT get_SLS_DeflectionReinfCorrection(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrection { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrection As Boolean
```

**Description**

Flag managing automatic correction of deflections by increasing the area of reinforcement .

**Version**

Available since version 3.5.

**II.19.2.8 SLS\_Factor****C++**

```
HRESULT get_SLS_Factor(VARIANT_BOOL* );
HRESULT put_SLS_Factor(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Factor { get; set; }
```

**Visual Basic**

```
Public SLS_Factor As Boolean
```

**Description**

Flag indicating if the concrete creep coefficient should be considered in reinforcement definition .

**Version**

Available since version 3.5.

**II.19.2.9 SLS\_FactorValue****C++**

```
HRESULT get_SLS_FactorValue(double* );
HRESULT put_SLS_FactorValue(double);
```

**C#**

```
public double SLS_FactorValue { get; set; }
```

**Visual Basic**

```
Public SLS_FactorValue As double
```

**Description**

Value of concrete creep coefficient.

**Version**

Available since version 3.5.

**II.19.2.10 SLS\_LoadActionPeriod****C++**

```
HRESULT get_SLS_LoadActionPeriod(IRConcr_ACI318_LoadActionPeriodType* );
HRESULT put_SLS_LoadActionPeriod(IRConcr_ACI318_LoadActionPeriodType);
```

**C#**

```
public IRConcr_ACI318_LoadActionPeriodType SLS_LoadActionPeriod { get; set; }
```

**Visual Basic**

```
Public SLS_LoadActionPeriod As IRConcr_ACI318_LoadActionPeriodType
```

**Description**

Duration of variable long-term loads.

**Version**

Available since version 3.5.

**II.19.2.11 SLS\_LoadRatio****C++**

```
HRESULT get_SLS_LoadRatio(double* );
HRESULT put_SLS_LoadRatio(double);
```

**C#**

```
public double SLS_LoadRatio { get; set; }
```

**Visual Basic**

```
Public SLS_LoadRatio As double
```

**Description**

Ratio of long-term loads to total loads.

**Version**

Available since version 3.5.

**II.19.2.12 SLS\_MaxCracking****C++**

```
HRESULT get_SLS_MaxCracking(double* );
HRESULT put_SLS_MaxCracking(double);
```

**C#**

```
public double SLS_MaxCracking { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCracking As double
```

**Description**

Admissible value of cracking width.

**Version**

Available since version 3.5.

**II.19.2.13 SLS\_MaxDeflection****C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As Double
```

**Description**

Admissible value of deflection.

**Version**

Available since version 3.5.

**II.19.3 IRConcr\_ACI318\_ReinforceData Methods**

The methods of the IRConcr\_ACI318\_ReinforceData class are listed here.

**Public Methods**

	Name	Description
💡	GetBarDim ( <a href="#">see page 1738</a> )	Function takes diameter of the specified bar..
💡	GetMetricBarDim ( <a href="#">see page 1739</a> )	Equivalent of the GetBarDim ( <a href="#">see page 1738</a> ) function for the metric variant of the code..
💡	SetBarDim ( <a href="#">see page 1739</a> )	Function sets a diameter of the specified bar.
💡	SetMetricBarDim ( <a href="#">see page 1739</a> )	Equivalent of the SetBarDim ( <a href="#">see page 1739</a> ) function for the metric variant of the code..

**II.19.3.1 GetBarDim****C++**

```
HRESULT GetBarDim( IRConcrReinforceBarType _type, IRConcr_ACI318_BarDown* ret);
```

**C#**

```
public IRConcr_ACI318_BarDown GetBarDim(IRConcrReinforceBarType _type);
```

**Visual Basic**

```
Public Function GetBarDim(_type As IRConcrReinforceBarType) As IRConcr_ACI318_BarDown
```

**Description**

Function takes diameter of the specified bar..

**Version**

Available since version 3.5.

### II.19.3.2 GetMetricBarDim

#### C++

```
HRESULT GetMetricBarDim(IRConnrReinforceBarType _bar_type, IRConnr_ACI318_MetricBarDim*  
ret);
```

#### C#

```
public IRConnr_ACI318_MetricBarDim GetMetricBarDim(IRConnrReinforceBarType _bar_type);
```

#### Visual Basic

```
Public Function GetMetricBarDim(_bar_type As IRConnrReinforceBarType) As  
IRConnr_ACI318_MetricBarDim
```

#### Description

Equivalent of the GetBarDim (see page 1738) function for the metric variant of the code. .

#### Version

Available since version 3.5.

### II.19.3.3 SetBarDim

#### C++

```
HRESULT SetBarDim(IRConcrReinforceBarType _type, IRConcr_ACI318_Bardim _val);
```

#### C#

```
public void SetBarDim(IRConcrReinforceBarType _type, IRConcr_ACI318_Bardim _val);
```

#### Visual Basic

```
Public Sub SetBarDim(_type As IRConcrReinforceBarType, _val As IRConcr_ACI318_Bardim)
```

#### Description

Function sets a diameter of the specified bar.

#### Version

Available since version 3.5.

### II.19.3.4 SetMetricBarDim

#### C++

```
HRESULT SetMetricBarDim(IRConnrReinforceBarType _type, IRConnr_ACI318_MetricBarDim _val);
```

#### C#

```
public void SetMetricBarDim(IRConnrReinforceBarType _type, IRConnr_ACI318_MetricBarDim  
_val);
```

#### Visual Basic

```
Public Sub SetMetricBarDim(_type As IRConnrReinforceBarType, _val As  
IRConnr_ACI318_MetricBarDim)
```

#### Description

Equivalent of the SetBarDim (see page 1739) function for the metric variant of the code. .

#### Version

Available since version 3.5.

## II.20 IRConcr\_ACI318\_LoadActionPeriodType

### C++

```
enum IRConcr_ACI318_LoadActionPeriodType;
```

### C#

```
public enum IRConcr_ACI318_LoadActionPeriodType;
```

### Visual Basic

```
Public Enum IRConcr_ACI318_LoadActionPeriodType
```

### Members

Members	Description
I_ACI318_LAPT_3 = 0	Available since version 3.5.
I_ACI318_LAPT_6 = 1	Available since version 3.5.
I_ACI318_LAPT_12 = 2	Available since version 3.5.
I_ACI318_LAPT_60 = 3	Available since version 3.5.

### Description

Available duration times of variable long-term loads. .

### Version

Available since version 3.5.

## II.21 IRConcr\_ACI318\_SteelGrades

### C++

```
enum IRConcr_ACI318_SteelGrades;
```

### C#

```
public enum IRConcr_ACI318_SteelGrades;
```

### Visual Basic

```
Public Enum IRConcr_ACI318_SteelGrades
```

### Members

Members	Description
I_ACI318_ST_40 = 0	Available since version 3.5.
I_ACI318_ST_50 = 1	Available since version 3.5.
I_ACI318_ST_60 = 2	Available since version 3.5.
I_ACI318_ST_75 = 3	Available since version 3.5.

### Version

Available since version 3.5.

## II.22 IRConcr\_ACI318\_ConcreteParams

### Class Hierarchy

### C++

```
interface IRConcr_ACI318_ConcreteParams : IRConcrConcreteParams;
```

**C#**

```
public interface IRConcr_ACI318_ConcreteParams : IRConcrConcreteParams;
```

**Visual Basic**

```
Public Interface IRConcr_ACI318_ConcreteParams
```

**Description**

Parameters describing concrete for the code ACI 318/99. .

**Version**

Available since version 3.5.

**II.22.1 IRConcr\_ACI318\_ConcreteParams Members**

The following tables list the members exposed by IRConcr\_ACI318\_ConcreteParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AsInStructure ( <a href="#">see page 1742</a> )	Flag indicating if reinforcement definition considers compression resistance of the material currently added to the panel .
❖	Ec ( <a href="#">see page 1701</a> )	Elasticity modulus for concrete; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fc ( <a href="#">see page 1701</a> )	Characteristic compressive strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fc_calc ( <a href="#">see page 1702</a> )	Design compressive strength ( $F_{c\_calc} = F_c$ ( <a href="#">see page 1701</a> ) / gammaC, where gammaC denotes safety factor); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fcj ( <a href="#">see page 1702</a> )	Value that is used only in the French code and denotes Fc ( <a href="#">see page 1701</a> ) (characteristic compressive strength); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Ft ( <a href="#">see page 1702</a> )	Characteristic tensile strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Grade ( <a href="#">see page 1702</a> )	Identifier denoting a predefined concrete class (according to the code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.

**II.22.2 IRConcr\_ACI318\_ConcreteParams Fields**

The fields of the IRConcr\_ACI318\_ConcreteParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AsInStructure ( <a href="#">see page 1742</a> )	Flag indicating if reinforcement definition considers compression resistance of the material currently added to the panel .

**II.22.2.1 AsInStructure****C++**

```
HRESULT get_AsInStructure(VARIANT_BOOL* );
HRESULT put_AsInStructure(VARIANT_BOOL );
```

**C#**

```
public bool AsInStructure { get; set; }
```

**Visual Basic**

```
Public AsInStructure As Boolean
```

**Description**

Flag indicating if reinforcement definition considers compression resistance of the material currently added to the panel .

**Version**

Available since version 3.5.

## II.23 IRConcr\_ACI318\_BarDim

**C++**

```
enum IRConcr_ACI318_BarDim;
```

**C#**

```
public enum IRConcr_ACI318_BarDim;
```

**Visual Basic**

```
Public Enum IRConcr_ACI318_BarDim
```

**Members**

Members	Description
I_ACI318_BR_3 = 0	Available since version 3.5.
I_ACI318_BR_5 = 2	Available since version 3.5.
I_ACI318_BR_6 = 3	Available since version 3.5.
I_ACI318_BR_7 = 4	Available since version 3.5.
I_ACI318_BR_8 = 5	Available since version 3.5.
I_ACI318_BR_9 = 6	Available since version 3.5.
I_ACI318_BR_10 = 7	Available since version 3.5.
I_ACI318_BR_11 = 8	Available since version 3.5.
I_ACI318_BR_14 = 9	Available since version 3.5.
I_ACI318_BR_18 = 10	Available since version 3.5.

**Description**

Available bar diameters for the code ACI 318/99.

**Version**

Available since version 3.5.

## II.24 IRConcr\_ACI318\_MetricBarDim

**C++**

```
enum IRConcr_ACI318_MetricBarDim;
```

**C#**

```
public enum IRConcr_ACI318_MetricBarDim;
```

**Visual Basic**

```
Public Enum IRConcr_ACI318_MetricBarDim
```

## Members

Members	Description
I_ACI318_MBD_5 = 0	Available since version 3.5.
I_ACI318_MBD_6 = 1	Available since version 3.5.
I_ACI318_MBD_8 = 2	Available since version 3.5.
I_ACI318_MBD_10 = 3	Available since version 3.5.
I_ACI318_MBD_12 = 4	Available since version 3.5.
I_ACI318_MBD_14 = 5	Available since version 3.5.
I_ACI318_MBD_16 = 6	Available since version 3.5.
I_ACI318_MBD_20 = 7	Available since version 3.5.
I_ACI318_MBD_25 = 8	Available since version 3.5.
I_ACI318_MBD_32 = 9	Available since version 3.5.
I_ACI318_MBD_40 = 10	Available since version 3.5.

## Description

Available bar diameters for the code ACI 318/99 metric.

## Version

Available since version 3.5.

## II.25 IRConcrReinforceBarType

### C++

```
enum IRConcrReinforceBarType;
```

### C#

```
public enum IRConcrReinforceBarType;
```

### Visual Basic

```
Public Enum IRConcrReinforceBarType
```

## Members

Members	Description
I_RBT_D1_TOP = 0	Available since version 3.5.
I_RBT_D1_BOTTOM = 1	Available since version 3.5.
I_RBT_D2_TOP = 2	Available since version 3.5.
I_RBT_D2_BOTTOM = 3	Available since version 3.5.

## Description

Types (designations) of reinforcing bars singled out during reinforcement definition. .

## Version

Available since version 3.5.

## II.26 IRConcr\_BS8110\_ReinforceData

### Class Hierarchy

### C++

```
interface IRConcr_BS8110_ReinforceData : IDispatch;
```

**C#**

```
public interface IRConcr_BS8110_ReinforceData;
```

**Visual Basic**

```
Public Interface IRConcr_BS8110_ReinforceData
```

**Description**

Interface of reinforcement parameters for the code BS8110.

**Version**

Available since version 3.5.

**II.26.1 IRConcr\_BS8110\_ReinforceData Members**

The following tables list the members exposed by IRConcr\_BS8110\_ReinforceData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Main ( <a href="#">see page 1745</a> )	
◆	PartialSafetyFactors ( <a href="#">see page 1745</a> )	Safety factor.
◆	SLS_ConcreteAge ( <a href="#">see page 1746</a> )	Concrete age (days) .
◆	SLS_Cracking ( <a href="#">see page 1746</a> )	Flag determinig calculation of cracking width.
◆	SLS_CrackingReinfCorection ( <a href="#">see page 1746</a> )	Flag managing automatic correction of cracking width by increasing reinforcement area.
◆	SLS_CreepingCoef ( <a href="#">see page 1747</a> )	Flag indicating if concrete creep coefficient should be considered in reinforcement definition.
◆	SLS_CreepingCoefValue ( <a href="#">see page 1747</a> )	Concrete creep coefficient.
◆	SLS_Deflection ( <a href="#">see page 1747</a> )	Flag determinig calculation of a deflection value.
◆	SLS_DeflectionReinfCorection ( <a href="#">see page 1748</a> )	Flag managing automatic correction of deflections by increasing reinforcement area.
◆	SLS_EnvHumidity ( <a href="#">see page 1748</a> )	Humidity (%).
◆	SLS_Exposure ( <a href="#">see page 1748</a> )	Exposure.
◆	SLS_LoadRatio ( <a href="#">see page 1749</a> )	Ratio of long-term loads to total loads.
◆	SLS_MaxCracking ( <a href="#">see page 1749</a> )	Admissible value of cracking width.
◆	SLS_MaxDeflection ( <a href="#">see page 1749</a> )	Admissible deflection value.

**II.26.2 IRConcr\_BS8110\_ReinforceData Fields**

The fields of the IRConcr\_BS8110\_ReinforceData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Main ( <a href="#">see page 1745</a> )	
◆	PartialSafetyFactors ( <a href="#">see page 1745</a> )	Safety factor.
◆	SLS_ConcreteAge ( <a href="#">see page 1746</a> )	Concrete age (days) .
◆	SLS_Cracking ( <a href="#">see page 1746</a> )	Flag determinig calculation of cracking width.

◆	SLS_CrackingReinfCorection (see page 1746)	Flag managing automatic correction of cracking width by increasing reinforcement area.
◆	SLS_CreepingCoef (see page 1747)	Flag indicating if concrete creep coefficient should be considered in reinforcement definition.
◆	SLS_CreepingCoefValue (see page 1747)	Concrete creep coefficient.
◆	SLS_Deflection (see page 1747)	Flag determining calculation of a deflection value.
◆	SLS_DeflectionReinfCorection (see page 1748)	Flag managing automatic correction of deflections by increasing reinforcement area.
◆	SLS_EnvHumidity (see page 1748)	Humidity (%).
◆	SLS_Exposure (see page 1748)	Exposure.
◆	SLS_LoadRatio (see page 1749)	Ratio of long-term loads to total loads.
◆	SLS_MaxCracking (see page 1749)	Admissible value of cracking width.
◆	SLS_MaxDeflection (see page 1749)	Admissible deflection value.

### II.26.2.1 Main

**C++**

```
HRESULT get_Main(IRConcrReinforceDataMain**);
```

**C#**

```
public IRConcrReinforceDataMain Main { get; }
```

**Visual Basic**

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

### II.26.2.2 PartialSafetyFactors

**C++**

```
HRESULT get_PartialSafetyFactors(IRConcr_BS8110_PartialSafetyFactors*);  
HRESULT put_PartialSafetyFactors(IRConcr_BS8110_PartialSafetyFactors);
```

**C#**

```
public IRConcr_BS8110_PartialSafetyFactors PartialSafetyFactors { get; set; }
```

**Visual Basic**

```
Public PartialSafetyFactors As IRConcr_BS8110_PartialSafetyFactors
```

**Description**

Safety factor.

**Version**

Available since version 3.5.

### II.26.2.3 SLS\_ConcreteAge

**C++**

```
HRESULT get_SLS_ConcreteAge(IRConcr_BS8110_ConcreteAge*);  
HRESULT put_SLS_ConcreteAge(IRConcr_BS8110_ConcreteAge);
```

**C#**

```
public IRConcr_BS8110_ConcreteAge SLS_ConcreteAge { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteAge As IRConcr_BS8110_ConcreteAge
```

**Description**

Concrete age (days).

**Version**

Available since version 3.5.

**II.26.2.4 SLS\_Cracking****C++**

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**Description**

Flag determinig calculation of cracking width.

**Version**

Available since version 3.5.

**II.26.2.5 SLS\_CrackingReinfCorection****C++**

```
HRESULT get_SLS_CrackingReinfCorection(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorection { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorection As Boolean
```

**Description**

Flag managing automatic correction of cracking width by increasing reinforcement area.

**Version**

Available since version 3.5.

**II.26.2.6 SLS\_CreepingCoef****C++**

```
HRESULT get_SLS_CreepingCoef(VARIANT_BOOL* );
HRESULT put_SLS_CreepingCoef(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CreepingCoef { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoef As Boolean
```

**Description**

Flag indicating if concrete creep coefficient should be considered in reinforcement definition.

**Version**

Available since version 3.5.

**II.26.2.7 SLS\_CreepingCoefValue****C++**

```
HRESULT get_SLS_CreepingCoefValue(double* );
HRESULT put_SLS_CreepingCoefValue(double);
```

**C#**

```
public double SLS_CreepingCoefValue { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoefValue As Double
```

**Description**

Concrete creep coefficient.

**Version**

Available since version 3.5.

**II.26.2.8 SLS\_Deflection****C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Description**

Flag determinig calculation of a deflection value.

**Version**

Available since version 3.5.

**II.26.2.9 SLS\_DeflectionReinfCorection****C++**

```
HRESULT get_SLS_DeflectionReinfCorection(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorection { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorection As Boolean
```

**Description**

Flag managing automatic correction of deflections by increasing reinforcement area.

**Version**

Available since version 3.5.

**II.26.2.10 SLS\_EnvHumidity****C++**

```
HRESULT get_SLS_EnvHumidity(short*);  
HRESULT put_SLS_EnvHumidity(short);
```

**C#**

```
public short SLS_EnvHumidity { get; set; }
```

**Visual Basic**

```
Public SLS_EnvHumidity As short
```

**Description**

Humidity (%).

**Version**

Available since version 3.5.

**II.26.2.11 SLS\_Exposure****C++**

```
HRESULT get_SLS_Exposure(IRConcr_BS8110_ExposureRatings*);  
HRESULT put_SLS_Exposure(IRConcr_BS8110_ExposureRatings);
```

**C#**

```
public IRConcr_BS8110_ExposureRatings SLS_Exposure { get; set; }
```

**Visual Basic**

```
Public SLS_Exposure As IRConcr_BS8110_ExposureRatings
```

**Description**

Exposure.

**Version**

Available since version 3.5.

**II.26.2.12 SLS\_LoadRatio****C++**

```
HRESULT get_SLS_LoadRatio(double*);  
HRESULT put_SLS_LoadRatio(double);
```

**C#**

```
public double SLS_LoadRatio { get; set; }
```

**Visual Basic**

```
Public SLS_LoadRatio As double
```

**Description**

Ratio of long-term loads to total loads.

**Version**

Available since version 3.5.

## II.26.2.13 SLS\_MaxCracking

### C++

```
HRESULT get_SLS_MaxCracking(double* );
HRESULT put_SLS_MaxCracking(double);
```

### C#

```
public double SLS_MaxCracking { get; set; }
```

### Visual Basic

```
Public SLS_MaxCracking As Double
```

### Description

Admissible value of cracking width.

### Version

Available since version 3.5.

## II.26.2.14 SLS\_MaxDeflection

### C++

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

### C#

```
public double SLS_MaxDeflection { get; set; }
```

### Visual Basic

```
Public SLS_MaxDeflection As Double
```

### Description

Admissible deflection value.

### Version

Available since version 3.5.

## II.27 IRConcr\_BS8110\_ExposureRatings

### C++

```
enum IRConcr_BS8110_ExposureRatings;
```

### C#

```
public enum IRConcr_BS8110_ExposureRatings;
```

### Visual Basic

```
Public Enum IRConcr_BS8110_ExposureRatings
```

### Members

Members	Description
I_BS8110_ER_MILD = 0	Available since version 3.5.
I_BS8110_ER_MODERATE = 1	Available since version 3.5.
I_BS8110_ER_SEVERE = 2	Available since version 3.5.
I_BS8110_ER VERY_SEVERE = 3	Available since version 3.5.
I_BS8110_ER MOST_SEVERE = 4	Available since version 3.5.

**Description**

Exposure types defined for the code BS 8110. .

**Version**

Available since version 3.5.

**II.28 IRConcr\_BS8110\_ConcreteAge****C++**

```
enum IRConcr_BS8110_ConcreteAge;
```

**C#**

```
public enum IRConcr_BS8110_ConcreteAge;
```

**Visual Basic**

```
Public Enum IRConcr_BS8110_ConcreteAge
```

**Members**

Members	Description
I_BS8110_CA_1 = 0	1 day. Available since version 3.5.
I_BS8110_CA_3 = 1	3 days. Available since version 3.5.
I_BS8110_CA_7 = 2	7 days. Available since version 3.5.
I_BS8110_CA_28 = 3	28 days. Available since version 3.5.
I_BS8110_CA_90 = 4	90 days. Available since version 3.5.
I_BS8110_CA_365 = 5	365 days. Available since version 3.5.

**Description**

Predicted values of concrete age.

**Version**

Available since version 3.5.

**II.29 IRConcr\_BS8110\_ConcreteGrades****C++**

```
enum IRConcr_BS8110_ConcreteGrades;
```

**C#**

```
public enum IRConcr_BS8110_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_BS8110_ConcreteGrades
```

**Members**

Members	Description
I_BS8110(CG)_AUTO = 0	Available since version 3.5.

I_BS8110(CG_C12_15 = 1	Available since version 3.5.
I_BS8110(CG_C16_20 = 2	Available since version 3.5.
I_BS8110(CG_C20_25 = 3	Available since version 3.5.
I_BS8110(CG_C25_30 = 4	Available since version 3.5.
I_BS8110(CG_C30_37 = 5	Available since version 3.5.
I_BS8110(CG_C35_45 = 6	Available since version 3.5.
I_BS8110(CG_C40_50 = 7	Available since version 3.5.
I_BS8110(CG_C45_55 = 8	Available since version 3.5.
I_BS8110(CG_C50_60 = 9	Available since version 3.5.

**Description**

Concrete classes according to the code BS 8110. .

**Version**

Available since version 3.5.

## II.30 IRConcr\_BS8110\_PartialSafetyFactors

**C++**

```
enum IRConcr_BS8110_PartialSafetyFactors;
```

**C#**

```
public enum IRConcr_BS8110_PartialSafetyFactors;
```

**Visual Basic**

```
Public Enum IRConcr_BS8110_PartialSafetyFactors
```

**Members**

Members	Description
I_BS8110_PSF_1985 = 0	Available since version 3.5.
I_BS8110_PSF_1997 = 1	Available since version 3.5.

**Description**

Safety factor types.

**Version**

Available since version 3.5.

## II.31 IRConcrSlabRequiredReinfCalcParams

**Class Hierarchy****C++**

```
interface IRConcrSlabRequiredReinfCalcParams : IDispatch;
```

**C#**

```
public interface IRConcrSlabRequiredReinfCalcParams;
```

**Visual Basic**

```
Public Interface IRConcrSlabRequiredReinfCalcParams
```

**Description**

Calculation parameters for the required (theoretical) reinforcement for plates and shells. .

## Version

Available since version 5.5.

### II.31.1 IRConcrSlabRequiredReinfCalcParams Members

The following tables list the members exposed by IRConcrSlabRequiredReinfCalcParams.

#### Public Fields

	Name	Description
❖	CasesACC ( <a href="#">see page 1753</a> )	
❖	CasesSLS ( <a href="#">see page 1753</a> )	
❖	CasesULS ( <a href="#">see page 1753</a> )	
❖	DisplayErrors ( <a href="#">see page 1754</a> )	
❖	ForcesReduction ( <a href="#">see page 1754</a> )	
❖	GloballyAvgDesginForces ( <a href="#">see page 1754</a> )	
❖	Method ( <a href="#">see page 1754</a> )	Calculation method.
❖	Panels ( <a href="#">see page 1755</a> )	Selection of objects for which calculations will be performed.

### II.31.2 IRConcrSlabRequiredReinfCalcParams Fields

The fields of the IRConcrSlabRequiredReinfCalcParams class are listed here.

#### Public Fields

	Name	Description
❖	CasesACC ( <a href="#">see page 1753</a> )	
❖	CasesSLS ( <a href="#">see page 1753</a> )	
❖	CasesULS ( <a href="#">see page 1753</a> )	
❖	DisplayErrors ( <a href="#">see page 1754</a> )	
❖	ForcesReduction ( <a href="#">see page 1754</a> )	
❖	GloballyAvgDesginForces ( <a href="#">see page 1754</a> )	
❖	Method ( <a href="#">see page 1754</a> )	Calculation method.
❖	Panels ( <a href="#">see page 1755</a> )	Selection of objects for which calculations will be performed.

#### II.31.2.1 CasesACC

##### C++

```
HRESULT get_CasesACC(IRobotSelection**);
```

##### C#

```
public IRobotSelection CasesACC { get; }
```

##### Visual Basic

```
Public ReadOnly CasesACC As IRobotSelection
```

#### Version

Available since version 5.5.

#### II.31.2.2 CasesSLS

##### C++

```
HRESULT get_CasesSLS(IRobotSelection**);
```

**C#**

```
public IRobotSelection CasesSLS { get; }
```

**Visual Basic**

```
Public ReadOnly CasesSLS As IRobotSelection
```

**Version**

Available since version 5.5.

### II.31.2.3 CasesULS

**C++**

```
HRESULT get_CasesULS(IRobotSelection**);
```

**C#**

```
public IRobotSelection CasesULS { get; }
```

**Visual Basic**

```
Public ReadOnly CasesULS As IRobotSelection
```

**Version**

Available since version 5.5.

### II.31.2.4 DisplayErrors

**C++**

```
HRESULT get_DisplayErrors(VARIANT_BOOL*);  
HRESULT put_DisplayErrors(VARIANT_BOOL);
```

**C#**

```
public bool DisplayErrors { get; set; }
```

**Visual Basic**

```
Public DisplayErrors As Boolean
```

**Version**

Available since version 5.5.

### II.31.2.5 ForcesReduction

**C++**

```
HRESULT get_ForceReduction(VARIANT_BOOL*);  
HRESULT put_ForceReduction(VARIANT_BOOL);
```

**C#**

```
public bool ForcesReduction { get; set; }
```

**Visual Basic**

```
Public ForcesReduction As Boolean
```

**Version**

Available since version 5.5.

### II.31.2.6 GloballyAvgDesginForces

**C++**

```
HRESULT get_GloballyAvgDesginForces(VARIANT_BOOL*);  
HRESULT put_GloballyAvgDesginForces(VARIANT_BOOL);
```

**C#**

```
public bool GloballyAvgDesginForces { get; set; }
```

**Visual Basic**

```
Public GloballyAvgDesginForces As Boolean
```

**Version**

Available since version 5.5.

### II.31.2.7 Method

**C++**

```
HRESULT get_Method(IRobotReinforceCalcMethods* );
HRESULT put_Method(IRobotReinforceCalcMethods* );
```

**C#**

```
public IRobotReinforceCalcMethods Method { get; set; }
```

**Visual Basic**

```
Public Method As IRobotReinforceCalcMethods
```

**Description**

Calculation method.

**Version**

Available since version 5.5.

### II.31.2.8 Panels

**C++**

```
HRESULT get_Panels(IRobotSelection** );
```

**C#**

```
public IRobotSelection Panels { get; }
```

**Visual Basic**

```
Public ReadOnly Panels As IRobotSelection
```

**Description**

Selection of objects for which calculations will be performed.

**Version**

Available since version 5.5.

## II.32 IRConcrSlabRequiredReinfEngine

**Class Hierarchy****C++**

```
interface IRConcrSlabRequiredReinfEngine : IDispatch;
```

**C#**

```
public interface IRConcrSlabRequiredReinfEngine;
```

**Visual Basic**

```
Public Interface IRConcrSlabRequiredReinfEngine
```

**Description**

Calculation module for required (theoretical) reinforcement of plates and shells. .

**Version**

Available since version 5.5.

**II.32.1 IRConcrSlabRequiredReinfEngine Members**

The following tables list the members exposed by IRConcrSlabRequiredReinfEngine.

**Public Fields**

	Name	Description
◆	Params (see page 1756)	Calculation parameters.

**Public Methods**

	Name	Description
◆	Calculate (see page 1756)	Function performs calculations of the required (theoretical) reinforcement of plates and shells according to the currently-set parameters. .
◆	GetCalculatedPanels (see page 1757)	Function returns selection of slabs for which results of required reinforcement calculations are available.

**II.32.2 IRConcrSlabRequiredReinfEngine Fields**

The fields of the IRConcrSlabRequiredReinfEngine class are listed here.

**Public Fields**

	Name	Description
◆	Params (see page 1756)	Calculation parameters.

**II.32.2.1 Params****C++**

```
HRESULT get_Params( IRConcrSlabRequiredReinfCalcParams** );
```

**C#**

```
public IRConcrSlabRequiredReinfCalcParams Params { get; }
```

**Visual Basic**

```
Public ReadOnly Params As IRConcrSlabRequiredReinfCalcParams
```

**Description**

Calculation parameters.

**Version**

Available since version 5.5.

**II.32.3 IRConcrSlabRequiredReinfEngine Methods**

The methods of the IRConcrSlabRequiredReinfEngine class are listed here.

## Public Methods

	Name	Description
-Calculates the reinforcement required for plates and shells based on current parameters.	Calculate (see page 1756)	Function performs calculations of the required (theoretical) reinforcement of plates and shells according to the currently-set parameters..
-Returns a selection of slabs for which results of required reinforcement calculations are available.	GetCalculatedPanels (see page 1757)	Function returns selection of slabs for which results of required reinforcement calculations are available.

### II.32.3.1 Calculate

#### C++

```
HRESULT Calculate(VARIANT_BOOL* ret);
```

#### C#

```
public bool Calculate();
```

#### Visual Basic

```
Public Function Calculate() As Boolean
```

#### Description

Function performs calculations of the required (theoretical) reinforcement of plates and shells according to the currently-set parameters..

#### Version

Available since version 5.5.

### II.32.3.2 GetCalculatedPanels

#### C++

```
HRESULT GetCalculatedPanels(IRobotSelection** ret);
```

#### C#

```
public IRobotSelection GetCalculatedPanels();
```

#### Visual Basic

```
Public Function GetCalculatedPanels() As IRobotSelection
```

#### Description

Function returns selection of slabs for which results of required reinforcement calculations are available.

#### Version

Available since version 11.

## II.33 IRConcrReinforceCalcType

#### C++

```
enum IRConcrReinforceCalcType;
```

#### C#

```
public enum IRConcrReinforceCalcType;
```

#### Visual Basic

```
Public Enum IRConcrReinforceCalcType
```

## II.34 IRConcrReinforceData2

### Class Hierarchy

#### C++

```
interface IRConcrReinforceData2 : IRConcrReinforceData;
```

#### C#

```
public interface IRConcrReinforceData2 : IRConcrReinforceData;
```

#### Visual Basic

```
Public Interface IRConcrReinforceData2
```

### II.34.1 IRConcrReinforceData2 Members

The following tables list the members exposed by IRConcrReinforceData2.

#### Public Fields

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (↗ see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	Main (↗ see page 1758)	
◆	ReinforcingSteel (↗ see page 1698)	Available since version 1.7.

#### Public Methods

	Name	Description
◆	GetMainDirection (↗ see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection (↗ see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

### II.34.2 IRConcrReinforceData2 Fields

The fields of the IRConcrReinforceData2 class are listed here.

#### Public Fields

	Name	Description
◆	Main (↗ see page 1758)	

### II.34.2.1 Main

#### C++

```
HRESULT get_Main( IRConcrReinforceDataMain** );
```

#### C#

```
public IRConcrReinforceDataMain Main { get; }
```

#### Visual Basic

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

## II.35 IRConcrReinforceDataMain

#### Class Hierarchy

#### C++

```
interface IRConcrReinforceDataMain : IRConcrReinforceData;
```

#### C#

```
public interface IRConcrReinforceDataMain : IRConcrReinforceData;
```

#### Visual Basic

```
Public Interface IRConcrReinforceDataMain
```

### II.35.1 IRConcrReinforceDataMain Members

The following tables list the members exposed by IRConcrReinforceDataMain.

#### Public Fields

	Name	Description
◆	BarDim_D1_Bot ( <a href="#">see page 1696</a> )	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up ( <a href="#">see page 1696</a> )	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot ( <a href="#">see page 1696</a> )	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up ( <a href="#">see page 1697</a> )	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName ( <a href="#">see page 1697</a> )	Code name Available since version 1.7.
◆	Concrete ( <a href="#">see page 1697</a> )	Available since version 1.7.
◆	Cover_Bot ( <a href="#">see page 1697</a> )	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up ( <a href="#">see page 1698</a> )	Cover - top reinforcement Available since version 1.7.
◆	MembraneReinfInOneLayer ( <a href="#">see page 1760</a> )	Membrane reinforcement in one layer.
◆	MinimumReinf ( <a href="#">see page 1760</a> )	
◆	ReinfCalcType ( <a href="#">see page 1760</a> )	
◆	ReinforcingSteel ( <a href="#">see page 1698</a> )	Available since version 1.7.
◆	UnidirReinf ( <a href="#">see page 1760</a> )	Unidirectional reinforcement.

## Public Methods

	Name	Description
💡	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
💡	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

## II.35.2 IRConcrReinforceDataMain Fields

The fields of the IRConcrReinforceDataMain class are listed here.

### Public Fields

	Name	Description
💡	MembraneReinfInOneLayer (see page 1760)	Membrane reinforcement in one layer.
💡	MinimumReinf (see page 1760)	
💡	ReinfCalcType (see page 1760)	
💡	UnidirReinf (see page 1760)	Unidirectional reinforcement.

### II.35.2.1 MembraneReinfInOneLayer

#### C++

```
HRESULT get_MembraneReinfInOneLayer(VARIANT_BOOL* );
HRESULT put_MembraneReinfInOneLayer(VARIANT_BOOL );
```

#### C#

```
public bool MembraneReinfInOneLayer { get; set; }
```

#### Visual Basic

```
Public MembraneReinfInOneLayer As Boolean
```

#### Description

Membrane reinforcement in one layer.

### II.35.2.2 MinimumReinf

#### C++

```
HRESULT get_MinimumReinf(IRConcrMinimumReinforcementType* );
HRESULT put_MinimumReinf(IRConcrMinimumReinforcementType );
```

#### C#

```
public IRConcrMinimumReinforcementType MinimumReinf { get; set; }
```

#### Visual Basic

```
Public MinimumReinf As IRConcrMinimumReinforcementType
```

#### Version

Available since version 8.5.

### II.35.2.3 ReinfCalcType

**C++**

```
HRESULT get_ReinfCalcType(IRConcrReinforceCalcType* );
HRESULT put_ReinfCalcType(IRConcrReinforceCalcType);
```

**C#**

```
public IRConcrReinforceCalcType ReinfCalcType { get; set; }
```

**Visual Basic**

```
Public ReinfCalcType As IRConcrReinforceCalcType
```

### II.35.2.4 UnidirReinf

**C++**

```
HRESULT get_UnidirReinf(VARIANT_BOOL* );
HRESULT put_UnidirReinf(VARIANT_BOOL);
```

**C#**

```
public bool UnidirReinf { get; set; }
```

**Visual Basic**

```
Public UnidirReinf As Boolean
```

**Description**

Unidirectional reinforcement.

## II.36 IRConcr\_SNIP\_ReinforceData

**Class Hierarchy**

**C++**

```
interface IRConcr_SNIP_ReinforceData : IRConcrReinforceData2;
```

**C#**

```
public interface IRConcr_SNIP_ReinforceData : IRConcrReinforceData2;
```

**Visual Basic**

```
Public Interface IRConcr_SNIP_ReinforceData
```

**Description**

Interface of plate/shell reinforcement parameters for the code "SNIP 2.03.01-84".

### II.36.1 IRConcr\_SNIP\_ReinforceData Members

The following tables list the members exposed by IRConcr\_SNIP\_ReinforceData.

**Public Fields**

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.

◆	BarDim_D2_Up (see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (see page 1697)	Code name Available since version 1.7.
◆	Concrete (see page 1697)	Available since version 1.7.
◆	Cover_Bot (see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	Main (see page 1758)	
◆	ReinforcingSteel (see page 1698)	Available since version 1.7.
◆	SLS_Cracking (see page 1762)	
◆	SLS_CrackingReinfCorrection (see page 1762)	
◆	SLS_Exposure (see page 1762)	

**Public Methods**

	Name	Description
◆	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

**II.36.2 IRConcr\_SNIP\_ReinforceData Fields**

The fields of the IRConcr\_SNIP\_ReinforceData class are listed here.

**Public Fields**

	Name	Description
◆	SLS_Cracking (see page 1762)	
◆	SLS_CrackingReinfCorrection (see page 1762)	
◆	SLS_Exposure (see page 1762)	

**II.36.2.1 SLS\_Cracking****C++**

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**II.36.2.2 SLS\_CrackingReinfCorrection****C++**

```
HRESULT get_SLS_CrackingReinfCorrection(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorrection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorrection { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorrection As Boolean
```

### II.36.2.3 SLS\_Exposure

**C++**

```
HRESULT get_SLS_Exposure(IRConcr_SNIP_Exposure* );
HRESULT put_SLS_Exposure(IRConcr_SNIP_Exposure);
```

**C#**

```
public IRConcr_SNIP_Exposure SLS_Exposure { get; set; }
```

**Visual Basic**

```
Public SLS_Exposure As IRConcr_SNIP_Exposure
```

## II.37 IRConcr\_SNIP\_SteelGrades

**C++**

```
enum IRConcr_SNIP_SteelGrades;
```

**C#**

```
public enum IRConcr_SNIP_SteelGrades;
```

**Visual Basic**

```
Public Enum IRConcr_SNIP_SteelGrades
```

## II.38 IRConcr\_SNIP\_ConcreteGrades

**C++**

```
enum IRConcr_SNIP_ConcreteGrades;
```

**C#**

```
public enum IRConcr_SNIP_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_SNIP_ConcreteGrades
```

## II.39 IRConcr\_SNIP\_ConcreteTypes

**C++**

```
enum IRConcr_SNIP_ConcreteTypes;
```

**C#**

```
public enum IRConcr_SNIP_ConcreteTypes;
```

**Visual Basic**

```
Public Enum IRConcr_SNIP_ConcreteTypes
```

## II.40 IRConcr\_SNIP\_CuringMethods

### C++

```
enum IRConcr_SNIP_CuringMethods;
```

### C#

```
public enum IRConcr_SNIP_CuringMethods;
```

### Visual Basic

```
Public Enum IRConcr_SNIP_CuringMethods
```

## II.41 IRConcr\_SNIP\_ConcreteParams

### Class Hierarchy

### C++

```
interface IRConcr_SNIP_ConcreteParams : IRConcrConcreteParams;
```

### C#

```
public interface IRConcr_SNIP_ConcreteParams : IRConcrConcreteParams;
```

### Visual Basic

```
Public Interface IRConcr_SNIP_ConcreteParams
```

### II.41.1 IRConcr\_SNIP\_ConcreteParams Members

The following tables list the members exposed by IRConcr\_SNIP\_ConcreteParams.

#### Public Fields

	Name	Description
❖	ConcretingInLayers (see page 1765)	
❖	CuringMethod (see page 1765)	
❖	Ec (see page 1701)	Elasticity modulus for concrete; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fc (see page 1701)	Characteristic compressive strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fc_calc (see page 1702)	Design compressive strength ( $F_{c\_calc} = F_c$ (see page 1701) / gammaC, where gammaC denotes safety factor); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Fcj (see page 1702)	Value that is used only in the French code and denotes Fc (see page 1701) (characteristic compressive strength); parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Ft (see page 1702)	Characteristic tensile strength; parameter assumes value that equals -1 when the value of the appropriate material assigned to a panel is taken for calculations Available since version 1.7.
❖	Grade (see page 1702)	Identifier denoting a predefined concrete class (according to the code it should be compared with the appropriate set of defined identifiers) Available since version 1.7.
❖	HighHumidity (see page 1765)	

	Type ( <a href="#">see page 1765</a> )	
--	----------------------------------------	--

## II.41.2 IRConcr\_SNIP\_ConcreteParams Fields

The fields of the IRConcr\_SNIP\_ConcreteParams class are listed here.

### Public Fields

	Name	Description
	ConcretingInLayers ( <a href="#">see page 1765</a> )	
	CuringMethod ( <a href="#">see page 1765</a> )	
	HighHumidity ( <a href="#">see page 1765</a> )	
	Type ( <a href="#">see page 1765</a> )	

### II.41.2.1 ConcretingInLayers

#### C++

```
HRESULT get_ConcretingInLayers(VARIANT_BOOL* );
HRESULT put_ConcretingInLayers(VARIANT_BOOL);
```

#### C#

```
public bool ConcretingInLayers { get; set; }
```

#### Visual Basic

```
Public ConcretingInLayers As Boolean
```

### II.41.2.2 CuringMethod

#### C++

```
HRESULT get_CuringMethod(IRConcr_SNIP_CuringMethods* );
HRESULT put_CuringMethod(IRConcr_SNIP_CuringMethods);
```

#### C#

```
public IRConcr_SNIP_CuringMethods CuringMethod { get; set; }
```

#### Visual Basic

```
Public CuringMethod As IRConcr_SNIP_CuringMethods
```

### II.41.2.3 HighHumidity

#### C++

```
HRESULT get_HighHumidity(VARIANT_BOOL* );
HRESULT put_HighHumidity(VARIANT_BOOL);
```

#### C#

```
public bool HighHumidity { get; set; }
```

#### Visual Basic

```
Public HighHumidity As Boolean
```

### II.41.2.4 Type

#### C++

```
HRESULT get_Type(IRConcr_SNIP_ConcreteTypes* );
HRESULT put_Type(IRConcr_SNIP_ConcreteTypes);
```

#### C#

```
public IRConcr_SNIP_ConcreteTypes Type { get; set; }
```

**Visual Basic**

```
Public Type As IRConcr_SNIP_ConcreteTypes
```

**II.42 IRConcr\_SNIP\_Exposure****C++**

```
enum IRConcr_SNIP_Exposure;
```

**C#**

```
public enum IRConcr_SNIP_Exposure;
```

**Visual Basic**

```
Public Enum IRConcr_SNIP_Exposure
```

**II.43 IRConcr\_EC2\_ConcreteGrades****C++**

```
enum IRConcr_EC2_ConcreteGrades;
```

**C#**

```
public enum IRConcr_EC2_ConcreteGrades;
```

**Visual Basic**

```
Public Enum IRConcr_EC2_ConcreteGrades
```

**Members**

Members	Description
I_CEC2CG_AUTOMATIC = -1	Material defined for a panel will be used.

**Description**

Concrete classes according to Eurocode 2.

**II.44 IRConcr\_EC2\_ITALIAN\_SteelGrades****C++**

```
enum IRConcr_EC2_ITALIAN_SteelGrades;
```

**C#**

```
public enum IRConcr_EC2_ITALIAN_SteelGrades;
```

**Visual Basic**

```
Public Enum IRConcr_EC2_ITALIAN_SteelGrades
```

**II.45 IRConcr\_EC2\_ExposureRatings****C++**

```
enum IRConcr_EC2_ExposureRatings;
```

**C#**

```
public enum IRConcr_EC2_ExposureRatings;
```

## Visual Basic

```
Public Enum IRConcr_EC2_ExposureRatings
```

### Description

Exposure types according to Eurocode 2.

## II.46 IRConcr\_EC2\_ReinforceData

### Class Hierarchy

#### C++

```
interface IRConcr_EC2_ReinforceData : IRConcrReinforceData2;
```

#### C#

```
public interface IRConcr_EC2_ReinforceData : IRConcrReinforceData2;
```

### Visual Basic

```
Public Interface IRConcr_EC2_ReinforceData
```

## II.46.1 IRConcr\_EC2\_ReinforceData Members

The following tables list the members exposed by IRConcr\_EC2\_ReinforceData.

### Public Fields

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.
◆	Cover_Up (↗ see page 1698)	Cover - top reinforcement Available since version 1.7.
◆	Main (↗ see page 1758)	
◆	NAD (↗ see page 1768)	
◆	ReinforcingSteel (↗ see page 1698)	Available since version 1.7.
◆	SLS_ConcreteAge (↗ see page 1769)	
◆	SLS_Cracking (↗ see page 1769)	
◆	SLS_CrackingReinfCorrect (↗ see page 1769)	
◆	SLS_CreepingCoef (↗ see page 1769)	
◆	SLS_CreepingCoefValue (↗ see page 1769)	
◆	SLS_Deflection (↗ see page 1770)	
◆	SLS_DeflectionReinfCorrect (↗ see page 1770)	
◆	SLS_EnvHumidityVal (↗ see page 1770)	

❖	SLS_Exposure (see page 1770)	
❖	SLS_MaxCracking (see page 1770)	
❖	SLS_MaxDeflection (see page 1771)	

**Public Methods**

	Name	Description
❖	GetMainDirection (see page 1698)	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
❖	SetMainDirection (see page 1699)	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

**II.46.2 IRConcr\_EC2\_ReinforceData Fields**

The fields of the IRConcr\_EC2\_ReinforceData class are listed here.

**Public Fields**

	Name	Description
❖	NAD (see page 1768)	
❖	SLS_ConcreteAge (see page 1769)	
❖	SLS_Cracking (see page 1769)	
❖	SLS_CrackingReinfCorrect (see page 1769)	
❖	SLS_CreepingCoef (see page 1769)	
❖	SLS_CreepingCoefValue (see page 1769)	
❖	SLS_Deflection (see page 1770)	
❖	SLS_DeflectionReinfCorrect (see page 1770)	
❖	SLS_EnvHumidityVal (see page 1770)	
❖	SLS_Exposure (see page 1770)	
❖	SLS_MaxCracking (see page 1770)	
❖	SLS_MaxDeflection (see page 1771)	

**II.46.2.1 NAD****C++**

```
HRESULT get_NAD( IRConcr_EC2_NAD* );
HRESULT put_NAD( IRConcr_EC2_NAD );
```

**C#**

```
public IRConcr_EC2_NAD NAD { get; set; }
```

**Visual Basic**

```
Public NAD As IRConcr_EC2_NAD
```

**II.46.2.2 SLS\_ConcreteAge****C++**

```
HRESULT get_SLS_ConcreteAge(short* );
HRESULT put_SLS_ConcreteAge(short);
```

**C#**

```
public short SLS_ConcreteAge { get; set; }
```

**Visual Basic**

```
Public SLS_ConcreteAge As short
```

**II.46.2.3 SLS\_Cracking****C++**

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**II.46.2.4 SLS\_CrackingReinfCorrect****C++**

```
HRESULT get_SLS_CrackingReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorrect As Boolean
```

**II.46.2.5 SLS\_CreepingCoef****C++**

```
HRESULT get_SLS_CreepingCoef(VARIANT_BOOL* );
HRESULT put_SLS_CreepingCoef(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CreepingCoef { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoef As Boolean
```

**II.46.2.6 SLS\_CreepingCoefValue****C++**

```
HRESULT get_SLS_CreepingCoefValue(double* );
HRESULT put_SLS_CreepingCoefValue(double);
```

**C#**

```
public double SLS_CreepingCoefValue { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoefValue As double
```

**II.46.2.7 SLS\_Deflection****C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**II.46.2.8 SLS\_DeflectionReinfCorrect****C++**

```
HRESULT get_SLS_DeflectionReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrect As Boolean
```

**II.46.2.9 SLS\_EnvHumidityVal****C++**

```
HRESULT get_SLS_EnvHumidityVal(double* );
HRESULT put_SLS_EnvHumidityVal(double);
```

**C#**

```
public double SLS_EnvHumidityVal { get; set; }
```

**Visual Basic**

```
Public SLS_EnvHumidityVal As double
```

**II.46.2.10 SLS\_Exposure****C++**

```
HRESULT get_SLS_Exposure(IRConcr_EC2_ExposureRatings* );
HRESULT put_SLS_Exposure(IRConcr_EC2_ExposureRatings);
```

**C#**

```
public IRConcr_EC2_ExposureRatings SLS_Exposure { get; set; }
```

**Visual Basic**

```
Public SLS_Exposure As IRConcr_EC2_ExposureRatings
```

**II.46.2.11 SLS\_MaxCracking****C++**

```
HRESULT get_SLS_MaxCracking(double* );
HRESULT put_SLS_MaxCracking(double);
```

**C#**

```
public double SLS_MaxCracking { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCracking As double
```

**II.46.2.12 SLS\_MaxDeflection****C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As double
```

**II.47 IRConcr\_EC2\_NAD****C++**

```
enum IRConcr_EC2_NAD;
```

**C#**

```
public enum IRConcr_EC2_NAD;
```

**Visual Basic**

```
Public Enum IRConcr_EC2_NAD
```

**II.48 IRConcrMinimumReinforcementType****C++**

```
enum IRConcrMinimumReinforcementType;
```

**C#**

```
public enum IRConcrMinimumReinforcementType;
```

**Visual Basic**

```
Public Enum IRConcrMinimumReinforcementType
```

**Members**

Members	Description
I_CMRT_NONE = 0	Available since version 8.5.
I_CMRT_FOR_NONZERO_AS = 1	Available since version 8.5.
I_CMRT_FOR_WHOLE_PANEL = 2	Available since version 8.5.

**Version**

Available since version 8.5.

**II.49 IRConcr\_IS2000\_EnvironmentType****C++**

```
enum IRConcr_IS2000_EnvironmentType;
```

**C#**

```
public enum IRConcr_IS2000_EnvironmentType;
```

## Visual Basic

```
Public Enum IRConcr_IS2000_EnvironmentType
```

## Members

Members	Description
I_IS2000_ER_MILD = 0	Available since version 9.5.
I_IS2000_ER_MODERATE = 1	Available since version 9.5.
I_IS2000_ER_SEVERE = 2	Available since version 9.5.
I_IS2000_ER_VERY_SEVERE = 3	Available since version 9.5.
I_IS2000_ER_MOST_SEVERE = 4	Available since version 9.5.

## Description

Exposure types defined for the IS 456:2000 code.

## Version

Available since version 9.5.

# II.50 IRConcr\_IS2000\_ReinforceData

## Class Hierarchy

## C++

```
interface IRConcr_IS2000_ReinforceData : IRConcrReinforceData;
```

## C#

```
public interface IRConcr_IS2000_ReinforceData : IRConcrReinforceData;
```

## Visual Basic

```
Public Interface IRConcr_IS2000_ReinforceData
```

## Description

Reinforcement parameters for the Indian code IS 456:2000.

## Version

Available since version 9.5.

## II.50.1 IRConcr\_IS2000\_ReinforceData Members

The following tables list the members exposed by IRConcr\_IS2000\_ReinforceData.

## Public Fields

	Name	Description
◆	BarDim_D1_Bot (↗ see page 1696)	Diameter of reinforcing bars of bottom main reinforcement Available since version 1.7.
◆	BarDim_D1_Up (↗ see page 1696)	Diameter of reinforcing bars of top main reinforcement Available since version 1.7.
◆	BarDim_D2_Bot (↗ see page 1696)	Reinforcing bar diameter of bottom reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	BarDim_D2_Up (↗ see page 1697)	Reinforcing bar diameter of top reinforcement in the direction perpendicular to the main reinforcement Available since version 1.7.
◆	CodeName (↗ see page 1697)	Code name Available since version 1.7.
◆	Concrete (↗ see page 1697)	Available since version 1.7.
◆	Cover_Bot (↗ see page 1697)	Cover - bottom reinforcement Available since version 1.7.

◆	Cover_Up ( <a href="#">see page 1698</a> )	Cover - top reinforcement Available since version 1.7.
◆	Main ( <a href="#">see page 1774</a> )	
◆	ReinforcingSteel ( <a href="#">see page 1698</a> )	Available since version 1.7.
◆	SLS_ConcreteAge ( <a href="#">see page 1774</a> )	
◆	SLS_Cracking ( <a href="#">see page 1775</a> )	
◆	SLS_CrackingReinfCorrect ( <a href="#">see page 1775</a> )	
◆	SLS_CreepingCoeff ( <a href="#">see page 1775</a> )	
◆	SLS_CreepingCoeffValue ( <a href="#">see page 1775</a> )	
◆	SLS_Deflection ( <a href="#">see page 1776</a> )	
◆	SLS_DeflectionReinfCorrect ( <a href="#">see page 1776</a> )	
◆	SLS_ExposureBot ( <a href="#">see page 1776</a> )	
◆	SLS_ExposureTop ( <a href="#">see page 1776</a> )	
◆	SLS_LoadRatio ( <a href="#">see page 1777</a> )	
◆	SLS_MaxCrackingBot ( <a href="#">see page 1777</a> )	
◆	SLS_MaxCrackingBotEnabled ( <a href="#">see page 1777</a> )	
◆	SLS_MaxCrackingTop ( <a href="#">see page 1777</a> )	
◆	SLS_MaxCrackingTopEnabled ( <a href="#">see page 1778</a> )	
◆	SLS_MaxDeflection ( <a href="#">see page 1778</a> )	

## Public Methods

	Name	Description
◆	GetMainDirection ( <a href="#">see page 1698</a> )	Function returns the main reinforcement direction. Interpretation of returned coordinate values depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT). Available since version 1.7.
◆	SetMainDirection ( <a href="#">see page 1699</a> )	Function defines the main reinforcement direction. Parameter interpretation depends on the manner of determining direction (direction vector coordinates, if the direction is determined by means of the vector, ALONG_VECTOR, or the point coordinates in the polar system - POLAR_POINT, in the remaining cases the coordinate values are ignored). Available since version 1.7.

## II.50.2 IRConcr\_IS2000\_ReinforceData Fields

The fields of the IRConcr\_IS2000\_ReinforceData class are listed here.

### Public Fields

	Name	Description
◆	Main ( <a href="#">see page 1774</a> )	
◆	SLS_ConcreteAge ( <a href="#">see page 1774</a> )	
◆	SLS_Cracking ( <a href="#">see page 1775</a> )	

◆	SLS_CrackingReinfCorrect (☞ see page 1775)	
◆	SLS_CreepingCoeff (☞ see page 1775)	
◆	SLS_CreepingCoeffValue (☞ see page 1775)	
◆	SLS_Deflection (☞ see page 1776)	
◆	SLS_DeflectionReinfCorrect (☞ see page 1776)	
◆	SLS_ExposureBot (☞ see page 1776)	
◆	SLS_ExposureTop (☞ see page 1776)	
◆	SLS_LoadRatio (☞ see page 1777)	
◆	SLS_MaxCrackingBot (☞ see page 1777)	
◆	SLS_MaxCrackingBotEnabled (☞ see page 1777)	
◆	SLS_MaxCrackingTop (☞ see page 1777)	
◆	SLS_MaxCrackingTopEnabled (☞ see page 1778)	
◆	SLS_MaxDeflection (☞ see page 1778)	

## II.50.2.1 Main

### C++

```
HRESULT get_Main(IRConcrReinforceDataMain**);
```

### C#

```
public IRConcrReinforceDataMain Main { get; }
```

### Visual Basic

```
Public ReadOnly Main As IRConcrReinforceDataMain
```

### Version

Available since version 9.5.

## II.50.2.2 SLS\_ConcreteAge

### C++

```
HRESULT get_SLS_ConcreteAge(short*);  
HRESULT put_SLS_ConcreteAge(short);
```

### C#

```
public short SLS_ConcreteAge { get; set; }
```

### Visual Basic

```
Public SLS_ConcreteAge As short
```

### Version

Available since version 9.5.

### II.50.2.3 SLS\_Cracking

**C++**

```
HRESULT get_SLS_Cracking(VARIANT_BOOL* );
HRESULT put_SLS_Cracking(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Cracking { get; set; }
```

**Visual Basic**

```
Public SLS_Cracking As Boolean
```

**Version**

Available since version 9.5.

### II.50.2.4 SLS\_CrackingReinfCorrect

**C++**

```
HRESULT get_SLS_CrackingReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_CrackingReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CrackingReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_CrackingReinfCorrect As Boolean
```

**Version**

Available since version 9.5.

### II.50.2.5 SLS\_CreepingCoeff

**C++**

```
HRESULT get_SLS_CreepingCoeff(VARIANT_BOOL* );
HRESULT put_SLS_CreepingCoeff(VARIANT_BOOL);
```

**C#**

```
public bool SLS_CreepingCoeff { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoeff As Boolean
```

**Version**

Available since version 9.5.

### II.50.2.6 SLS\_CreepingCoeffValue

**C++**

```
HRESULT get_SLS_CreepingCoeffValue(double* );
HRESULT put_SLS_CreepingCoeffValue(double);
```

**C#**

```
public double SLS_CreepingCoeffValue { get; set; }
```

**Visual Basic**

```
Public SLS_CreepingCoeffValue As double
```

**Version**

Available since version 9.5.

**II.50.2.7 SLS\_Deflection****C++**

```
HRESULT get_SLS_Deflection(VARIANT_BOOL* );
HRESULT put_SLS_Deflection(VARIANT_BOOL);
```

**C#**

```
public bool SLS_Deflection { get; set; }
```

**Visual Basic**

```
Public SLS_Deflection As Boolean
```

**Version**

Available since version 9.5.

**II.50.2.8 SLS\_DeflectionReinfCorrect****C++**

```
HRESULT get_SLS_DeflectionReinfCorrect(VARIANT_BOOL* );
HRESULT put_SLS_DeflectionReinfCorrect(VARIANT_BOOL);
```

**C#**

```
public bool SLS_DeflectionReinfCorrect { get; set; }
```

**Visual Basic**

```
Public SLS_DeflectionReinfCorrect As Boolean
```

**Version**

Available since version 9.5.

**II.50.2.9 SLS\_ExposureBot****C++**

```
HRESULT get_SLS_ExposureBot(short* );
HRESULT put_SLS_ExposureBot(short);
```

**C#**

```
public short SLS_ExposureBot { get; set; }
```

**Visual Basic**

```
Public SLS_ExposureBot As short
```

**Version**

Available since version 9.5.

**II.50.2.10 SLS\_ExposureTop****C++**

```
HRESULT get_SLS_ExposureTop(short* );
HRESULT put_SLS_ExposureTop(short);
```

**C#**

```
public short SLS_ExposureTop { get; set; }
```

**Visual Basic**

```
Public SLS_ExposureTop As short
```

**Version**

Available since version 9.5.

**II.50.2.11 SLS\_LoadRatio****C++**

```
HRESULT get_SLS_LoadRatio(double*);  
HRESULT put_SLS_LoadRatio(double);
```

**C#**

```
public double SLS_LoadRatio { get; set; }
```

**Visual Basic**

```
Public SLS_LoadRatio As double
```

**Version**

Available since version 9.5.

**II.50.2.12 SLS\_MaxCrackingBot****C++**

```
HRESULT get_SLS_MaxCrackingBot(double*);  
HRESULT put_SLS_MaxCrackingBot(double);
```

**C#**

```
public double SLS_MaxCrackingBot { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCrackingBot As double
```

**Version**

Available since version 9.5.

**II.50.2.13 SLS\_MaxCrackingBotEnabled****C++**

```
HRESULT get_SLS_MaxCrackingBotEnabled(VARIANT_BOOL*);  
HRESULT put_SLS_MaxCrackingBotEnabled(VARIANT_BOOL);
```

**C#**

```
public bool SLS_MaxCrackingBotEnabled { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCrackingBotEnabled As Boolean
```

**Version**

Available since version 9.5.

**II.50.2.14 SLS\_MaxCrackingTop****C++**

```
HRESULT get_SLS_MaxCrackingTop(double*);  
HRESULT put_SLS_MaxCrackingTop(double);
```

**C#**

```
public double SLS_MaxCrackingTop { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCrackingTop As Double
```

**Version**

Available since version 9.5.

## II.50.2.15 SLS\_MaxCrackingTopEnabled

**C++**

```
HRESULT get_SLS_MaxCrackingTopEnabled(VARIANT_BOOL* );
HRESULT put_SLS_MaxCrackingTopEnabled(VARIANT_BOOL);
```

**C#**

```
public bool SLS_MaxCrackingTopEnabled { get; set; }
```

**Visual Basic**

```
Public SLS_MaxCrackingTopEnabled As Boolean
```

**Version**

Available since version 9.5.

## II.50.2.16 SLS\_MaxDeflection

**C++**

```
HRESULT get_SLS_MaxDeflection(double* );
HRESULT put_SLS_MaxDeflection(double);
```

**C#**

```
public double SLS_MaxDeflection { get; set; }
```

**Visual Basic**

```
Public SLS_MaxDeflection As Double
```

**Version**

Available since version 9.5.

## II.51 IRConcrPlateCodeService2

**Class Hierarchy****C++**

```
interface IRConcrPlateCodeService2 : IDispatch;
```

**C#**

```
public interface IRConcrPlateCodeService2;
```

**Visual Basic**

```
Public Interface IRConcrPlateCodeService2
```

**Version**

Available since version 12.5.

## II.51.1 IRConcrPlateCodeService2 Members

The following tables list the members exposed by IRConcrPlateCodeService2.

### Public Fields

	Name	Description
❖	RobotVersion (see page 1779)	
❖	ShowLongTermCracking (see page 1779)	

## II.51.2 IRConcrPlateCodeService2 Fields

The fields of the IRConcrPlateCodeService2 class are listed here.

### Public Fields

	Name	Description
❖	RobotVersion (see page 1779)	
❖	ShowLongTermCracking (see page 1779)	

### II.51.2.1 RobotVersion

#### C++

```
HRESULT get_RobotVersion(BSTR*);
```

#### C#

```
public String RobotVersion { get; }
```

#### Visual Basic

```
Public ReadOnly RobotVersion As String
```

#### Version

Available since version 12.5.

### II.51.2.2 ShowLongTermCracking

#### C++

```
HRESULT get_ShowLongTermCracking(VARIANT_BOOL*);
```

#### C#

```
public bool ShowLongTermCracking { get; }
```

#### Visual Basic

```
Public ReadOnly ShowLongTermCracking As Boolean
```

#### Version

Available since version 12.5.

## III Design of RC members

Available since version 7.5.

## Enumerations

	Name	Description
	IRConcrBeamCalcPointDefinitionType	Available methods of definition of calculation points for a beam. ( <a href="#">see page 1785</a> )

## Interfaces

	Name	Description
	IRConcrMemberRequiredReinfEngine	Calculation module for the design of required reinforcement of RC members.
	IRConcrMemberRequiredReinfCalcParams	Calculation parameters for the design of required reinforcement of RC members.

## III.1 IRConcrMemberRequiredReinfEngine

### Class Hierarchy

#### C++

```
interface IRConcrMemberRequiredReinfEngine : IDispatch;
```

#### C#

```
public interface IRConcrMemberRequiredReinfEngine;
```

#### Visual Basic

```
Public Interface IRConcrMemberRequiredReinfEngine
```

#### Description

Calculation module for the design of required reinforcement of RC members.

#### Version

Available since version 7.5.

### III.1.1 IRConcrMemberRequiredReinfEngine Members

The following tables list the members exposed by IRConcrMemberRequiredReinfEngine.

#### Public Fields

	Name	Description
	Params ( <a href="#">see page 1781</a> )	Calculation parameters.

#### Public Methods

	Name	Description
	Calculate ( <a href="#">see page 1781</a> )	Function performs calculations of the required reinforcement of RC members according to the currently-set parameters.
	GetCalculatedMembers ( <a href="#">see page 1781</a> )	Function returns selection of members for which results of required reinforcement calculations are available.

### III.1.2 IRConcrMemberRequiredReinfEngine Fields

The fields of the IRConcrMemberRequiredReinfEngine class are listed here.

#### Public Fields

	Name	Description
	Params ( <a href="#">see page 1781</a> )	Calculation parameters.

### III.1.2.1 Params

#### C++

```
HRESULT get_Params( IRConcrMemberRequiredReinfCalcParams** );
```

#### C#

```
public IRConcrMemberRequiredReinfCalcParams Params { get; }
```

#### Visual Basic

```
Public ReadOnly Params As IRConcrMemberRequiredReinfCalcParams
```

#### Description

Calculation parameters.

#### Version

Available since version 7.5.

## III.1.3 IRConcrMemberRequiredReinfEngine Methods

The methods of the IRConcrMemberRequiredReinfEngine class are listed here.

#### Public Methods

	Name	Description
💡	Calculate ( see page 1781)	Function performs calculations of the required reinforcement of RC members according to the currently-set parameters.
💡	GetCalculatedMembers ( see page 1781)	Function returns selection of members for which results of required reinforcement calculations are available.

### III.1.3.1 Calculate

#### C++

```
HRESULT Calculate(VARIANT_BOOL* ret);
```

#### C#

```
public bool Calculate();
```

#### Visual Basic

```
Public Function Calculate() As Boolean
```

#### Description

Function performs calculations of the required reinforcement of RC members according to the currently-set parameters.

#### Version

Available since version 7.5.

### III.1.3.2 GetCalculatedMembers

#### C++

```
HRESULT GetCalculatedMembers(IRobotSelection** ret);
```

#### C#

```
public IRobotSelection GetCalculatedMembers();
```

#### Visual Basic

```
Public Function GetCalculatedMembers() As IRobotSelection
```

**Description**

Function returns selection of members for which results of required reinforcement calculations are available.

**Version**

Available since version 11.

## III.2 IRConcrMemberRequiredReinfCalcParams

**Class Hierarchy****C++**

```
interface IRConcrMemberRequiredReinfCalcParams : IDispatch;
```

**C#**

```
public interface IRConcrMemberRequiredReinfCalcParams;
```

**Visual Basic**

```
Public Interface IRConcrMemberRequiredReinfCalcParams
```

**Description**

Calculation parameters for the design of required reinforcement of RC members.

**Version**

Available since version 7.5.

### III.2.1 IRConcrMemberRequiredReinfCalcParams Members

The following tables list the members exposed by IRConcrMemberRequiredReinfCalcParams.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BeamPointsType (↗ see page 1783)	Method of definition of division points for a beam.
❖	BeamPointsValue (↗ see page 1783)	Value determining points of the beam division.
❖	CasesALS (↗ see page 1784)	
❖	CasesSLS (↗ see page 1784)	
❖	CasesULS (↗ see page 1784)	
❖	CombALS (↗ see page 1784)	
❖	CombSLS (↗ see page 1785)	
❖	CombULS (↗ see page 1785)	
❖	Members (↗ see page 1785)	Bar selection.

### III.2.2 IRConcrMemberRequiredReinfCalcParams Fields

The fields of the IRConcrMemberRequiredReinfCalcParams class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BeamPointsType (↗ see page 1783)	Method of definition of division points for a beam.
❖	BeamPointsValue (↗ see page 1783)	Value determining points of the beam division.
❖	CasesALS (↗ see page 1784)	
❖	CasesSLS (↗ see page 1784)	

❖	CasesULS ( [ see page 1784 )	
❖	CombALS ( [ see page 1784 )	
❖	CombSLS ( [ see page 1785 )	
❖	CombULS ( [ see page 1785 )	
❖	Members ( [ see page 1785 )	Bar selection.

### III.2.2.1 BeamPointsType

#### C++

```
HRESULT get_BeamPointsType(IRConcrBeamCalcPointDefinitionType* );
HRESULT put_BeamPointsType(IRConcrBeamCalcPointDefinitionType);
```

#### C#

```
public IRConcrBeamCalcPointDefinitionType BeamPointsType { get; set; }
```

#### Visual Basic

```
Public BeamPointsType As IRConcrBeamCalcPointDefinitionType
```

#### Description

Method of definition of division points for a beam.

#### Version

Available since version 7.5.

### III.2.2.2 BeamPointsValue

#### C++

```
HRESULT get_BeamPointsValue(double* );
HRESULT put_BeamPointsValue(double);
```

#### C#

```
public double BeamPointsValue { get; set; }
```

#### Visual Basic

```
Public BeamPointsValue As Double
```

#### Description

Value determining points of the beam division.

#### Version

Available since version 7.5.

### III.2.2.3 CasesALS

#### C++

```
HRESULT get_CasesALS(IRobotSelection** );
```

#### C#

```
public IRobotSelection CasesALS { get; }
```

#### Visual Basic

```
Public ReadOnly CasesALS As IRobotSelection
```

#### Version

Available since version 7.5.

### III.2.2.4 CasesSLS

**C++**

```
HRESULT get_CasesSLS(IRobotSelection**);
```

**C#**

```
public IRobotSelection CasesSLS { get; }
```

**Visual Basic**

```
Public ReadOnly CasesSLS As IRobotSelection
```

**Version**

Available since version 7.5.

### III.2.2.5 CasesULS

**C++**

```
HRESULT get_CasesULS(IRobotSelection**);
```

**C#**

```
public IRobotSelection CasesULS { get; }
```

**Visual Basic**

```
Public ReadOnly CasesULS As IRobotSelection
```

**Version**

Available since version 7.5.

### III.2.2.6 CombALS

**C++**

```
HRESULT get_CombALS(VARIANT_BOOL*);  
HRESULT put_CombALS(VARIANT_BOOL);
```

**C#**

```
public bool CombALS { get; set; }
```

**Visual Basic**

```
Public CombALS As Boolean
```

**Version**

Available since version 7.5.

### III.2.2.7 CombSLS

**C++**

```
HRESULT get_CombSLS(VARIANT_BOOL*);  
HRESULT put_CombSLS(VARIANT_BOOL);
```

**C#**

```
public bool CombSLS { get; set; }
```

**Visual Basic**

```
Public CombSLS As Boolean
```

**Version**

Available since version 7.5.

### III.2.2.8 CombULS

#### C++

```
HRESULT get_CombULS(VARIANT_BOOL* );
HRESULT put_CombULS(VARIANT_BOOL);
```

#### C#

```
public bool CombULS { get; set; }
```

#### Visual Basic

```
Public CombULS As Boolean
```

#### Version

Available since version 7.5.

### III.2.2.9 Members

#### C++

```
HRESULT get_Members(IRobotSelection** );
```

#### C#

```
public IRobotSelection Members { get; }
```

#### Visual Basic

```
Public ReadOnly Members As IRobotSelection
```

#### Description

Bar selection.

#### Version

Available since version 7.5.

## III.3 IRConcrBeamCalcPointDefinitionType

#### C++

```
enum IRConcrBeamCalcPointDefinitionType;
```

#### C#

```
public enum IRConcrBeamCalcPointDefinitionType;
```

#### Visual Basic

```
Public Enum IRConcrBeamCalcPointDefinitionType
```

#### Members

Members	Description
I_CBCPDT_POINT_COUNT = 0	Number of points for beam. Available since version 7.5.
I_CBCPDT_POINT_SPACING = 1	Distance between successive division points expressed in meters. Available since version 7.5.

#### Description

Available methods of definition of calculation points for a beam.

**Version**

Available since version 7.5.

## IV IRConcrBarSectionData

**Class Hierarchy****C++**

```
interface IRConcrBarSectionData : IDispatch;
```

**C#**

```
public interface IRConcrBarSectionData;
```

**Visual Basic**

```
Public Interface IRConcrBarSectionData
```

**Description**

A (↗ see page 1787) simplified interface describing an RC member section.

### IV.1 IRConcrBarSectionData Members

The following tables list the members exposed by IRConcrBarSectionData.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	A (↗ see page 1787)	
❖	Ac (↗ see page 1787)	
❖	Ap (↗ see page 1787)	
❖	B (↗ see page 1788)	
❖	Bf (↗ see page 1788)	
❖	Bfb (↗ see page 1788)	
❖	H (↗ see page 1788)	
❖	Hf (↗ see page 1788)	
❖	Hfb (↗ see page 1789)	
❖	N (↗ see page 1789)	
❖	Per (↗ see page 1789)	
❖	Type (↗ see page 1789)	Section geometry type.

### IV.2 IRConcrBarSectionData Fields

The fields of the IRConcrBarSectionData class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	A (↗ see page 1787)	
❖	Ac (↗ see page 1787)	
❖	Ap (↗ see page 1787)	
❖	B (↗ see page 1788)	
❖	Bf (↗ see page 1788)	
❖	Bfb (↗ see page 1788)	

◆	H (see page 1788)	
◆	Hf (see page 1788)	
◆	Hfb (see page 1789)	
◆	N (see page 1789)	
◆	Per (see page 1789)	
◆	Type (see page 1789)	Section geometry type.

## IV.2.1 A

### C++

```
HRESULT get_A(double* );
HRESULT put_A(double);
```

### C#

```
public double A { get; set; }
```

### Visual Basic

```
Public A As double
```

## IV.2.2 Ac

### C++

```
HRESULT get_Ac(double* );
HRESULT put_Ac(double);
```

### C#

```
public double Ac { get; set; }
```

### Visual Basic

```
Public Ac As double
```

## IV.2.3 Ap

### C++

```
HRESULT get_Ap(double* );
HRESULT put_Ap(double);
```

### C#

```
public double Ap { get; set; }
```

### Visual Basic

```
Public Ap As double
```

## IV.2.4 B

### C++

```
HRESULT get_B(double* );
HRESULT put_B(double);
```

### C#

```
public double B { get; set; }
```

### Visual Basic

```
Public B As double
```

## IV.2.5 Bf

**C++**

```
HRESULT get_Bf(double* );
HRESULT put_Bf(double);
```

**C#**

```
public double Bf { get; set; }
```

**Visual Basic**

```
Public Bf As Double
```

## IV.2.6 Bfb

**C++**

```
HRESULT get_Bfb(double* );
HRESULT put_Bfb(double);
```

**C#**

```
public double Bfb { get; set; }
```

**Visual Basic**

```
Public Bfb As Double
```

## IV.2.7 H

**C++**

```
HRESULT get_H(double* );
HRESULT put_H(double);
```

**C#**

```
public double H { get; set; }
```

**Visual Basic**

```
Public H As Double
```

## IV.2.8 Hf

**C++**

```
HRESULT get_Hf(double* );
HRESULT put_Hf(double);
```

**C#**

```
public double Hf { get; set; }
```

**Visual Basic**

```
Public Hf As Double
```

## IV.2.9 Hfb

**C++**

```
HRESULT get_Hfb(double* );
HRESULT put_Hfb(double);
```

**C#**

```
public double Hfb { get; set; }
```

**Visual Basic**

```
Public Hfb As double
```

**IV.2.10 N****C++**

```
HRESULT get_N(int* );
HRESULT put_N(int );
```

**C#**

```
public int N { get; set; }
```

**Visual Basic**

```
Public N As int
```

**IV.2.11 Per****C++**

```
HRESULT get_Per(double* );
HRESULT put_Per(double );
```

**C#**

```
public double Per { get; set; }
```

**Visual Basic**

```
Public Per As double
```

**IV.2.12 Type****C++**

```
HRESULT get_Type(IRConcrBarSectionGeometryType* );
HRESULT put_Type(IRConcrBarSectionGeometryType );
```

**C#**

```
public IRConcrBarSectionGeometryType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRConcrBarSectionGeometryType
```

**Description**

Section geometry type.

**V IRConcrBarSectionGeometryType****C++**

```
enum IRConcrBarSectionGeometryType;
```

**C#**

```
public enum IRConcrBarSectionGeometryType;
```

**Visual Basic**

```
Public Enum IRConcrBarSectionGeometryType
```

**Description**

Types of geometry of an RC member section .

## VI IRConcrCalcEngine

### Class Hierarchy

#### C++

```
interface IRConcrCalcEngine : IDispatch;
```

#### C#

```
public interface IRConcrCalcEngine;
```

### Visual Basic

```
Public Interface IRConcrCalcEngine
```

### Description

RC calculation module.

### Version

Available since version 5.5.

## VI.1 IRConcrCalcEngine Members

The following tables list the members exposed by IRConcrCalcEngine.

### Public Fields

	Name	Description
❖	MemberRequiredReinf (see page 1791)	Module calculating the required reinforcement for RC members.
❖	SlabRequiredReinf (see page 1791)	Module for calculation of the required (theoretical) reinforcement for plates and shells .

## VI.2 IRConcrCalcEngine Fields

The fields of the IRConcrCalcEngine class are listed here.

### Public Fields

	Name	Description
❖	MemberRequiredReinf (see page 1791)	Module calculating the required reinforcement for RC members.
❖	SlabRequiredReinf (see page 1791)	Module for calculation of the required (theoretical) reinforcement for plates and shells .

### VI.2.1 MemberRequiredReinf

#### C++

```
HRESULT get_MemberRequiredReinf( IRConcrMemberRequiredReinfEngine** );
```

#### C#

```
public IRConcrMemberRequiredReinfEngine MemberRequiredReinf { get; }
```

### Visual Basic

```
Public ReadOnly MemberRequiredReinf As IRConcrMemberRequiredReinfEngine
```

### Description

Module calculating the required reinforcement for RC members.

**Version**

Available since version 7.5.

**VI.2.2 SlabRequiredReinf****C++**

```
HRESULT get_SlabRequiredReinf( IRConcrSlabRequiredReinfEngine** );
```

**C#**

```
public IRConcrSlabRequiredReinfEngine SlabRequiredReinf { get; }
```

**Visual Basic**

```
Public ReadOnly SlabRequiredReinf As IRConcrSlabRequiredReinfEngine
```

**Description**

Module for calculation of the required (theoretical) reinforcement for plates and shells .

**Version**

Available since version 5.5.

**VII Supplementing RC modules with new codes**

Available since version 12.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRConcrCodeColumnCommand ( <a href="#">see page 1795</a> )	
	IRConcrCodeBeamCommand ( <a href="#">see page 1798</a> )	

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRConcrCodeService ( <a href="#">see page 1792</a> )	RConcrCodeService allows code-specific definitions and calculations for provided reinforcement of RC elements. Available from 12.0 version.
	IRConcrCodeColumn ( <a href="#">see page 1793</a> )	
	IRConcrCodeReport ( <a href="#">see page 1795</a> )	
	IRConcrCodeBeam ( <a href="#">see page 1796</a> )	

**VII.1 IRConcrCodeService****Class Hierarchy****C++**

```
interface IRConcrCodeService : IDispatch;
```

**C#**

```
public interface IRConcrCodeService;
```

**Visual Basic**

```
Public Interface IRConcrCodeService
```

## Description

RConcrCodeService allows code-specific definitions and calculations for provided reinforcement of RC elements. Available from 12.0 version.

## Version

Available since version 12.

### VII.1.1 IRConcrCodeService Members

The following tables list the members exposed by IRConcrCodeService.

#### Public Fields

	Name	Description
◆	Beam ( <a href="#">see page 1792</a> )	
◆	Column ( <a href="#">see page 1793</a> )	

#### Public Methods

	Name	Description
◆	IsConcrComponentServed ( <a href="#">see page 1793</a> )	

### VII.1.2 IRConcrCodeService Fields

The fields of the IRConcrCodeService class are listed here.

#### Public Fields

	Name	Description
◆	Beam ( <a href="#">see page 1792</a> )	
◆	Column ( <a href="#">see page 1793</a> )	

#### VII.1.2.1 Beam

##### C++

```
HRESULT get_Beam( IRConcrCodeBeam\*\* );
```

##### C#

```
public IRConcrCodeBeam Beam { get; }
```

##### Visual Basic

```
Public ReadOnly Beam As IRConcrCodeBeam
```

## Version

Available since version 12.

#### VII.1.2.2 Column

##### C++

```
HRESULT get_Column( IRConcrCodeColumn\*\* );
```

##### C#

```
public IRConcrCodeColumn Column { get; }
```

##### Visual Basic

```
Public ReadOnly Column As IRConcrCodeColumn
```

**Version**

Available since version 12.

**VII.1.3 IRConcrCodeService Methods**

The methods of the IRConcrCodeService class are listed here.

**Public Methods**

	Name	Description
≡	IsConcrComponentServed (see page 1793)	

**VII.1.3.1 IsConcrComponentServed****C++**

```
HRESULT IsConcrComponentServed(IRobotProjectComponentType _module, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsConcrComponentServed(IRobotProjectComponentType _module);
```

**Visual Basic**

```
Public Function IsConcrComponentServed(_module As IRobotProjectComponentType) As Boolean
```

**Version**

Available since version 12.

**VII.2 IRConcrCodeColumn****Class Hierarchy****C++**

```
interface IRConcrCodeColumn : IDispatch;
```

**C#**

```
public interface IRConcrCodeColumn;
```

**Visual Basic**

```
Public Interface IRConcrCodeColumn
```

**Version**

Available since version 12.

**VII.2.1 IRConcrCodeColumn Members**

The following tables list the members exposed by IRConcrCodeColumn.

**Public Methods**

	Name	Description
≡	IsCommandEnabled (see page 1794)	
≡	Verification (see page 1794)	

**VII.2.2 IRConcrCodeColumn Methods**

The methods of the IRConcrCodeColumn class are listed here.

## Public Methods

	Name	Description
💡	IsCommandEnabled (see page 1794)	
💡	Verification (see page 1794)	

### VII.2.2.1 IsCommandEnabled

#### C++

```
HRESULT IsCommandEnabled(IRConcrCodeColumnCommand _command, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsCommandEnabled(IRConcrCodeColumnCommand _command);
```

#### Visual Basic

```
Public Function IsCommandEnabled(_command As IRConcrCodeColumnCommand) As Boolean
```

#### Version

Available since version 12.

### VII.2.2.2 Verification

#### C++

```
HRESULT Verification(IRConcrColumn* _column, IRConcrCodeReport* _report, VARIANT_BOOL* ret);
```

#### C#

```
public bool Verification(IRConcrColumn _column, IRConcrCodeReport _report);
```

#### Visual Basic

```
Public Function Verification(ByRef _column As IRConcrColumn, ByRef _report As IRConcrCodeReport) As Boolean
```

#### Version

Available since version 12.

## VII.3 IRConcrCodeColumnCommand

#### C++

```
enum IRConcrCodeColumnCommand;
```

#### C#

```
public enum IRConcrCodeColumnCommand;
```

#### Visual Basic

```
Public Enum IRConcrCodeColumnCommand
```

#### Members

Members	Description
I_CCCC_CL_VERIFICATION = 1	Available since version 12.

#### Version

Available since version 12.

## VII.4 IRConcrCodeReport

### Class Hierarchy

#### C++

```
interface IRConcrCodeReport : IDispatch;
```

#### C#

```
public interface IRConcrCodeReport;
```

### Visual Basic

```
Public Interface IRConcrCodeReport
```

### Version

Available since version 12.

## VII.4.1 IRConcrCodeReport Members

The following tables list the members exposed by IRConcrCodeReport.

### Public Methods

	Name	Description
≡	AddError (see page 1796)	
≡	AddMessage (see page 1796)	
≡	AddWarning (see page 1796)	

## VII.4.2 IRConcrCodeReport Methods

The methods of the IRConcrCodeReport class are listed here.

### Public Methods

	Name	Description
≡	AddError (see page 1796)	
≡	AddMessage (see page 1796)	
≡	AddWarning (see page 1796)	

### VII.4.2.1 AddError

#### C++

```
HRESULT AddError(BSTR _text);
```

#### C#

```
public void AddError(String _text);
```

### Visual Basic

```
Public Sub AddError(_text As String)
```

### Version

Available since version 12.

### VII.4.2.2 AddMessage

#### C++

```
HRESULT AddMessage(BSTR _message);
```

**C#**

```
public void AddMessage(String _message);
```

**Visual Basic**

```
Public Sub AddMessage(_message As String)
```

**Version**

Available since version 12.

**VII.4.2.3 AddWarning****C++**

```
HRESULT AddWarning(BSTR _warning);
```

**C#**

```
public void AddWarning(String _warning);
```

**Visual Basic**

```
Public Sub AddWarning(_warning As String)
```

**Version**

Available since version 12.

**VII.5 IRConcrCodeBeam****Class Hierarchy****C++**

```
interface IRConcrCodeBeam : IDispatch;
```

**C#**

```
public interface IRConcrCodeBeam;
```

**Visual Basic**

```
Public Interface IRConcrCodeBeam
```

**Version**

Available since version 12.

**VII.5.1 IRConcrCodeBeam Members**

The following tables list the members exposed by IRConcrCodeBeam.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	IsCommandEnabled (↗ see page 1797)	
≡	Verification (↗ see page 1797)	

**VII.5.2 IRConcrCodeBeam Methods**

The methods of the IRConcrCodeBeam class are listed here.

## Public Methods

	Name	Description
✖	IsCommandEnabled (see page 1797)	
✖	Verification (see page 1797)	

### VII.5.2.1 IsCommandEnabled

#### C++

```
HRESULT IsCommandEnabled(IRConcrCodeBeamCommand _command, VARIANT_BOOL* ret);
```

#### C#

```
public bool IsCommandEnabled(IRConcrCodeBeamCommand _command);
```

#### Visual Basic

```
Public Function IsCommandEnabled(_command As IRConcrCodeBeamCommand) As Boolean
```

#### Version

Available since version 12.

### VII.5.2.2 Verification

#### C++

```
HRESULT Verification(IRConcrBeam* _beam, IRConcrCodeReport* _report, VARIANT_BOOL* ret);
```

#### C#

```
public bool Verification(IRConcrBeam _beam, IRConcrCodeReport _report);
```

#### Visual Basic

```
Public Function Verification(ByRef _beam As IRConcrBeam, ByRef _report As IRConcrCodeReport) As Boolean
```

#### Version

Available since version 12.

## VII.6 IRConcrCodeBeamCommand

#### C++

```
enum IRConcrCodeBeamCommand;
```

#### C#

```
public enum IRConcrCodeBeamCommand;
```

#### Visual Basic

```
Public Enum IRConcrCodeBeamCommand
```

#### Members

Members	Description
I_CCBC_BM_VERIFICATION = 1	Available since version 12.

#### Version

Available since version 12.

# Steel and timber design

## Enumerations

	Name	Description
	IRDimStreamType (see page 1881)	
	IRDimMembDefType (see page 1884)	
	IRDimMembDefMatType (see page 1884)	
	IRDimMembDefLengthDataType (see page 1884)	
	IRDimMembDefBucklingDataType (see page 1885)	
	IRDimMembDefDispDataType (see page 1885)	
	IRDimMembDefDeflDataType (see page 1885)	
	IRDimEffDefParamType (see page 1889)	
	IRDimEffDefDirType (see page 1890)	
	IRDimEffDefIntPsType (see page 1890)	
	IRDimMatDefType (see page 1899)	
	IRDimMatDefValType (see page 1899)	
	IRDimMatDefLongExValType (see page 1900)	
	IRDimMatDefDblExValType (see page 1900)	
	IRDimProfDefType (see page 1903)	
	IRDimProfDefItemType (see page 1904)	
	IRDimProfDefValType (see page 1904)	
	IRDimCalcStateFlagType (see page 1906)	
	IRDimCalcStateParamType (see page 1907)	
	IRDimCalcStateParamValue (see page 1907)	
	IRDimCalcStateValueType (see page 1907)	
	IRDimMembCalcRetValue (see page 1910)	
	IRDimMembCalcBuckType (see page 1911)	
	IRDimMembResTableLineType (see page 1911)	

	IRDimMembResTableComp (see page 1911)	
	IRDimUnitType (see page 1926)	

**Interfaces**

	Name	Description
	IRDimStream (see page 1881)	
	IRDimMembDef (see page 1885)	
	IRDimEffDef (see page 1890)	
	IRDimMatDef (see page 1900)	
	IRDimProfDef (see page 1904)	
	IRDimCalcState (see page 1907)	
	IRDimMembSrv (see page 1909)	
	IRDimMembRes (see page 1911)	
	IRDimMembCalc (see page 1919)	
	IRDimCodeService (see page 1923)	
	IRDimClient (see page 1925)	
	IRDimUnits (see page 1926)	
	IRDimServer (see page 1928)	Server responsible for design and verification of steel and timber members.

## I EC3 Code

Available since version 3.5.

**Enumerations**

	Name	Description
	IRDimLoadTypeEC3 (see page 1808)	
	IRDimLaterBuckTypeEC3 (see page 1809)	
	IRDimLoadLevelEC3 (see page 1809)	
	IRDimLatBuckCoeffDiagramEC3 (see page 1809)	
	IRDimBuckDiagramEC3 (see page 1810)	
	IRDimYieldStrengthTypeEC3 (see page 1844)	

**Interfaces**

	Name	Description
	IRDimMembParamsEC3 (see page 1800)	
	IRDimCodeResEC3 (see page 1810)	

### I.1 IRDimMembParamsEC3

**Class Hierarchy**

**C++**

```
interface IRDimMembParamsEC3 : IDispatch;
```

**C#**

```
public interface IRDimMembParamsEC3;
```

**Visual Basic**

```
Public Interface IRDimMembParamsEC3
```

**Version**

Available since version 3.5.

**I.1.1 IRDimMembParamsEC3 Members**

The following tables list the members exposed by IRDimMembParamsEC3.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BuckLengthCoeffY ( <a href="#">see page 1802</a> )	
◆	BuckLengthCoeffZ ( <a href="#">see page 1803</a> )	
◆	BucklingDiagramY ( <a href="#">see page 1803</a> )	
◆	BucklingDiagramZ ( <a href="#">see page 1803</a> )	
◆	HotRolledPipes ( <a href="#">see page 1803</a> )	
◆	LatBuckType ( <a href="#">see page 1804</a> )	
◆	LatCoeffLowerFlange ( <a href="#">see page 1804</a> )	
◆	LatCoeffLowerFlangeValue ( <a href="#">see page 1804</a> )	
◆	LatCoeffUpperFlange ( <a href="#">see page 1804</a> )	
◆	LatCoeffUpperFlangeValue ( <a href="#">see page 1805</a> )	
◆	LoadLevel ( <a href="#">see page 1805</a> )	
◆	LoadLevelValue ( <a href="#">see page 1805</a> )	
◆	LoadTypeY ( <a href="#">see page 1805</a> )	
◆	LoadTypeZ ( <a href="#">see page 1806</a> )	
◆	MaterCoeffGamma0 ( <a href="#">see page 1806</a> )	
◆	MaterCoeffGamma1 ( <a href="#">see page 1806</a> )	
◆	RelLimitDeflUy ( <a href="#">see page 1806</a> )	
◆	RelLimitdeflUz ( <a href="#">see page 1807</a> )	
◆	TensAreaNetGros ( <a href="#">see page 1807</a> )	
◆	TubeControl ( <a href="#">see page 1807</a> )	
◆	YieldStrengthType ( <a href="#">see page 1808</a> )	
◆	YieldStrengthValue ( <a href="#">see page 1808</a> )	

## I.1.2 IRDimMembParamsEC3 Fields

The fields of the IRDimMembParamsEC3 class are listed here.

### Public Fields

	Name	Description
◆	BuckLengthCoeffY (↗ see page 1802)	
◆	BuckLengthCoeffZ (↗ see page 1803)	
◆	BucklingDiagramY (↗ see page 1803)	
◆	BucklingDiagramZ (↗ see page 1803)	
◆	HotRolledPipes (↗ see page 1803)	
◆	LatBuckType (↗ see page 1804)	
◆	LatCoeffLowerFlange (↗ see page 1804)	
◆	LatCoeffLowerFlangeValue (↗ see page 1804)	
◆	LatCoeffUpperFlange (↗ see page 1804)	
◆	LatCoeffUpperFlangeValue (↗ see page 1805)	
◆	LoadLevel (↗ see page 1805)	
◆	LoadLevelValue (↗ see page 1805)	
◆	LoadTypeY (↗ see page 1805)	
◆	LoadTypeZ (↗ see page 1806)	
◆	MaterCoeffGamma0 (↗ see page 1806)	
◆	MaterCoeffGamma1 (↗ see page 1806)	
◆	RelLimitDeflUy (↗ see page 1806)	
◆	RelLimitdeflUz (↗ see page 1807)	
◆	TensAreaNetGros (↗ see page 1807)	
◆	TubeControl (↗ see page 1807)	
◆	YieldStrengthType (↗ see page 1808)	
◆	YieldStrengthValue (↗ see page 1808)	

### I.1.2.1 BuckLengthCoeffY

#### C++

```
HRESULT get_BuckLengthCoeffY(double* );
HRESULT put_BuckLengthCoeffY(double);
```

#### C#

```
public double BuckLengthCoeffY { get; set; }
```

#### Visual Basic

```
Public BuckLengthCoeffY As Double
```

**Version**

Available since version 3.

**I.1.2.2 BuckLengthCoeffZ****C++**

```
HRESULT get_BuckLengthCoeffZ(double* );
HRESULT put_BuckLengthCoeffZ(double);
```

**C#**

```
public double BuckLengthCoeffZ { get; set; }
```

**Visual Basic**

```
Public BuckLengthCoeffZ As Double
```

**Version**

Available since version 3.

**I.1.2.3 BucklingDiagramY****C++**

```
HRESULT get_BucklingDiagramY(IRDimBuckDiagramEC3* );
HRESULT put_BucklingDiagramY(IRDimBuckDiagramEC3);
```

**C#**

```
public IRDimBuckDiagramEC3 BucklingDiagramY { get; set; }
```

**Visual Basic**

```
Public BucklingDiagramY As IRDimBuckDiagramEC3
```

**Version**

Available since version 3.

**I.1.2.4 BucklingDiagramZ****C++**

```
HRESULT get_BucklingDiagramZ(IRDimBuckDiagramEC3* );
HRESULT put_BucklingDiagramZ(IRDimBuckDiagramEC3);
```

**C#**

```
public IRDimBuckDiagramEC3 BucklingDiagramZ { get; set; }
```

**Visual Basic**

```
Public BucklingDiagramZ As IRDimBuckDiagramEC3
```

**Version**

Available since version 3.

**I.1.2.5 HotRolledPipes****C++**

```
HRESULT get_HotRolledPipes(VARIANT_BOOL* );
HRESULT put_HotRolledPipes(VARIANT_BOOL);
```

**C#**

```
public bool HotRolledPipes { get; set; }
```

**Visual Basic**

```
Public HotRolledPipes As Boolean
```

**Version**

Available since version 3.5.

**I.1.2.6 LatBuckType****C++**

```
HRESULT get_LatBuckType(IRDimLaterBuckTypeEC3* );
HRESULT put_LatBuckType(IRDimLaterBuckTypeEC3);
```

**C#**

```
public IRDimLaterBuckTypeEC3 LatBuckType { get; set; }
```

**Visual Basic**

```
Public LatBuckType As IRDimLaterBuckTypeEC3
```

**Version**

Available since version 3.

**I.1.2.7 LatCoeffLowerFlange****C++**

```
HRESULT get_LatCoeffLowerFlange(IRDimLatBuckCoeffDiagramEC3* );
HRESULT put_LatCoeffLowerFlange(IRDimLatBuckCoeffDiagramEC3);
```

**C#**

```
public IRDimLatBuckCoeffDiagramEC3 LatCoeffLowerFlange { get; set; }
```

**Visual Basic**

```
Public LatCoeffLowerFlange As IRDimLatBuckCoeffDiagramEC3
```

**Version**

Available since version 3.

**I.1.2.8 LatCoeffLowerFlangeValue****C++**

```
HRESULT get_LatCoeffLowerFlangeValue(double* );
HRESULT put_LatCoeffLowerFlangeValue(double);
```

**C#**

```
public double LatCoeffLowerFlangeValue { get; set; }
```

**Visual Basic**

```
Public LatCoeffLowerFlangeValue As Double
```

**Version**

Available since version 3.

**I.1.2.9 LatCoeffUpperFlange****C++**

```
HRESULT get_LatCoeffUpperFlange(IRDimLatBuckCoeffDiagramEC3* );
HRESULT put_LatCoeffUpperFlange(IRDimLatBuckCoeffDiagramEC3);
```

**C#**

```
public IRDimLatBuckCoeffDiagramEC3 LatCoeffUpperFlange { get; set; }
```

**Visual Basic**

```
Public LatCoeffUpperFlange As IRDimLatBuckCoeffDiagramEC3
```

**Version**

Available since version 3.

**I.1.2.10 LatCoeffUpperFlangeValue****C++**

```
HRESULT get_LatCoeffUpperFlangeValue(double*);  
HRESULT put_LatCoeffUpperFlangeValue(double);
```

**C#**

```
public double LatCoeffUpperFlangeValue { get; set; }
```

**Visual Basic**

```
Public LatCoeffUpperFlangeValue As Double
```

**Version**

Available since version 3.

**I.1.2.11 LoadLevel****C++**

```
HRESULT get_LoadLevel(IRDimLoadLevelEC3*);  
HRESULT put_LoadLevel(IRDimLoadLevelEC3);
```

**C#**

```
public IRDimLoadLevelEC3 LoadLevel { get; set; }
```

**Visual Basic**

```
Public LoadLevel As IRDimLoadLevelEC3
```

**Version**

Available since version 3.

**I.1.2.12 LoadLevelValue****C++**

```
HRESULT get_LoadLevelValue(double*);  
HRESULT put_LoadLevelValue(double);
```

**C#**

```
public double LoadLevelValue { get; set; }
```

**Visual Basic**

```
Public LoadLevelValue As Double
```

**Version**

Available since version 3.

**I.1.2.13 LoadTypeY****C++**

```
HRESULT get_LoadTypeY(IRDimLoadTypeEC3*);
```

```
HRESULT put_LoadTypeY(IRDIMLoadTypeEC3);
```

**C#**

```
public IRDIMLoadTypeEC3 LoadTypeY { get; set; }
```

**Visual Basic**

```
Public LoadTypeY As IRDIMLoadTypeEC3
```

**Version**

Available since version 3.

### I.1.2.14 LoadTypeZ

**C++**

```
HRESULT get_LoadTypeZ(IRDIMLoadTypeEC3*);  
HRESULT put_LoadTypeZ(IRDIMLoadTypeEC3);
```

**C#**

```
public IRDIMLoadTypeEC3 LoadTypeZ { get; set; }
```

**Visual Basic**

```
Public LoadTypeZ As IRDIMLoadTypeEC3
```

**Version**

Available since version 3.

### I.1.2.15 MaterCoeffGamma0

**C++**

```
HRESULT get_MaterCoeffGamma0(double*);  
HRESULT put_MaterCoeffGamma0(double);
```

**C#**

```
public double MaterCoeffGamma0 { get; set; }
```

**Visual Basic**

```
Public MaterCoeffGamma0 As double
```

**Version**

Available since version 3.

### I.1.2.16 MaterCoeffGamma1

**C++**

```
HRESULT get_MaterCoeffGamma1(double*);  
HRESULT put_MaterCoeffGamma1(double);
```

**C#**

```
public double MaterCoeffGamma1 { get; set; }
```

**Visual Basic**

```
Public MaterCoeffGamma1 As double
```

**Version**

Available since version 3.

### I.1.2.17 RelLimitDeflUy

#### C++

```
HRESULT get_RelLimitDeflUy(double*);  
HRESULT put_RelLimitDeflUy(double);
```

#### C#

```
public double RelLimitDeflUy { get; set; }
```

#### Visual Basic

```
Public RelLimitDeflUy As double
```

#### Version

Available since version 3.

### I.1.2.18 RelLimitdeflUz

#### C++

```
HRESULT get_RelLimitdeflUz(double*);  
HRESULT put_RelLimitdeflUz(double);
```

#### C#

```
public double RelLimitdeflUz { get; set; }
```

#### Visual Basic

```
Public RelLimitdeflUz As double
```

#### Version

Available since version 3.

### I.1.2.19 TensAreaNetGros

#### C++

```
HRESULT get_TensAreaNetGros(double*);  
HRESULT put_TensAreaNetGros(double);
```

#### C#

```
public double TensAreaNetGros { get; set; }
```

#### Visual Basic

```
Public TensAreaNetGros As double
```

#### Version

Available since version 3.

### I.1.2.20 TubeControl

#### C++

```
HRESULT get_TubeControl(VARIANT_BOOL*);  
HRESULT put_TubeControl(VARIANT_BOOL);
```

#### C#

```
public bool TubeControl { get; set; }
```

#### Visual Basic

```
Public TubeControl As Boolean
```

**Version**

Available since version 3.

**I.1.2.21 YieldStrengthType****C++**

```
HRESULT get_YieldStrengthType(IRDIMYieldStrengthTypeEC3* );
HRESULT put_YieldStrengthType(IRDIMYieldStrengthTypeEC3);
```

**C#**

```
public IRDIMYieldStrengthTypeEC3 YieldStrengthType { get; set; }
```

**Visual Basic**

```
Public YieldStrengthType As IRDIMYieldStrengthTypeEC3
```

**Version**

Available since version 3.5.

**I.1.2.22 YieldStrengthValue****C++**

```
HRESULT get_YieldStrengthValue(double* );
HRESULT put_YieldStrengthValue(double);
```

**C#**

```
public double YieldStrengthValue { get; set; }
```

**Visual Basic**

```
Public YieldStrengthValue As Double
```

**Version**

Available since version 3.

**I.2 IRDimLoadTypeEC3****C++**

```
enum IRDimLoadTypeEC3;
```

**C#**

```
public enum IRDimLoadTypeEC3;
```

**Visual Basic**

```
Public Enum IRDimLoadTypeEC3
```

**Members**

Members	Description
I_DLT_EC3_UNIFORM_LOAD = 0	Available since version 3.
I_DLT_EC3_UNIFORM_MOMENT = 1	Available since version 3.
I_DLT_EC3_CONCENTRATED_FORCE = 2	Available since version 3.
I_DLT_EC3_MOMENTS_AT_ENDS = 3	Available since version 3.

**Version**

Available since version 3.5.

## I.3 IRDimLaterBuckTypeEC3

### C++

```
enum IRDimLaterBuckTypeEC3;
```

### C#

```
public enum IRDimLaterBuckTypeEC3;
```

### Visual Basic

```
Public Enum IRDimLaterBuckTypeEC3
```

### Members

Members	Description
I_DLBT_EC3_SYMMETR_LOADED = 0	Available since version 3.
I_DLBT_EC3_CANTILEVER = 1	Available since version 3.
I_DLBT_EC3_NO = 2	Available since version 3.

### Version

Available since version 3.5.

## I.4 IRDimLoadLevelEC3

### C++

```
enum IRDimLoadLevelEC3;
```

### C#

```
public enum IRDimLoadLevelEC3;
```

### Visual Basic

```
Public Enum IRDimLoadLevelEC3
```

### Members

Members	Description
I_DLL_EC3_UPP_EDGE_LOADED = 0	Available since version 3.
I_DLL_EC3_UPP_SECT_PAR_LOADED = 1	Available since version 3.
I_DLL_EC3_CENTER_LOADED = 2	Available since version 3.
I_DLL_EC3_LOW_SEC_PART_LOADED = 3	Available since version 3.
I_DLL_EC3_LOW_EDGE_LOADED = 4	Available since version 3.

### Version

Available since version 3.5.

## I.5 IRDimLatBuckCoeffDiagramEC3

### C++

```
enum IRDimLatBuckCoeffDiagramEC3;
```

### C#

```
public enum IRDimLatBuckCoeffDiagramEC3;
```

**Visual Basic**

```
Public Enum IRDimLatBuckCoeffDiagramEC3
```

**Members**

Members	Description
I_DLBCD_EC3_CANTILEVER_2_0 = 0	Available since version 3.
I_DLBCD_EC3_INTERNAL_BRACINGS = 5	Available since version 3.
I_DLBCD_EC3_NO = 4	Available since version 3.
I_DLBCD_EC3_PINNED_PINNED_1_0 = 1	Available since version 3.
I_DLBCD_EC3_STIFF_STIFF_0_5 = 2	Available since version 3.
I_DLBCD_EC3_USER_DEFINED = 3	Available since version 3.

**Version**

Available since version 3.5.

**I.6 IRDimBuckDiagramEC3****C++**

```
enum IRDimBuckDiagramEC3;
```

**C#**

```
public enum IRDimBuckDiagramEC3;
```

**Visual Basic**

```
Public Enum IRDimBuckDiagramEC3
```

**Members**

Members	Description
I_DBD_EC3_AUTO = 12	Available since version 3.
I_DBD_EC3_CANTILEVER_2_0 = 3	Available since version 3.
I_DBD_EC3_INTERN_ADJBAR_6 = 8	Available since version 3.
I_DBD_EC3_INTERNAL_BRACINGS = 13	Available since version 3.
I_DBD_EC3_NO = 11	Available since version 3.
I_DBD_EC3_PINNED_ADJBAR_1 = 4	Available since version 3.
I_DBD_EC3_PINNED_ADJBAR_3 = 6	Available since version 3.
I_DBD_EC3_PINNED_PINNED_1_0 = 0	Available since version 3.
I_DBD_EC3_PINNED_STIFF_0_7 = 2	Available since version 3.
I_DBD_EC3_STIFF_ADJBAR_1 = 5	Available since version 3.
I_DBD_EC3_STIFF_ADJBAR_3 = 7	Available since version 3.
I_DBD_EC3_STIFF_STIFF_0_5 = 1	Available since version 3.
I_DBD_EC3_TRUSS_CHORD_0_9 = 9	Available since version 3.
I_DBD_EC3_TRUSS_DIAGONAL_0_8 = 10	Available since version 3.
I_DBD_EC3_USER_DEFINED = -1	Available since version 3.

**Version**

Available since version 3.5.

**I.7 IRDimCodeResEC3****Class Hierarchy**

**C++**

```
interface IRDimCodeResEC3 : IDispatch;
```

**C#**

```
public interface IRDimCodeResEC3;
```

**Visual Basic**

```
Public Interface IRDimCodeResEC3
```

**Version**

Available since version 3.5.

**I.7.1 IRDimCodeResEC3 Members**

The following tables list the members exposed by IRDimCodeResEC3.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	AxForceExcentrEny (↗ see page 1818)	
◆	AxForcExcentrEnz (↗ see page 1819)	
◆	BendParamBetaMlty (↗ see page 1819)	
◆	BendParamBetaMltz (↗ see page 1819)	
◆	BendParamBetaMz (↗ see page 1819)	
◆	BendParamBetMy (↗ see page 1820)	
◆	BuckCoeffMinXi (↗ see page 1820)	
◆	BuckCoeffXy (↗ see page 1820)	
◆	BuckCoeffXz (↗ see page 1820)	
◆	BuckCurveCoeffAlfy (↗ see page 1821)	
◆	BuckCurveCoeffAlfz (↗ see page 1821)	
◆	BuckCurveNumbY (↗ see page 1821)	
◆	BuckCurveNumbZ (↗ see page 1821)	
◆	BuckParamFiy (↗ see page 1822)	
◆	BuckParamFiz (↗ see page 1822)	
◆	BuckParamKy (↗ see page 1822)	
◆	BuckParamKz (↗ see page 1822)	
◆	BuckRelSlendLaby (↗ see page 1823)	
◆	BuckRelSlendLabz (↗ see page 1823)	
◆	BuckSLendLamy (↗ see page 1823)	
◆	BuckStrenNbryd (↗ see page 1823)	
◆	BuckStrenNbzrd (↗ see page 1824)	
◆	BucISlendLamz (↗ see page 1824)	

◆	ClassOfSect (☞ see page 1824)	
◆	ClassOfSectElem1 (☞ see page 1824)	
◆	ClassOfSectElem2 (☞ see page 1825)	
◆	ClassOfSectElem3 (☞ see page 1825)	
◆	ClassOfSectElem4 (☞ see page 1825)	
◆	CompParamBetaA (☞ see page 1825)	
◆	CritMomenMcr (☞ see page 1826)	
◆	EffectiveMomenMeff (☞ see page 1826)	
◆	EffSectAreaSeff (☞ see page 1826)	
◆	EffSectModulWyeff (☞ see page 1826)	
◆	EffSectModulWzeff (☞ see page 1827)	
◆	ElastMomStrenMelyrd (☞ see page 1827)	
◆	ElastMomStrenMelzrd (☞ see page 1827)	
◆	ElastSectModulWyel (☞ see page 1827)	
◆	ElastSectModulWzel (☞ see page 1828)	
◆	FlangeAreaAf (☞ see page 1828)	
◆	FlangeAreaAw (☞ see page 1828)	
◆	InteractParamAlfa (☞ see page 1828)	
◆	InteractParamBeta (☞ see page 1829)	
◆	InteractParamMilt (☞ see page 1829)	
◆	InteractParamMiy (☞ see page 1829)	
◆	InteractParamMiz (☞ see page 1829)	
◆	IsBuckY (☞ see page 1830)	
◆	IsBuckZ (☞ see page 1830)	
◆	LaodLevel (☞ see page 1830)	
◆	LatBuckCoeffXlt (☞ see page 1830)	
◆	LatBuckLengthLd (☞ see page 1831)	
◆	LatBuckMomStrenmbrd (☞ see page 1831)	
◆	LatBuckParamBetaW (☞ see page 1831)	
◆	LatBuckParamC1 (☞ see page 1831)	
◆	LatBuckParamC2 (☞ see page 1832)	

◆	LatBuckParamFilt (see page 1832)	
◆	LatBuckParamKit (see page 1832)	
◆	LatBuckParamLd (see page 1832)	
◆	LatBuckSlendLamlt (see page 1833)	
◆	LowFlanSlend1 (see page 1833)	
◆	LowFlanSlend2 (see page 1833)	
◆	MaEffRatio (see page 1833)	
◆	MaterCoeffGamma0 (see page 1834)	
◆	MaterCoeffGamma1 (see page 1834)	
◆	MaterCoeffGamma2 (see page 1834)	
◆	MaterialCapacityFy (see page 1834)	
◆	MaxBuckSlend (see page 1835)	
◆	MomStrengthMpzEuro (see page 1835)	
◆	OverallBuckStrenNbrd (see page 1835)	
◆	PartEffRatio1 (see page 1835)	
◆	PartEffRatio2 (see page 1836)	
◆	PartEffRatio3 (see page 1836)	
◆	PartEffRatio4 (see page 1836)	
◆	PartEffRatio5 (see page 1836)	
◆	PlastAxForcStrenNplrd (see page 1837)	
◆	PlastCompStrenNcrd (see page 1837)	
◆	PlastMomStrenMplyrd (see page 1837)	
◆	PlastMomStrenMplzrd (see page 1837)	
◆	PlastSectModulWypl (see page 1838)	
◆	PlastSectModulWzpl (see page 1838)	
◆	PlastTensStrenNtrd (see page 1838)	
◆	ReducMomStrenMcyrd (see page 1838)	
◆	ReducMomStrenMczrd (see page 1839)	
◆	ReducMomStrenMnyrd (see page 1839)	
◆	ReducMomStrenMnzrd (see page 1839)	
◆	ReducMomStrenMvyrd (see page 1839)	

◆	ReducMomStrenMvzrd (see page 1840)	
◆	ShearStrenTplyrd (see page 1840)	
◆	ShearStrenTplzrd (see page 1840)	
◆	StrsComp (see page 1840)	
◆	StrsLftEdgeMZ (see page 1841)	
◆	StrsLowEdgeMY (see page 1841)	
◆	StrsRgtEdgeMZ (see page 1841)	
◆	StrsShearY (see page 1841)	
◆	StrsShearZ (see page 1842)	
◆	StrsTens (see page 1842)	
◆	StrsUpEdgeMY (see page 1842)	
◆	TensStrenNurd (see page 1842)	
◆	TorsMomInertIlt (see page 1843)	
◆	UppFlanSlend (see page 1843)	
◆	WarpingConstantlw (see page 1843)	
◆	WebSlend (see page 1843)	
◆	WebSlend1 (see page 1844)	

## I.7.2 IRDimCodeResEC3 Fields

The fields of the IRDimCodeResEC3 class are listed here.

### Public Fields

Name	Description
AxForceExcentrEny (see page 1818)	
AxForcExcentrEnz (see page 1819)	
BendParamBetaMlty (see page 1819)	
BendParamBetaMltz (see page 1819)	
BendParamBetaMz (see page 1819)	
BendParamBetMy (see page 1820)	
BuckCoeffMinXi (see page 1820)	
BuckCoeffXy (see page 1820)	
BuckCoeffXz (see page 1820)	
BuckCurveCoeffAlfy (see page 1821)	
BuckCurveCoeffAlfz (see page 1821)	
BuckCurveNumbY (see page 1821)	
BuckCurveNumbZ (see page 1821)	
BuckParamFiy (see page 1822)	
BuckParamFiz (see page 1822)	

◆	BuckParamKy (see page 1822)	
◆	BuckParamKz (see page 1822)	
◆	BuckRelSlendLaby (see page 1823)	
◆	BuckRelSlendLabz (see page 1823)	
◆	BuckSLendLamy (see page 1823)	
◆	BuckStrenNbyrd (see page 1823)	
◆	BuckStrenNbzrd (see page 1824)	
◆	BuciSlendLamz (see page 1824)	
◆	ClassOfSect (see page 1824)	
◆	ClassOfSectElem1 (see page 1824)	
◆	ClassOfSectElem2 (see page 1825)	
◆	ClassOfSectElem3 (see page 1825)	
◆	ClassOfSectElem4 (see page 1825)	
◆	CompParamBetaA (see page 1825)	
◆	CritMomenMcr (see page 1826)	
◆	EffectiveMomenMeff (see page 1826)	
◆	EffSectAreaSeff (see page 1826)	
◆	EffSectModulWyeff (see page 1826)	
◆	EffSectModulWzeff (see page 1827)	
◆	ElastMomStrenMelyrd (see page 1827)	
◆	ElastMomStrenMelzrd (see page 1827)	
◆	ElastSectModulWyel (see page 1827)	
◆	ElastSectModulWzel (see page 1828)	
◆	FlangeAreaAf (see page 1828)	
◆	FlangeAreaAw (see page 1828)	
◆	InteractParamAlfa (see page 1828)	
◆	InteractParamBeta (see page 1829)	
◆	InteractParamMilt (see page 1829)	
◆	InteractParamMiy (see page 1829)	
◆	InteractParamMiz (see page 1829)	
◆	IsBuckY (see page 1830)	
◆	IsBuckZ (see page 1830)	
◆	LaodLevel (see page 1830)	
◆	LatBuckCoeffXlt (see page 1830)	

◆	LatBuckLengthLd (☞ see page 1831)	
◆	LatBuckMomStrenmbrd (☞ see page 1831)	
◆	LatBuckParamBetaW (☞ see page 1831)	
◆	LatBuckParamC1 (☞ see page 1831)	
◆	LatBuckParamC2 (☞ see page 1832)	
◆	LatBuckParamFilt (☞ see page 1832)	
◆	LatBuckParamKit (☞ see page 1832)	
◆	LatBuckParamLd (☞ see page 1832)	
◆	LatBuckSlendLamlt (☞ see page 1833)	
◆	LowFlanSlend1 (☞ see page 1833)	
◆	LowFlanSlend2 (☞ see page 1833)	
◆	MaEffRatio (☞ see page 1833)	
◆	MaterCoeffGamma0 (☞ see page 1834)	
◆	MaterCoeffGamma1 (☞ see page 1834)	
◆	MaterCoeffGamma2 (☞ see page 1834)	
◆	MaterialCapacityFy (☞ see page 1834)	
◆	MaxBuckSlend (☞ see page 1835)	
◆	MomStrengthMpzEuro (☞ see page 1835)	
◆	OverallBuckStrenNbrd (☞ see page 1835)	
◆	PartEffRatio1 (☞ see page 1835)	
◆	PartEffRatio2 (☞ see page 1836)	
◆	PartEffRatio3 (☞ see page 1836)	
◆	PartEffRatio4 (☞ see page 1836)	
◆	PartEffRatio5 (☞ see page 1836)	
◆	PlastAxForcStrenNplrd (☞ see page 1837)	
◆	PlastCompStrenNcrd (☞ see page 1837)	
◆	PlastMomStrenMplyrd (☞ see page 1837)	
◆	PlastMomStrenMplzrd (☞ see page 1837)	
◆	PlastSectModulWypl (☞ see page 1838)	
◆	PlastSectModulWzpl (☞ see page 1838)	
◆	PlastTensStrenNtrd (☞ see page 1838)	

◆ ReducMomStrenMcyrd (see page 1838)	
◆ ReducMomStrenMczrd (see page 1839)	
◆ ReducMomStrenMnyrd (see page 1839)	
◆ ReducMomStrenMnzrd (see page 1839)	
◆ ReducMomStrenMvyrd (see page 1839)	
◆ ReducMomStrenMvzrd (see page 1840)	
◆ ShearStrenTplyrd (see page 1840)	
◆ ShearStrenTpIzrd (see page 1840)	
◆ StrsComp (see page 1840)	
◆ StrsLftEdgeMZ (see page 1841)	
◆ StrsLowEdgeMY (see page 1841)	
◆ StrsRgtEdgeMZ (see page 1841)	
◆ StrsShearY (see page 1841)	
◆ StrsShearZ (see page 1842)	
◆ StrsTens (see page 1842)	
◆ StrsUprEdgeMY (see page 1842)	
◆ TensStrenNurd (see page 1842)	
◆ TorsMomInertIt (see page 1843)	
◆ UppFlanSlend (see page 1843)	
◆ WarpingConstantlw (see page 1843)	
◆ WebSlend (see page 1843)	
◆ WebSlend1 (see page 1844)	

### I.7.2.1 AxForceExcentrEny

**C++**

```
HRESULT get_AxForceExcentrEny(double*);
```

**C#**

```
public double AxForceExcentrEny { get; }
```

**Visual Basic**

```
Public ReadOnly AxForceExcentrEny As double
```

**Version**

Available since version 3.

### I.7.2.2 AxForcExcentrEnz

**C++**

```
HRESULT get_AxForcExcentrEnz(double*);
```

**C#**

```
public double AxForcExcentrEnz { get; }
```

**Visual Basic**

```
Public ReadOnly AxForcExcentrEnz As double
```

**Version**

Available since version 3.

**I.7.2.3 BendParamBetaMlty****C++**

```
HRESULT get_BendParamBetaMlty(double*);
```

**C#**

```
public double BendParamBetaMlty { get; }
```

**Visual Basic**

```
Public ReadOnly BendParamBetaMlty As double
```

**Version**

Available since version 3.

**I.7.2.4 BendParamBetaMltz****C++**

```
HRESULT get_BendParamBetaMltz(double*);
```

**C#**

```
public double BendParamBetaMltz { get; }
```

**Visual Basic**

```
Public ReadOnly BendParamBetaMltz As double
```

**Version**

Available since version 3.

**I.7.2.5 BendParamBetaMz****C++**

```
HRESULT get_BendParamBetaMz(double*);
```

**C#**

```
public double BendParamBetaMz { get; }
```

**Visual Basic**

```
Public ReadOnly BendParamBetaMz As double
```

**Version**

Available since version 3.

**I.7.2.6 BendParamBetMy****C++**

```
HRESULT get_BendParamBetMy(double*);
```

**C#**

```
public double BendParamBetMy { get; }
```

**Visual Basic**

```
Public ReadOnly BendParamBetMy As double
```

**Version**

Available since version 3.

**I.7.2.7 BuckCoeffMinXi****C++**

```
HRESULT get_BuckCoeffMinXi(double*);
```

**C#**

```
public double BuckCoeffMinXi { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCoeffMinXi As double
```

**Version**

Available since version 3.

**I.7.2.8 BuckCoeffXy****C++**

```
HRESULT get_BuckCoeffXy(double*);
```

**C#**

```
public double BuckCoeffXy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCoeffXy As double
```

**Version**

Available since version 3.

**I.7.2.9 BuckCoeffXz****C++**

```
HRESULT get_BuckCoeffXz(double*);
```

**C#**

```
public double BuckCoeffXz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCoeffXz As double
```

**Version**

Available since version 3.

**I.7.2.10 BuckCurveCoeffAlfy****C++**

```
HRESULT get_BuckCurveCoeffAlfy(double*);
```

**C#**

```
public double BuckCurveCoeffAlfy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCurveCoeffAlfy As double
```

**Version**

Available since version 3.

**I.7.2.11 BuckCurveCoeffAlfz****C++**

```
HRESULT get_BuckCurveCoeffAlfz(double*);
```

**C#**

```
public double BuckCurveCoeffAlfz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCurveCoeffAlfz As double
```

**Version**

Available since version 3.

**I.7.2.12 BuckCurveNumbY****C++**

```
HRESULT get_BuckCurveNumbY(double*);
```

**C#**

```
public double BuckCurveNumbY { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCurveNumbY As double
```

**Version**

Available since version 3.

**I.7.2.13 BuckCurveNumbZ****C++**

```
HRESULT get_BuckCurveNumbZ(double*);
```

**C#**

```
public double BuckCurveNumbZ { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCurveNumbZ As double
```

**Version**

Available since version 3.

**I.7.2.14 BuckParamFiy****C++**

```
HRESULT get_BuckParamFiy(double*);
```

**C#**

```
public double BuckParamFiy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckParamFiy As double
```

**Version**

Available since version 3.

**I.7.2.15 BuckParamFiz****C++**

```
HRESULT get_BuckParamFiz(double*);
```

**C#**

```
public double BuckParamFiz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckParamFiz As double
```

**Version**

Available since version 3.

**I.7.2.16 BuckParamKy****C++**

```
HRESULT get_BuckParamKy(double*);
```

**C#**

```
public double BuckParamKy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckParamKy As double
```

**Version**

Available since version 3.

**I.7.2.17 BuckParamKz****C++**

```
HRESULT get_BuckParamKz(double*);
```

**C#**

```
public double BuckParamKz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckParamKz As double
```

**Version**

Available since version 3.

**I.7.2.18 BuckRelSlendLaby****C++**

```
HRESULT get_BuckRelSlendLaby(double*);
```

**C#**

```
public double BuckRelSlendLaby { get; }
```

**Visual Basic**

```
Public ReadOnly BuckRelSlendLabz As double
```

**Version**

Available since version 3.

**I.7.2.19 BuckRelSlendLabz****C++**

```
HRESULT get_BuckRelSlendLabz(double*);
```

**C#**

```
public double BuckRelSlendLabz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckRelSlendLabz As double
```

**Version**

Available since version 3.

**I.7.2.20 BuckSLendLamy****C++**

```
HRESULT get_BuckSLendLamy(double*);
```

**C#**

```
public double BuckSLendLamy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckSLendLamy As double
```

**Version**

Available since version 3.

**I.7.2.21 BuckStrenNbyrd****C++**

```
HRESULT get_BuckStrenNbyrd(double*);
```

**C#**

```
public double BuckStrenNbyrd { get; }
```

**Visual Basic**

```
Public ReadOnly BuckStrenNbyrd As double
```

**Version**

Available since version 3.

**I.7.2.22 BuckStrenNbzrd****C++**

```
HRESULT get_BuckStrenNbzrd(double*);
```

**C#**

```
public double BuckStrenNbzrd { get; }
```

**Visual Basic**

```
Public ReadOnly BuckStrenNbzrd As double
```

**Version**

Available since version 3.

**I.7.2.23 BuclSlendLamz****C++**

```
HRESULT get_BuclSlendLamz(double*);
```

**C#**

```
public double BuclSlendLamz { get; }
```

**Visual Basic**

```
Public ReadOnly BuclSlendLamz As double
```

**Version**

Available since version 3.

**I.7.2.24 ClassOfSect****C++**

```
HRESULT get_ClassOfSect(long*);
```

**C#**

```
public long ClassOfSect { get; }
```

**Visual Basic**

```
Public ReadOnly ClassOfSect As long
```

**Version**

Available since version 3.

**I.7.2.25 ClassOfSectElem1****C++**

```
HRESULT get_ClassOfSectElem1(long*);
```

**C#**

```
public long ClassOfSectElem1 { get; }
```

**Visual Basic**

```
Public ReadOnly ClassOfSectElem1 As long
```

**Version**

Available since version 3.

**I.7.2.26 ClassOfSectElem2****C++**

```
HRESULT get_ClassOfSectElem2(long*);
```

**C#**

```
public long ClassOfSectElem2 { get; }
```

**Visual Basic**

```
Public ReadOnly ClassOfSectElem2 As long
```

**Version**

Available since version 3.

**I.7.2.27 ClassOfSectElem3****C++**

```
HRESULT get_ClassOfSectElem3(long*);
```

**C#**

```
public long ClassOfSectElem3 { get; }
```

**Visual Basic**

```
Public ReadOnly ClassOfSectElem3 As long
```

**Version**

Available since version 3.

**I.7.2.28 ClassOfSectElem4****C++**

```
HRESULT get_ClassOfSectElem4(long*);
```

**C#**

```
public long ClassOfSectElem4 { get; }
```

**Visual Basic**

```
Public ReadOnly ClassOfSectElem4 As long
```

**Version**

Available since version 3.

**I.7.2.29 CompParamBetaA****C++**

```
HRESULT get_CompParamBetaA(double*);
```

**C#**

```
public double CompParamBetaA { get; }
```

**Visual Basic**

```
Public ReadOnly CompParamBetaA As double
```

**Version**

Available since version 3.

**I.7.2.30 CritMomenMcr****C++**

```
HRESULT get_CritMomenMcr(double*);
```

**C#**

```
public double CritMomenMcr { get; }
```

**Visual Basic**

```
Public ReadOnly CritMomenMcr As double
```

**Version**

Available since version 3.

**I.7.2.31 EffectiveMomenMeff****C++**

```
HRESULT get_EffectiveMomenMeff(double*);
```

**C#**

```
public double EffectiveMomenMeff { get; }
```

**Visual Basic**

```
Public ReadOnly EffectiveMomenMeff As double
```

**Version**

Available since version 3.

**I.7.2.32 EffSectAreaSeff****C++**

```
HRESULT get_EffSectAreaSeff(double*);
```

**C#**

```
public double EffSectAreaSeff { get; }
```

**Visual Basic**

```
Public ReadOnly EffSectAreaSeff As double
```

**Version**

Available since version 3.

**I.7.2.33 EffSectModulWyeff****C++**

```
HRESULT get_EffSectModulWyeff(double*);
```

**C#**

```
public double EffSectModulWyeff { get; }
```

**Visual Basic**

```
Public ReadOnly EffSectModulWyeff As double
```

**Version**

Available since version 3.

**I.7.2.34 EffSectModulWzeff****C++**

```
HRESULT get_EffSectModulWzeff(double*);
```

**C#**

```
public double EffSectModulWzeff { get; }
```

**Visual Basic**

```
Public ReadOnly EffSectModulWzeff As double
```

**Version**

Available since version 3.

**I.7.2.35 ElastMomStrenMelyrd****C++**

```
HRESULT get_ElastMomStrenMelyrd(double*);
```

**C#**

```
public double ElastMomStrenMelyrd { get; }
```

**Visual Basic**

```
Public ReadOnly ElastMomStrenMelyrd As double
```

**Version**

Available since version 3.

**I.7.2.36 ElastMomStrenMelzrd****C++**

```
HRESULT get_ElastMomStrenMelzrd(double*);
```

**C#**

```
public double ElastMomStrenMelzrd { get; }
```

**Visual Basic**

```
Public ReadOnly ElastMomStrenMelzrd As double
```

**Version**

Available since version 3.

**I.7.2.37 ElastSectModulWyel****C++**

```
HRESULT get_ElastSectModulWyel(double*);
```

**C#**

```
public double ElastSectModulWyel { get; }
```

**Visual Basic**

```
Public ReadOnly ElastSectModulWyel As double
```

**Version**

Available since version 3.

**I.7.2.38 ElastSectModulWzel****C++**

```
HRESULT get_ElastSectModulWzel(double*);
```

**C#**

```
public double ElastSectModulWzel { get; }
```

**Visual Basic**

```
Public ReadOnly ElastSectModulWzel As double
```

**Version**

Available since version 3.

**I.7.2.39 FlangeAreaAf****C++**

```
HRESULT get_FlangeAreaAf(double*);
```

**C#**

```
public double FlangeAreaAf { get; }
```

**Visual Basic**

```
Public ReadOnly FlangeAreaAf As double
```

**Version**

Available since version 3.

**I.7.2.40 FlangeAreaAw****C++**

```
HRESULT get_FlangeAreaAw(double*);
```

**C#**

```
public double FlangeAreaAw { get; }
```

**Visual Basic**

```
Public ReadOnly FlangeAreaAw As double
```

**Version**

Available since version 3.

**I.7.2.41 InteractParamAlfa****C++**

```
HRESULT get_InteractParamAlfa(double*);
```

**C#**

```
public double InteractParamAlfa { get; }
```

**Visual Basic**

```
Public ReadOnly InteractParamAlfa As double
```

**Version**

Available since version 3.

**I.7.2.42 InteractParamBeta****C++**

```
HRESULT get_InteractParamBeta(double*);
```

**C#**

```
public double InteractParamBeta { get; }
```

**Visual Basic**

```
Public ReadOnly InteractParamBeta As double
```

**Version**

Available since version 3.

**I.7.2.43 InteractParamMilt****C++**

```
HRESULT get_InteractParamMilt(double*);
```

**C#**

```
public double InteractParamMilt { get; }
```

**Visual Basic**

```
Public ReadOnly InteractParamMilt As double
```

**Version**

Available since version 3.

**I.7.2.44 InteractParamMiy****C++**

```
HRESULT get_InteractParamMiy(double*);
```

**C#**

```
public double InteractParamMiy { get; }
```

**Visual Basic**

```
Public ReadOnly InteractParamMiy As double
```

**Version**

Available since version 3.

**I.7.2.45 InteractParamMiz****C++**

```
HRESULT get_InteractParamMiz(double*);
```

**C#**

```
public double InteractParamMiz { get; }
```

**Visual Basic**

```
Public ReadOnly InteractParamMiz As double
```

**Version**

Available since version 3.

**I.7.2.46 IsBuckY****C++**

```
HRESULT get_IsBuckY(long*);
```

**C#**

```
public long IsBuckY { get; }
```

**Visual Basic**

```
Public ReadOnly IsBuckY As long
```

**Version**

Available since version 3.

**I.7.2.47 IsBuckZ****C++**

```
HRESULT get_IsBuckZ(long*);
```

**C#**

```
public long IsBuckZ { get; }
```

**Visual Basic**

```
Public ReadOnly IsBuckZ As long
```

**Version**

Available since version 3.

**I.7.2.48 LaodLevel****C++**

```
HRESULT get_LaodLevel(double*);
```

**C#**

```
public double LaodLevel { get; }
```

**Visual Basic**

```
Public ReadOnly LaodLevel As double
```

**Version**

Available since version 3.

**I.7.2.49 LatBuckCoeffXlt****C++**

```
HRESULT get_LatBuckCoeffXlt(double*);
```

**C#**

```
public double LatBuckCoeffXlt { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckCoeffXlt As double
```

**Version**

Available since version 3.

**I.7.2.50 LatBuckLengthLd****C++**

```
HRESULT get_LatBuckLengthLd(double*);
```

**C#**

```
public double LatBuckLengthLd { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckLengthLd As double
```

**Version**

Available since version 3.

**I.7.2.51 LatBuckMomStrenmbrd****C++**

```
HRESULT get_LatBuckMomStrenmbrd(double*);
```

**C#**

```
public double LatBuckMomStrenmbrd { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckMomStrenmbrd As double
```

**Version**

Available since version 3.

**I.7.2.52 LatBuckParamBetaW****C++**

```
HRESULT get_LatBuckParamBetaW(double*);
```

**C#**

```
public double LatBuckParamBetaW { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamBetaW As double
```

**Version**

Available since version 3.

**I.7.2.53 LatBuckParamC1****C++**

```
HRESULT get_LatBuckParamC1(double*);
```

**C#**

```
public double LatBuckParamC1 { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamC1 As double
```

**Version**

Available since version 3.

**I.7.2.54 LatBuckParamC2****C++**

```
HRESULT get_LatBuckParamC2(double*);
```

**C#**

```
public double LatBuckParamC2 { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamC2 As double
```

**Version**

Available since version 3.

**I.7.2.55 LatBuckParamFilt****C++**

```
HRESULT get_LatBuckParamFilt(double*);
```

**C#**

```
public double LatBuckParamFilt { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamFilt As double
```

**Version**

Available since version 3.

**I.7.2.56 LatBuckParamKlt****C++**

```
HRESULT get_LatBuckParamKlt(double*);
```

**C#**

```
public double LatBuckParamKlt { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamKlt As double
```

**Version**

Available since version 3.

**I.7.2.57 LatBuckParamLd****C++**

```
HRESULT get_LatBuckParamLd(long*);
```

**C#**

```
public long LatBuckParamLd { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckParamLd As long
```

**Version**

Available since version 3.

**I.7.2.58 LatBuckSlendLamlt****C++**

```
HRESULT get_LatBuckSlendLamlt(double*);
```

**C#**

```
public double LatBuckSlendLamlt { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckSlendLamlt As double
```

**Version**

Available since version 3.

**I.7.2.59 LowFlanSlend1****C++**

```
HRESULT get_LowFlanSlend1(double*);
```

**C#**

```
public double LowFlanSlend1 { get; }
```

**Visual Basic**

```
Public ReadOnly LowFlanSlend1 As double
```

**Version**

Available since version 3.

**I.7.2.60 LowFlanSlend2****C++**

```
HRESULT get_LowFlanSlend2(double*);
```

**C#**

```
public double LowFlanSlend2 { get; }
```

**Visual Basic**

```
Public ReadOnly LowFlanSlend2 As double
```

**Version**

Available since version 3.

**I.7.2.61 MaEffRatio****C++**

```
HRESULT get_MaEffRatio(double*);
```

**C#**

```
public double MaEffRatio { get; }
```

**Visual Basic**

```
Public ReadOnly MaEffRatio As double
```

**Version**

Available since version 3.

**I.7.2.62 MaterCoeffGamma0****C++**

```
HRESULT get_MaterCoeffGamma0(double*);
```

**C#**

```
public double MaterCoeffGamma0 { get; }
```

**Visual Basic**

```
Public ReadOnly MaterCoeffGamma0 As double
```

**Version**

Available since version 3.

**I.7.2.63 MaterCoeffGamma1****C++**

```
HRESULT get_MaterCoeffGamma1(double*);
```

**C#**

```
public double MaterCoeffGamma1 { get; }
```

**Visual Basic**

```
Public ReadOnly MaterCoeffGamma1 As double
```

**Version**

Available since version 3.

**I.7.2.64 MaterCoeffGamma2****C++**

```
HRESULT get_MaterCoeffGamma2(double*);
```

**C#**

```
public double MaterCoeffGamma2 { get; }
```

**Visual Basic**

```
Public ReadOnly MaterCoeffGamma2 As double
```

**Version**

Available since version 3.

**I.7.2.65 MaterialCapacityFy****C++**

```
HRESULT get_MaterialCapacityFy(double*);
```

**C#**

```
public double MaterialCapacityFy { get; }
```

**Visual Basic**

```
Public ReadOnly MaterialCapacityFy As double
```

**Version**

Available since version 3.

**I.7.2.66 MaxBuckSlend****C++**

```
HRESULT get_MaxBuckSlend(double*);
```

**C#**

```
public double MaxBuckSlend { get; }
```

**Visual Basic**

```
Public ReadOnly MaxBuckSlend As double
```

**Version**

Available since version 3.

**I.7.2.67 MomStrengthMpzEuro****C++**

```
HRESULT get_MomStrengthMpzEuro(double*);
```

**C#**

```
public double MomStrengthMpzEuro { get; }
```

**Visual Basic**

```
Public ReadOnly MomStrengthMpzEuro As double
```

**Version**

Available since version 3.

**I.7.2.68 OverallBuckStrenNbrd****C++**

```
HRESULT get_OverallBuckStrenNbrd(double*);
```

**C#**

```
public double OverallBuckStrenNbrd { get; }
```

**Visual Basic**

```
Public ReadOnly OverallBuckStrenNbrd As double
```

**Version**

Available since version 3.

**I.7.2.69 PartEffRatio1****C++**

```
HRESULT get_PartEffRatio1(double*);
```

**C#**

```
public double PartEffRatio1 { get; }
```

**Visual Basic**

```
Public ReadOnly PartEffRatio1 As double
```

**Version**

Available since version 3.

**I.7.2.70 PartEffRatio2****C++**

```
HRESULT get_PartEffRatio2(double*);
```

**C#**

```
public double PartEffRatio2 { get; }
```

**Visual Basic**

```
Public ReadOnly PartEffRatio2 As double
```

**Version**

Available since version 3.

**I.7.2.71 PartEffRatio3****C++**

```
HRESULT get_PartEffRatio3(double*);
```

**C#**

```
public double PartEffRatio3 { get; }
```

**Visual Basic**

```
Public ReadOnly PartEffRatio3 As double
```

**Version**

Available since version 3.

**I.7.2.72 PartEffRatio4****C++**

```
HRESULT get_PartEffRatio4(double*);
```

**C#**

```
public double PartEffRatio4 { get; }
```

**Visual Basic**

```
Public ReadOnly PartEffRatio4 As double
```

**Version**

Available since version 3.

**I.7.2.73 PartEffRatio5****C++**

```
HRESULT get_PartEffRatio5(double*);
```

**C#**

```
public double PartEffRatio5 { get; }
```

**Visual Basic**

```
Public ReadOnly PartEffRatio5 As double
```

**Version**

Available since version 3.

**I.7.2.74 PlastAxForcStrenNplrd****C++**

```
HRESULT get_PlastAxForcStrenNplrd(double*);
```

**C#**

```
public double PlastAxForcStrenNplrd { get; }
```

**Visual Basic**

```
Public ReadOnly PlastAxForcStrenNplrd As double
```

**Version**

Available since version 3.

**I.7.2.75 PlastCompStrenNcrd****C++**

```
HRESULT get_PlastCompStrenNcrd(double*);
```

**C#**

```
public double PlastCompStrenNcrd { get; }
```

**Visual Basic**

```
Public ReadOnly PlastCompStrenNcrd As double
```

**Version**

Available since version 3.

**I.7.2.76 PlastMomStrenMplyrd****C++**

```
HRESULT get_PlastMomStrenMplyrd(double*);
```

**C#**

```
public double PlastMomStrenMplyrd { get; }
```

**Visual Basic**

```
Public ReadOnly PlastMomStrenMplyrd As double
```

**Version**

Available since version 3.

**I.7.2.77 PlastMomStrenMplzrd****C++**

```
HRESULT get_PlastMomStrenMplzrd(double*);
```

**C#**

```
public double PlastMomStrenMplzrd { get; }
```

**Visual Basic**

```
Public ReadOnly PlastMomStrenMplzrd As double
```

**Version**

Available since version 3.

**I.7.2.78 PlastSectModulWyp1****C++**

```
HRESULT get_PlastSectModulWyp1(double*);
```

**C#**

```
public double PlastSectModulWyp1 { get; }
```

**Visual Basic**

```
Public ReadOnly PlastSectModulWzpl As double
```

**Version**

Available since version 3.

**I.7.2.79 PlastSectModulWzpl****C++**

```
HRESULT get_PlastSectModulWzpl(double*);
```

**C#**

```
public double PlastSectModulWzpl { get; }
```

**Visual Basic**

```
Public ReadOnly PlastSectModulWzpl As double
```

**Version**

Available since version 3.

**I.7.2.80 PlastTensStrenNtrd****C++**

```
HRESULT get_PlastTensStrenNtrd(double*);
```

**C#**

```
public double PlastTensStrenNtrd { get; }
```

**Visual Basic**

```
Public ReadOnly PlastTensStrenNtrd As double
```

**Version**

Available since version 3.

**I.7.2.81 ReducMomStrenMcyrd****C++**

```
HRESULT get_ReducMomStrenMcyrd(double*);
```

**C#**

```
public double ReducMomStrenMcyrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMcyrd As double
```

**Version**

Available since version 3.

**I.7.2.82 ReducMomStrenMczrd****C++**

```
HRESULT get_ReducMomStrenMczrd(double*);
```

**C#**

```
public double ReducMomStrenMczrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMczrd As double
```

**Version**

Available since version 3.

**I.7.2.83 ReducMomStrenMnyrd****C++**

```
HRESULT get_ReducMomStrenMnyrd(double*);
```

**C#**

```
public double ReducMomStrenMnyrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMnyrd As double
```

**Version**

Available since version 3.

**I.7.2.84 ReducMomStrenMnzrd****C++**

```
HRESULT get_ReducMomStrenMnzrd(double*);
```

**C#**

```
public double ReducMomStrenMnzrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMnzrd As double
```

**Version**

Available since version 3.

**I.7.2.85 ReducMomStrenMvyrd****C++**

```
HRESULT get_ReducMomStrenMvyrd(double*);
```

**C#**

```
public double ReducMomStrenMvyrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMvyrd As double
```

**Version**

Available since version 3.

**I.7.2.86 ReducMomStrenMvzrd****C++**

```
HRESULT get_ReducMomStrenMvzrd(double*);
```

**C#**

```
public double ReducMomStrenMvzrd { get; }
```

**Visual Basic**

```
Public ReadOnly ReducMomStrenMvzrd As double
```

**Version**

Available since version 3.

**I.7.2.87 ShearStrenTplyrd****C++**

```
HRESULT get_ShearStrenTplyrd(double*);
```

**C#**

```
public double ShearStrenTplyrd { get; }
```

**Visual Basic**

```
Public ReadOnly ShearStrenTplyrd As double
```

**Version**

Available since version 3.

**I.7.2.88 ShearStrenTplzrd****C++**

```
HRESULT get_ShearStrenTplzrd(double*);
```

**C#**

```
public double ShearStrenTplzrd { get; }
```

**Visual Basic**

```
Public ReadOnly ShearStrenTplzrd As double
```

**Version**

Available since version 3.

**I.7.2.89 StrsComp****C++**

```
HRESULT get_StrsComp(double*);
```

**C#**

```
public double StrsComp { get; }
```

**Visual Basic**

```
Public ReadOnly StrsComp As double
```

**Version**

Available since version 3.

**I.7.2.90 StrsLftEdgeMZ****C++**

```
HRESULT get_StrsLftEdgeMZ(double*);
```

**C#**

```
public double StrsLftEdgeMZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsLftEdgeMZ As double
```

**Version**

Available since version 3.

**I.7.2.91 StrsLowEdgeMY****C++**

```
HRESULT get_StrsLowEdgeMY(double*);
```

**C#**

```
public double StrsLowEdgeMY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsLowEdgeMY As double
```

**Version**

Available since version 3.

**I.7.2.92 StrsRgtEdgeMZ****C++**

```
HRESULT get_StrsRgtEdgeMZ(double*);
```

**C#**

```
public double StrsRgtEdgeMZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsRgtEdgeMZ As double
```

**Version**

Available since version 3.

**I.7.2.93 StrsShearY****C++**

```
HRESULT get_StrsShearY(double*);
```

**C#**

```
public double StrsShearY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsShearY As double
```

**Version**

Available since version 3.

**I.7.2.94 StrsShearZ****C++**

```
HRESULT get_StrsShearZ(double*);
```

**C#**

```
public double StrsShearZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsShearZ As double
```

**Version**

Available since version 3.

**I.7.2.95 StrsTens****C++**

```
HRESULT get_StrsTens(double*);
```

**C#**

```
public double StrsTens { get; }
```

**Visual Basic**

```
Public ReadOnly StrsTens As double
```

**Version**

Available since version 3.

**I.7.2.96 StrsUprEdgeMY****C++**

```
HRESULT get_StrsUprEdgeMY(double*);
```

**C#**

```
public double StrsUprEdgeMY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsUprEdgeMY As double
```

**Version**

Available since version 3.

**I.7.2.97 TensStrenNurd****C++**

```
HRESULT get_TensStrenNurd(double*);
```

**C#**

```
public double TensStrenNurd { get; }
```

**Visual Basic**

```
Public ReadOnly TensStrenNurd As double
```

**Version**

Available since version 3.

**I.7.2.98 TorsMomInertIt****C++**

```
HRESULT get_TorsMomInertIt(double*);
```

**C#**

```
public double TorsMomInertIt { get; }
```

**Visual Basic**

```
Public ReadOnly TorsMomInertIt As double
```

**Version**

Available since version 3.

**I.7.2.99 UppFlanSlend****C++**

```
HRESULT get_UppFlanSlend(double*);
```

**C#**

```
public double UppFlanSlend { get; }
```

**Visual Basic**

```
Public ReadOnly UppFlanSlend As double
```

**Version**

Available since version 3.

**I.7.2.100 WarpingConstantIw****C++**

```
HRESULT get_WarpingConstantIw(double*);
```

**C#**

```
public double WarpingConstantIw { get; }
```

**Visual Basic**

```
Public ReadOnly WarpingConstantIw As double
```

**Version**

Available since version 3.

**I.7.2.101 WebSlend****C++**

```
HRESULT get_WebSlend(double*);
```

**C#**

```
public double WebSlend { get; }
```

**Visual Basic**

```
Public ReadOnly WebSlend As double
```

**Version**

Available since version 3.

**I.7.2.102 WebSlend1****C++**

```
HRESULT get_WebSlend1(double*);
```

**C#**

```
public double WebSlend1 { get; }
```

**Visual Basic**

```
Public ReadOnly WebSlend1 As double
```

**Version**

Available since version 3.

**I.8 IRDimYieldStrengthTypeEC3****C++**

```
enum IRDimYieldStrengthTypeEC3;
```

**C#**

```
public enum IRDimYieldStrengthTypeEC3;
```

**Visual Basic**

```
Public Enum IRDimYieldStrengthTypeEC3
```

**Members**

Members	Description
I_DYST_EC3_BASIC = 0	Available since version 3.5.
I_DYST_EC3_AVERAGE = 1	Available since version 3.5.

**Version**

Available since version 3.5.

**II CB71 Code**

Available since version 3.5.

**Enumerations**

	Name	Description
⊕	IRDimBuckDiagramCB71 (see page 1878)	
⊕	IRDimLaterBuckTypeCB71 (see page 1879)	
⊕	IRDimLoadLevelCB71 (see page 1879)	
⊕	IRDimLoadTypeCB71 (see page 1879)	
⊕	IRDimFireMemberPositionsCB71 (see page 1880)	
⊕	IRDimFireProtectionTimesCB71 (see page 1880)	

**Interfaces**

	Name	Description
⊕	IRDimMembParamsCB71 (see page 1845)	
⊕	IRDimCodeResCB71 (see page 1855)	

## II.1 IRDimMembParamsCB71

### Class Hierarchy

#### C++

```
interface IRDimMembParamsCB71 : IDispatch;
```

#### C#

```
public interface IRDimMembParamsCB71;
```

### Visual Basic

```
Public Interface IRDimMembParamsCB71
```

### Version

Available since version 3.5.

## II.1.1 IRDimMembParamsCB71 Members

The following tables list the members exposed by IRDimMembParamsCB71.

### Public Fields

	Name	Description
◆	ArcBeamCheck (see page 1847)	
◆	ArcRadius (see page 1848)	
◆	BuckComposBeamCoefCey (see page 1848)	
◆	BuckComposBeamCoefCez (see page 1848)	
◆	BuckIsComposedY (see page 1848)	
◆	BuckIsComposedZ (see page 1849)	
◆	BuckLengthCoeffZ (see page 1849)	
◆	BucklingDiagramY (see page 1849)	
◆	BucklingDiagramZ (see page 1849)	
◆	BuckLengthCoeffY (see page 1850)	
◆	FireContinProtForComposBeam (see page 1850)	
◆	FireLftSideProt (see page 1850)	
◆	FireLowSideProt (see page 1850)	
◆	FireMembPosition (see page 1851)	
◆	FireProtectionTime (see page 1851)	
◆	FireRequiredResist (see page 1851)	
◆	FireRgtSideProt (see page 1851)	
◆	FireUpSideProt (see page 1852)	
◆	ForceDistFromBeginDistX (see page 1852)	

◆	HumChangeDeltaH (see page 1852)	
◆	InflDeflCoefTheta (see page 1853)	
◆	LatBuckType (see page 1853)	
◆	LatCoeffLowerFlangeValue (see page 1853)	
◆	LatCoeffUpperFlangeValue (see page 1853)	
◆	LoadLevel (see page 1854)	
◆	LoadTypeY (see page 1854)	
◆	TensAreaNetGros (see page 1854)	
◆	TubeControl (see page 1854)	

## II.1.2 IRDimMembParamsCB71 Fields

The fields of the IRDimMembParamsCB71 class are listed here.

### Public Fields

	Name	Description
◆	ArcBeamCheck (see page 1847)	
◆	ArcRadius (see page 1848)	
◆	BuckComposBeamCoefCey (see page 1848)	
◆	BuckComposBeamCoefCez (see page 1848)	
◆	BuckIsComposedY (see page 1848)	
◆	BuckIsComposedZ (see page 1849)	
◆	BuckLengthCoeffZ (see page 1849)	
◆	BucklingDiagramY (see page 1849)	
◆	BucklingDiagramZ (see page 1849)	
◆	BuckLengthCoeffY (see page 1850)	
◆	FireContinProtForComposBeam (see page 1850)	
◆	FireLftSideProt (see page 1850)	
◆	FireLowSideProt (see page 1850)	
◆	FireMembPosition (see page 1851)	
◆	FireProtectionTime (see page 1851)	
◆	FireRequiredResist (see page 1851)	
◆	FireRgtSideProt (see page 1851)	
◆	FireUprSideProt (see page 1852)	
◆	ForceDistFromBeginDistX (see page 1852)	
◆	HumChangeDeltaH (see page 1852)	

◆	InflDeflCoefTheta (see page 1853)	
◆	LatBuckType (see page 1853)	
◆	LatCoeffLowerFlangeValue (see page 1853)	
◆	LatCoeffUpperFlangeValue (see page 1853)	
◆	LoadLevel (see page 1854)	
◆	LoadTypeY (see page 1854)	
◆	TensAreaNetGros (see page 1854)	
◆	TubeControl (see page 1854)	

### II.1.2.1 ArcBeamCheck

**C++**

```
HRESULT get_ArcBeamCheck(VARIANT_BOOL* );
HRESULT put_ArcBeamCheck(VARIANT_BOOL);
```

**C#**

```
public bool ArcBeamCheck { get; set; }
```

**Visual Basic**

```
Public ArcBeamCheck As Boolean
```

**Version**

Available since version 3.5.

### II.1.2.2 ArcRadius

**C++**

```
HRESULT get_ArcRadius(double* );
HRESULT put_ArcRadius(double);
```

**C#**

```
public double ArcRadius { get; set; }
```

**Visual Basic**

```
Public ArcRadius As Double
```

**Version**

Available since version 3.5.

### II.1.2.3 BuckComposBeamCoefCey

**C++**

```
HRESULT get_BuckComposBeamCoefCey(double* );
HRESULT put_BuckComposBeamCoefCey(double);
```

**C#**

```
public double BuckComposBeamCoefCey { get; set; }
```

**Visual Basic**

```
Public BuckComposBeamCoefCey As Double
```

**Version**

Available since version 3.5.

#### II.1.2.4 BuckComposBeamCoefCez

**C++**

```
HRESULT get_BuckComposBeamCoefCez(double*);  
HRESULT put_BuckComposBeamCoefCez(double);
```

**C#**

```
public double BuckComposBeamCoefCez { get; set; }
```

**Visual Basic**

```
Public BuckComposBeamCoefCez As Double
```

**Version**

Available since version 3.5.

#### II.1.2.5 BuckIsComposedY

**C++**

```
HRESULT get_BuckIsComposedY(VARIANT_BOOL*);  
HRESULT put_BuckIsComposedY(VARIANT_BOOL);
```

**C#**

```
public bool BuckIsComposedY { get; set; }
```

**Visual Basic**

```
Public BuckIsComposedY As Boolean
```

**Version**

Available since version 3.5.

#### II.1.2.6 BuckIsComposedZ

**C++**

```
HRESULT get_BuckIsComposedZ(VARIANT_BOOL*);  
HRESULT put_BuckIsComposedZ(VARIANT_BOOL);
```

**C#**

```
public bool BuckIsComposedZ { get; set; }
```

**Visual Basic**

```
Public BuckIsComposedZ As Boolean
```

**Version**

Available since version 3.5.

#### II.1.2.7 BuckLengthCoeffZ

**C++**

```
HRESULT get_BuckLengthCoeffZ(double*);  
HRESULT put_BuckLengthCoeffZ(double);
```

**C#**

```
public double BuckLengthCoeffZ { get; set; }
```

**Visual Basic**

```
Public BuckLengthCoeffZ As Double
```

**Version**

Available since version 3.5.

**II.1.2.8 BucklingDiagramY****C++**

```
HRESULT get_BucklingDiagramY(IRDimBuckDiagramCB71* );
HRESULT put_BucklingDiagramY(IRDimBuckDiagramCB71);
```

**C#**

```
public IRDimBuckDiagramCB71 BucklingDiagramY { get; set; }
```

**Visual Basic**

```
Public BucklingDiagramY As IRDimBuckDiagramCB71
```

**Version**

Available since version 3.5.

**II.1.2.9 BucklingDiagramZ****C++**

```
HRESULT get_BucklingDiagramZ(IRDimBuckDiagramCB71* );
HRESULT put_BucklingDiagramZ(IRDimBuckDiagramCB71);
```

**C#**

```
public IRDimBuckDiagramCB71 BucklingDiagramZ { get; set; }
```

**Visual Basic**

```
Public BucklingDiagramZ As IRDimBuckDiagramCB71
```

**Version**

Available since version 3.5.

**II.1.2.10 BucklLengthCoeffY****C++**

```
HRESULT get_BucklLengthCoeffY(double* );
HRESULT put_BucklLengthCoeffY(double);
```

**C#**

```
public double BucklLengthCoeffY { get; set; }
```

**Visual Basic**

```
Public BucklLengthCoeffY As Double
```

**Version**

Available since version 3.5.

**II.1.2.11 FireContinProtForComposBeam****C++**

```
HRESULT get_FireContinProtForComposBeam(VARIANT_BOOL* );
HRESULT put_FireContinProtForComposBeam(VARIANT_BOOL);
```

**C#**

```
public bool FireContinProtForComposBeam { get; set; }
```

**Visual Basic**

```
Public FireContinProtForComposBeam As Boolean
```

**Version**

Available since version 3.5.

**II.1.2.12 FireLftSideProt****C++**

```
HRESULT get_FireLftSideProt(VARIANT_BOOL* );
HRESULT put_FireLftSideProt(VARIANT_BOOL);
```

**C#**

```
public bool FireLftSideProt { get; set; }
```

**Visual Basic**

```
Public FireLftSideProt As Boolean
```

**Version**

Available since version 3.5.

**II.1.2.13 FireLowSideProt****C++**

```
HRESULT get_FireLowSideProt(VARIANT_BOOL* );
HRESULT put_FireLowSideProt(VARIANT_BOOL);
```

**C#**

```
public bool FireLowSideProt { get; set; }
```

**Visual Basic**

```
Public FireLowSideProt As Boolean
```

**Version**

Available since version 3.5.

**II.1.2.14 FireMembPosition****C++**

```
HRESULT get_FireMembPosition(IRDIMFireMemberPositionsCB71* );
HRESULT put_FireMembPosition(IRDIMFireMemberPositionsCB71);
```

**C#**

```
public IRDIMFireMemberPositionsCB71 FireMembPosition { get; set; }
```

**Visual Basic**

```
Public FireMembPosition As IRDIMFireMemberPositionsCB71
```

**Version**

Available since version 3.5.

**II.1.2.15 FireProtectionTime****C++**

```
HRESULT get_FireProtectionTime(IRDIMFireProtectionTimesCB71* );
HRESULT put_FireProtectionTime(IRDIMFireProtectionTimesCB71);
```

**C#**

```
public IRDimFireProtectionTimesCB71 FireProtectionTime { get; set; }
```

**Visual Basic**

```
Public FireProtectionTime As IRDimFireProtectionTimesCB71
```

**Version**

Available since version 3.5.

**II.1.2.16 FireRequiredResist****C++**

```
HRESULT get_FireRequiredResist(double* );
HRESULT put_FireRequiredResist(double);
```

**C#**

```
public double FireRequiredResist { get; set; }
```

**Visual Basic**

```
Public FireRequiredResist As Double
```

**Version**

Available since version 3.5.

**II.1.2.17 FireRgtSideProt****C++**

```
HRESULT get_FireRgtSideProt(VARIANT_BOOL* );
HRESULT put_FireRgtSideProt(VARIANT_BOOL);
```

**C#**

```
public bool FireRgtSideProt { get; set; }
```

**Visual Basic**

```
Public FireRgtSideProt As Boolean
```

**Version**

Available since version 3.5.

**II.1.2.18 FireUpSideProt****C++**

```
HRESULT get_FireUpSideProt(VARIANT_BOOL* );
HRESULT put_FireUpSideProt(VARIANT_BOOL);
```

**C#**

```
public bool FireUpSideProt { get; set; }
```

**Visual Basic**

```
Public FireUpSideProt As Boolean
```

**Version**

Available since version 3.5.

**II.1.2.19 ForceDistFromBeginDistX****C++**

```
HRESULT get_ForceDistFromBeginDistX(double* );
```

```
HRESULT put_ForceDistFromBeginDistX(double);  
  
C#  
public double ForceDistFromBeginDistX { get; set; }
```

**Visual Basic**

```
Public ForceDistFromBeginDistX As double
```

**Version**

Available since version 3.5.

**II.1.2.20 HumChangeDeltaH****C++**

```
HRESULT get_HumChangeDeltaH(double*);  
HRESULT put_HumChangeDeltaH(double);
```

**C#**

```
public double HumChangeDeltaH { get; set; }
```

**Visual Basic**

```
Public HumChangeDeltaH As double
```

**Version**

Available since version 3.5.

**II.1.2.21 InflDeflCoefTheta****C++**

```
HRESULT get_InflDeflCoefTheta(double*);  
HRESULT put_InflDeflCoefTheta(double);
```

**C#**

```
public double InflDeflCoefTheta { get; set; }
```

**Visual Basic**

```
Public InflDeflCoefTheta As double
```

**Version**

Available since version 3.5.

**II.1.2.22 LatBuckType****C++**

```
HRESULT get_LatBuckType(IRDimLaterBuckTypeCB71*);  
HRESULT put_LatBuckType(IRDimLaterBuckTypeCB71);
```

**C#**

```
public IRDimLaterBuckTypeCB71 LatBuckType { get; set; }
```

**Visual Basic**

```
Public LatBuckType As IRDimLaterBuckTypeCB71
```

**Version**

Available since version 3.5.

### II.1.2.23 LatCoeffLowerFlangeValue

**C++**

```
HRESULT get_LatCoeffLowerFlangeValue(double* );
HRESULT put_LatCoeffLowerFlangeValue(double);
```

**C#**

```
public double LatCoeffLowerFlangeValue { get; set; }
```

**Visual Basic**

```
Public LatCoeffLowerFlangeValue As Double
```

**Version**

Available since version 3.5.

### II.1.2.24 LatCoeffUpperFlangeValue

**C++**

```
HRESULT get_LatCoeffUpperFlangeValue(double* );
HRESULT put_LatCoeffUpperFlangeValue(double);
```

**C#**

```
public double LatCoeffUpperFlangeValue { get; set; }
```

**Visual Basic**

```
Public LatCoeffUpperFlangeValue As Double
```

**Version**

Available since version 3.5.

### II.1.2.25 LoadLevel

**C++**

```
HRESULT get_LoadLevel(IRDimLoadLevelCB71* );
HRESULT put_LoadLevel(IRDimLoadLevelCB71);
```

**C#**

```
public IRDimLoadLevelCB71 LoadLevel { get; set; }
```

**Visual Basic**

```
Public LoadLevel As IRDimLoadLevelCB71
```

**Version**

Available since version 3.5.

### II.1.2.26 LoadTypeY

**C++**

```
HRESULT get_LoadTypeY(IRDimLoadTypeCB71* );
HRESULT put_LoadTypeY(IRDimLoadTypeCB71);
```

**C#**

```
public IRDimLoadTypeCB71 LoadTypeY { get; set; }
```

**Visual Basic**

```
Public LoadTypeY As IRDimLoadTypeCB71
```

**Version**

Available since version 3.5.

**II.1.2.27 TensAreaNetGros****C++**

```
HRESULT get_TensAreaNetGros(double*);  
HRESULT put_TensAreaNetGros(double);
```

**C#**

```
public double TensAreaNetGros { get; set; }
```

**Visual Basic**

```
Public TensAreaNetGros As Double
```

**Version**

Available since version 3.5.

**II.1.2.28 TubeControl****C++**

```
HRESULT get_TubeControl(VARIANT_BOOL*);  
HRESULT put_TubeControl(VARIANT_BOOL);
```

**C#**

```
public bool TubeControl { get; set; }
```

**Visual Basic**

```
Public TubeControl As Boolean
```

**Version**

Available since version 3.5.

**II.2 IRDimCodeResCB71****Class Hierarchy****C++**

```
interface IRDimCodeResCB71 : IDispatch;
```

**C#**

```
public interface IRDimCodeResCB71;
```

**Visual Basic**

```
Public Interface IRDimCodeResCB71
```

**Version**

Available since version 3.5.

**II.2.1 IRDimCodeResCB71 Members**

The following tables list the members exposed by IRDimCodeResCB71.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	BadWidthBfUnderFire ( <a href="#">see page 1860</a> )	
◆	BuckCoefKMax ( <a href="#">see page 1861</a> )	
◆	BuckCoefKy ( <a href="#">see page 1861</a> )	
◆	BuckCoefKz ( <a href="#">see page 1861</a> )	
◆	BuckLengthComposBeamLey ( <a href="#">see page 1861</a> )	
◆	BuckLengthComposBeamLez ( <a href="#">see page 1862</a> )	
◆	BuckLengthLfy ( <a href="#">see page 1862</a> )	
◆	BuckLengthLfz ( <a href="#">see page 1862</a> )	
◆	BuckSlendComposBeamLaym ( <a href="#">see page 1862</a> )	
◆	BuckSlendComposBeamLazm ( <a href="#">see page 1863</a> )	
◆	BuckSLendLay ( <a href="#">see page 1863</a> )	
◆	BuckSlendLaz ( <a href="#">see page 1863</a> )	
◆	CoefDependOnAngle ( <a href="#">see page 1863</a> )	
◆	CoefDependOnHeight ( <a href="#">see page 1864</a> )	Available since version 3.5.
◆	CoefDependOnHumAndBend ( <a href="#">see page 1864</a> )	
◆	CoefDependOnHumAndComp ( <a href="#">see page 1864</a> )	
◆	CoefDependOnWidth ( <a href="#">see page 1864</a> )	
◆	EfficiencyRatio ( <a href="#">see page 1864</a> )	
◆	FireHeightInMidSpanH ( <a href="#">see page 1865</a> )	
◆	FireInerMomInMidSpanly ( <a href="#">see page 1865</a> )	
◆	FireInerMomInMidSpanlz ( <a href="#">see page 1865</a> )	
◆	FireInerMomIx ( <a href="#">see page 1865</a> )	
◆	FireInerMomLy ( <a href="#">see page 1866</a> )	
◆	FireInerMomIz ( <a href="#">see page 1866</a> )	
◆	FireLftEdgeDistVpy ( <a href="#">see page 1866</a> )	
◆	FireLowEdgeDistVpz ( <a href="#">see page 1866</a> )	
◆	FireProtectCoefK2 ( <a href="#">see page 1867</a> )	
◆	FireProtection ( <a href="#">see page 1867</a> )	
◆	FireProtectionPosition ( <a href="#">see page 1867</a> )	
◆	FireRedCoefK1b ( <a href="#">see page 1867</a> )	
◆	FireRedCoefK1h ( <a href="#">see page 1868</a> )	
◆	FireResistance ( <a href="#">see page 1868</a> )	
◆	FireRgtEdgeDistVy ( <a href="#">see page 1868</a> )	

◆	FireSecAreaInMidSpanS (see page 1868)	
◆	FireSecAreaS (see page 1869)	
◆	FireSeModulWy (see page 1869)	
◆	FireSeModulWZ (see page 1869)	
◆	FireShearAreaSY (see page 1869)	
◆	FireShearAreaSZ (see page 1870)	
◆	FireUpEdgeDistVz (see page 1870)	
◆	FireWidthAtMembEndB (see page 1870)	
◆	FireWidthAtMembEndH (see page 1870)	
◆	FireWidthB (see page 1871)	
◆	FireWidthH (see page 1871)	
◆	FireWidthInMidSpanB (see page 1871)	
◆	IsBuckY (see page 1871)	
◆	IsBuckZ (see page 1872)	
◆	LatBuckCoefKinst (see page 1872)	
◆	LatBuckLengthCoef (see page 1872)	
◆	LatBuckLengthLd (see page 1872)	
◆	LatBuckRelSlendLam (see page 1873)	
◆	LatBuckStrsCrit (see page 1873)	
◆	MatAxCompResist (see page 1873)	
◆	MatAxTensResist (see page 1873)	
◆	MatBendResist (see page 1874)	
◆	MatShearResist (see page 1874)	
◆	MatTrCompResist (see page 1874)	
◆	MatTrTensResist (see page 1874)	
◆	StrsBend (see page 1875)	
◆	StrsBendInCompEdgeMY (see page 1875)	
◆	StrsBendInCompEdgeMZ (see page 1875)	
◆	StrsBendInCurveBeam (see page 1875)	
◆	StrsComp (see page 1876)	
◆	StrsFinal (see page 1876)	
◆	StrsLftEdgeMZ (see page 1876)	
◆	StrsLowEdgeMY (see page 1876)	
◆	StrsRgtEdgeMZ (see page 1877)	
◆	StrsShearY (see page 1877)	
◆	StrsShearZ (see page 1877)	

◆	StrsTens (see page 1877)	
◆	StrsUpEdgeMY (see page 1878)	
◆	TimberType (see page 1878)	

## II.2.2 IRDimCodeResCB71 Fields

The fields of the IRDimCodeResCB71 class are listed here.

### Public Fields

	Name	Description
◆	BadWidthBfUnderFire (see page 1860)	
◆	BuckCoefKMax (see page 1861)	
◆	BuckCoefKy (see page 1861)	
◆	BuckCoefKz (see page 1861)	
◆	BuckLengthComposBeamLey (see page 1861)	
◆	BuckLengthComposBeamLez (see page 1862)	
◆	BuckLengthLfy (see page 1862)	
◆	BuckLengthLfz (see page 1862)	
◆	BuckSlendComposBeamLaym (see page 1862)	
◆	BuckSlendComposBeamLazm (see page 1863)	
◆	BuckSLendLay (see page 1863)	
◆	BuckSlendLaz (see page 1863)	
◆	CoefDependOnAngle (see page 1863)	
◆	CoefDependOnHeight (see page 1864)	Available since version 3.5.
◆	CoefDependOnHumAndBend (see page 1864)	
◆	CoefDependOnHumAndComp (see page 1864)	
◆	CoefDependOnWidth (see page 1864)	
◆	EfficiencyRatio (see page 1864)	
◆	FireHightInMidSpanH (see page 1865)	
◆	FireInerMomInMidSpanly (see page 1865)	
◆	FireInerMomInMidSpanlz (see page 1865)	
◆	FireInerMomIx (see page 1865)	
◆	FireInerMomLy (see page 1866)	
◆	FireInerMomIz (see page 1866)	
◆	FireLftEdgeDistVpy (see page 1866)	
◆	FireLowEdgeDistVpz (see page 1866)	
◆	FireProtectCoefK2 (see page 1867)	
◆	FireProtection (see page 1867)	

◆	FireProtectionPosition (☞ see page 1867)	
◆	FireRedCoefK1b (☞ see page 1867)	
◆	FireRedCoefK1h (☞ see page 1868)	
◆	FireResistance (☞ see page 1868)	
◆	FireRgtEdgeDistVy (☞ see page 1868)	
◆	FireSecAreaInMidSpanS (☞ see page 1868)	
◆	FireSecAreaS (☞ see page 1869)	
◆	FireSeModulWy (☞ see page 1869)	
◆	FireSeModulWZ (☞ see page 1869)	
◆	FireShearAreaSY (☞ see page 1869)	
◆	FireShearAreaSZ (☞ see page 1870)	
◆	FireUpEdgeDistVz (☞ see page 1870)	
◆	FireWidthAtMembEndB (☞ see page 1870)	
◆	FireWidthAtMembEndH (☞ see page 1870)	
◆	FireWidthB (☞ see page 1871)	
◆	FireWidthH (☞ see page 1871)	
◆	FireWidthInMidSpanB (☞ see page 1871)	
◆	IsBuckY (☞ see page 1871)	
◆	IsBuckZ (☞ see page 1872)	
◆	LatBuckCoefKinst (☞ see page 1872)	
◆	LatBuckLengthCoef (☞ see page 1872)	
◆	LatBuckLengthLd (☞ see page 1872)	
◆	LatBuckRelSlendLam (☞ see page 1873)	
◆	LatBuckStrsCrit (☞ see page 1873)	
◆	MatAxCompResist (☞ see page 1873)	
◆	MatAxTensResist (☞ see page 1873)	
◆	MatBendResist (☞ see page 1874)	
◆	MatShearResist (☞ see page 1874)	
◆	MatTrCompResist (☞ see page 1874)	
◆	MatTrTensResist (☞ see page 1874)	
◆	StrsBend (☞ see page 1875)	
◆	StrsBendInCompEdgeMY (☞ see page 1875)	
◆	StrsBendInCompEdgeMZ (☞ see page 1875)	
◆	StrsBendInCurveBeam (☞ see page 1875)	
◆	StrsComp (☞ see page 1876)	

◆	StrsFinal ( [ see page 1876 )	
◆	StrsLftEdgeMZ ( [ see page 1876 )	
◆	StrsLowEdgeMY ( [ see page 1876 )	
◆	StrsRgtEdgeMZ ( [ see page 1877 )	
◆	StrsShearY ( [ see page 1877 )	
◆	StrsShearZ ( [ see page 1877 )	
◆	StrsTens ( [ see page 1877 )	
◆	StrsUprEdgeMY ( [ see page 1878 )	
◆	TimberType ( [ see page 1878 )	

### II.2.2.1 BadWidthBfUnderFire

#### C++

```
HRESULT get_BadWidthBfUnderFire(VARIANT_BOOL* );
```

#### C#

```
public bool BadWidthBfUnderFire { get; }
```

#### Visual Basic

```
Public ReadOnly BuckCoefKMax As Boolean
```

#### Version

Available since version 3.5.

### II.2.2.2 BuckCoefKMax

#### C++

```
HRESULT get_BuckCoefKMax(double* );
```

#### C#

```
public double BuckCoefKMax { get; }
```

#### Visual Basic

```
Public ReadOnly BuckCoefKMax As Double
```

#### Version

Available since version 3.5.

### II.2.2.3 BuckCoefKy

#### C++

```
HRESULT get_BuckCoefKy(double* );
```

#### C#

```
public double BuckCoefKy { get; }
```

#### Visual Basic

```
Public ReadOnly BuckCoefKy As Double
```

#### Version

Available since version 3.5.

### II.2.2.4 BuckCoefKz

#### C++

```
HRESULT get_BuckCoefKz(double* );
```

**C#**

```
public double BuckCoefKz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckCoefKz As Double
```

**Version**

Available since version 3.5.

**II.2.2.5 BuckLengthComposBeamLey****C++**

```
HRESULT get_BuckLengthComposBeamLey(double*);
```

**C#**

```
public double BuckLengthComposBeamLey { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthComposBeamLey As Double
```

**Version**

Available since version 3.5.

**II.2.2.6 BuckLengthComposBeamLez****C++**

```
HRESULT get_BuckLengthComposBeamLez(double*);
```

**C#**

```
public double BuckLengthComposBeamLez { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthComposBeamLez As Double
```

**Version**

Available since version 3.5.

**II.2.2.7 BuckLengthLfy****C++**

```
HRESULT get_BuckLengthLfy(double*);
```

**C#**

```
public double BuckLengthLfy { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthLfy As Double
```

**Version**

Available since version 3.5.

**II.2.2.8 BuckLengthLfz****C++**

```
HRESULT get_BuckLengthLfz(double*);
```

**C#**

```
public double BuckLengthLfz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckLengthLfz As Double
```

**Version**

Available since version 3.5.

**II.2.2.9 BuckSlendComposBeamLaym****C++**

```
HRESULT get_BuckSlendComposBeamLaym(double*);
```

**C#**

```
public double BuckSlendComposBeamLaym { get; }
```

**Visual Basic**

```
Public ReadOnly BuckSlendComposBeamLaym As Double
```

**Version**

Available since version 3.5.

**II.2.2.10 BuckSlendComposBeamLazm****C++**

```
HRESULT get_BuckSlendComposBeamLazm(double*);
```

**C#**

```
public double BuckSlendComposBeamLazm { get; }
```

**Visual Basic**

```
Public ReadOnly BuckSlendComposBeamLazm As Double
```

**Version**

Available since version 3.5.

**II.2.2.11 BuckSLendLay****C++**

```
HRESULT get_BuckSLendLay(double*);
```

**C#**

```
public double BuckSLendLay { get; }
```

**Visual Basic**

```
Public ReadOnly BuckSLendLay As Double
```

**Version**

Available since version 3.5.

**II.2.2.12 BuckSlendLaz****C++**

```
HRESULT get_BuckSlendLaz(double*);
```

**C#**

```
public double BuckSlendLaz { get; }
```

**Visual Basic**

```
Public ReadOnly BuckSlendLaz As Double
```

**Version**

Available since version 3.5.

### II.2.2.13 CoefDependOnAngle

**C++**

```
HRESULT get_CoefDependOnAngle(double*);
```

**C#**

```
public double CoefDependOnAngle { get; }
```

**Visual Basic**

```
Public ReadOnly CoefDependOnAngle As Double
```

**Version**

Available since version 3.5.

### II.2.2.14 CoefDependOnHeight

**C#**

```
void CoefDependOnHeight;
```

**Description**

Available since version 3.5.

### II.2.2.15 CoefDependOnHumAndBend

**C++**

```
HRESULT get_CoefDependOnHumAndBend(double*);
```

**C#**

```
public double CoefDependOnHumAndBend { get; }
```

**Visual Basic**

```
Public ReadOnly CoefDependOnHumAndBend As Double
```

**Version**

Available since version 3.5.

### II.2.2.16 CoefDependOnHumAndComp

**C++**

```
HRESULT get_CoefDependOnHumAndComp(double*);
```

**C#**

```
public double CoefDependOnHumAndComp { get; }
```

**Visual Basic**

```
Public ReadOnly CoefDependOnHumAndComp As Double
```

**Version**

Available since version 3.5.

**II.2.2.17 CoefDependOnWidth****C++**

```
HRESULT get_CoefDependOnWidth(double*);
```

**C#**

```
public double CoefDependOnWidth { get; }
```

**Visual Basic**

```
Public ReadOnly CoefDependOnWidth As double
```

**Version**

Available since version 3.5.

**II.2.2.18 EfficiencyRatio****C++**

```
HRESULT get_EfficiencyRatio(double*);
```

**C#**

```
public double EfficiencyRatio { get; }
```

**Visual Basic**

```
Public ReadOnly EfficiencyRatio As double
```

**Version**

Available since version 3.5.

**II.2.2.19 FireHightInMidSpanH****C++**

```
HRESULT get_FireHightInMidSpanH(double*);
```

**C#**

```
public double FireHightInMidSpanH { get; }
```

**Visual Basic**

```
Public ReadOnly FireHightInMidSpanH As double
```

**Version**

Available since version 3.5.

**II.2.2.20 FireInerMomInMidSpanIy****C++**

```
HRESULT get_FireInerMomInMidSpanIy(double*);
```

**C#**

```
public double FireInerMomInMidSpanIy { get; }
```

**Visual Basic**

```
Public ReadOnly FireInerMomInMidSpanIy As double
```

**Version**

Available since version 3.5.

**II.2.2.21 FireInerMomInMidSpanIz****C++**

```
HRESULT get_FireInerMomInMidSpanIz(double*);
```

**C#**

```
public double FireInerMomInMidSpanIz { get; }
```

**Visual Basic**

```
Public ReadOnly FireInerMomInMidSpanIz As double
```

**Version**

Available since version 3.5.

**II.2.2.22 FireInerMomIx****C++**

```
HRESULT get_FireInerMomIx(double*);
```

**C#**

```
public double FireInerMomIx { get; }
```

**Visual Basic**

```
Public ReadOnly FireInerMomIx As double
```

**Version**

Available since version 3.5.

**II.2.2.23 FireInerMomIy****C++**

```
HRESULT get_FireInerMomIy(double*);
```

**C#**

```
public double FireInerMomIy { get; }
```

**Visual Basic**

```
Public ReadOnly FireInerMomIy As double
```

**Version**

Available since version 3.5.

**II.2.2.24 FireInerMomIz****C++**

```
HRESULT get_FireInerMomIz(double*);
```

**C#**

```
public double FireInerMomIz { get; }
```

**Visual Basic**

```
Public ReadOnly FireInerMomIz As double
```

**Version**

Available since version 3.5.

**II.2.2.25 FireLftEdgeDistVpy****C++**

```
HRESULT get_FireLftEdgeDistVpy(double*);
```

**C#**

```
public double FireLftEdgeDistVpy { get; }
```

**Visual Basic**

```
Public ReadOnly FireLftEdgeDistVpy As double
```

**Version**

Available since version 3.5.

**II.2.2.26 FireLowEdgeDistVpz****C++**

```
HRESULT get_FireLowEdgeDistVpz(double*);
```

**C#**

```
public double FireLowEdgeDistVpz { get; }
```

**Visual Basic**

```
Public ReadOnly FireLowEdgeDistVpz As double
```

**Version**

Available since version 3.5.

**II.2.2.27 FireProtectCoefK2****C++**

```
HRESULT get_FireProtectCoefK2(double*);
```

**C#**

```
public double FireProtectCoefK2 { get; }
```

**Visual Basic**

```
Public ReadOnly FireProtectCoefK2 As double
```

**Version**

Available since version 3.5.

**II.2.2.28 FireProtection****C++**

```
HRESULT get_FireProtection(short*);
```

**C#**

```
public short FireProtection { get; }
```

**Visual Basic**

```
Public ReadOnly FireProtection As short
```

**Version**

Available since version 3.5.

**II.2.2.29 FireProtectionPosition****C++**

```
HRESULT get_FireProtectionPosition(short*);
```

**C#**

```
public short FireProtectionPosition { get; }
```

**Visual Basic**

```
Public ReadOnly FireProtectionPosition As short
```

**Version**

Available since version 3.5.

**II.2.2.30 FireRedCoefK1b****C++**

```
HRESULT get_FireRedCoefK1b(double*);
```

**C#**

```
public double FireRedCoefK1b { get; }
```

**Visual Basic**

```
Public ReadOnly FireRedCoefK1b As double
```

**Version**

Available since version 3.5.

**II.2.2.31 FireRedCoefK1h****C++**

```
HRESULT get_FireRedCoefK1h(double*);
```

**C#**

```
public double FireRedCoefK1h { get; }
```

**Visual Basic**

```
Public ReadOnly FireRedCoefK1h As double
```

**Version**

Available since version 3.5.

**II.2.2.32 FireResistance****C++**

```
HRESULT get_FireResistance(double*);
```

**C#**

```
public double FireResistance { get; }
```

**Visual Basic**

```
Public ReadOnly FireResistance As double
```

**Version**

Available since version 3.5.

**II.2.2.33 FireRgtEdgeDistVy****C++**

```
HRESULT get_FireRgtEdgeDistVy(double*);
```

**C#**

```
public double FireRgtEdgeDistVy { get; }
```

**Visual Basic**

```
Public ReadOnly FireRgtEdgeDistVy As double
```

**Version**

Available since version 3.5.

**II.2.2.34 FireSecAreaInMidSpanS****C++**

```
HRESULT get_FireSecAreaInMidSpanS(double*);
```

**C#**

```
public double FireSecAreaInMidSpanS { get; }
```

**Visual Basic**

```
Public ReadOnly FireSecAreaInMidSpanS As double
```

**Version**

Available since version 3.5.

**II.2.2.35 FireSecAreaS****C++**

```
HRESULT get_FireSecAreaS(double*);
```

**C#**

```
public double FireSecAreaS { get; }
```

**Visual Basic**

```
Public ReadOnly FireSecAreaS As double
```

**Version**

Available since version 3.5.

**II.2.2.36 FireSeModulWy****C++**

```
HRESULT get_FireSeModulWy(double*);
```

**C#**

```
public double FireSeModulWy { get; }
```

**Visual Basic**

```
Public ReadOnly FireSeModulWy As double
```

**Version**

Available since version 3.5.

**II.2.2.37 FireSeModulWZ****C++**

```
HRESULT get_FireSeModulWZ(double*);
```

**C#**

```
public double FireSeModulWZ { get; }
```

**Visual Basic**

```
Public ReadOnly FireSeModulWZ As double
```

**Version**

Available since version 3.5.

**II.2.2.38 FireShearAreaSY****C++**

```
HRESULT get_FireShearAreaSY(double*);
```

**C#**

```
public double FireShearAreaSY { get; }
```

**Visual Basic**

```
Public ReadOnly FireShearAreaSY As double
```

**Version**

Available since version 3.5.

**II.2.2.39 FireShearAreaSZ****C++**

```
HRESULT get_FireShearAreaSZ(double*);
```

**C#**

```
public double FireShearAreaSZ { get; }
```

**Visual Basic**

```
Public ReadOnly FireShearAreaSZ As double
```

**Version**

Available since version 3.5.

**II.2.2.40 FireUprEdgeDistVz****C++**

```
HRESULT get_FireUprEdgeDistVz(double*);
```

**C#**

```
public double FireUprEdgeDistVz { get; }
```

**Visual Basic**

```
Public ReadOnly FireUprEdgeDistVz As double
```

**Version**

Available since version 3.5.

**II.2.2.41 FireWidthAtMembEndB****C++**

```
HRESULT get_FireWidthAtMembEndB(double*);
```

**C#**

```
public double FireWidthAtMembEndB { get; }
```

**Visual Basic**

```
Public ReadOnly FireWidthAtMembEndB As double
```

**Version**

Available since version 3.5.

**II.2.2.42 FireWidthAtMembEndH****C++**

```
HRESULT get_FireWidthAtMembEndH(double*);
```

**C#**

```
public double FireWidthAtMembEndH { get; }
```

**Visual Basic**

```
Public ReadOnly FireWidthAtMembEndH As double
```

**Version**

Available since version 3.5.

**II.2.2.43 FireWidthB****C++**

```
HRESULT get_FireWidthB(double*);
```

**C#**

```
public double FireWidthB { get; }
```

**Visual Basic**

```
Public ReadOnly FireWidthB As double
```

**Version**

Available since version 3.5.

**II.2.2.44 FireWidthH****C++**

```
HRESULT get_FireWidthH(double*);
```

**C#**

```
public double FireWidthH { get; }
```

**Visual Basic**

```
Public ReadOnly FireWidthH As double
```

**Version**

Available since version 3.5.

**II.2.2.45 FireWidthInMidSpanB****C++**

```
HRESULT get_FireWidthInMidSpanB(double*);
```

**C#**

```
public double FireWidthInMidSpanB { get; }
```

**Visual Basic**

```
Public ReadOnly FireWidthInMidSpanB As double
```

**Version**

Available since version 3.5.

**II.2.2.46 IsBuckY****C++**

```
HRESULT get_IsBuckY(short*);
```

**C#**

```
public short IsBuckY { get; }
```

**Visual Basic**

```
Public ReadOnly IsBuckY As short
```

**Version**

Available since version 3.5.

**II.2.2.47 IsBuckZ****C++**

```
HRESULT get_IsBuckZ(short*);
```

**C#**

```
public short IsBuckZ { get; }
```

**Visual Basic**

```
Public ReadOnly IsBuckZ As short
```

**Version**

Available since version 3.5.

**II.2.2.48 LatBuckCoefKinst****C++**

```
HRESULT get_LatBuckCoefKinst(double*);
```

**C#**

```
public double LatBuckCoefKinst { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckCoefKinst As double
```

**Version**

Available since version 3.5.

**II.2.2.49 LatBuckLengthCoef****C++**

```
HRESULT get_LatBuckLengthCoef(double*);
```

**C#**

```
public double LatBuckLengthCoef { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckLengthCoef As double
```

**Version**

Available since version 3.5.

**II.2.2.50 LatBuckLengthLd****C++**

```
HRESULT get_LatBuckLengthLd(double*);
```

**C#**

```
public double LatBuckLengthLd { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckLengthLd As double
```

**Version**

Available since version 3.5.

**II.2.2.51 LatBuckRelSlendLam****C++**

```
HRESULT get_LatBuckRelSlendLam(double*);
```

**C#**

```
public double LatBuckRelSlendLam { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckRelSlendLam As double
```

**Version**

Available since version 3.5.

**II.2.2.52 LatBuckStrsCrit****C++**

```
HRESULT get_LatBuckStrsCrit(double*);
```

**C#**

```
public double LatBuckStrsCrit { get; }
```

**Visual Basic**

```
Public ReadOnly LatBuckStrsCrit As double
```

**Version**

Available since version 3.5.

**II.2.2.53 MatAxCompResist****C++**

```
HRESULT get_MatAxCompResist(double*);
```

**C#**

```
public double MatAxCompResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatAxCompResist As double
```

**Version**

Available since version 3.5.

**II.2.2.54 MatAxTensResist****C++**

```
HRESULT get_MatAxTensResist(double*);
```

**C#**

```
public double MatAxTensResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatAxTensResist As double
```

**Version**

Available since version 3.5.

**II.2.2.55 MatBendResist****C++**

```
HRESULT get_MatBendResist(double*);
```

**C#**

```
public double MatBendResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatBendResist As double
```

**Version**

Available since version 3.5.

**II.2.2.56 MatShearResist****C++**

```
HRESULT get_MatShearResist(double*);
```

**C#**

```
public double MatShearResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatShearResist As double
```

**Version**

Available since version 3.5.

**II.2.2.57 MatTrCompResist****C++**

```
HRESULT get_MatTrCompResist(double*);
```

**C#**

```
public double MatTrCompResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatTrCompResist As double
```

**Version**

Available since version 3.5.

**II.2.2.58 MatTrTensResist****C++**

```
HRESULT get_MatTrTensResist(double*);
```

**C#**

```
public double MatTrTensResist { get; }
```

**Visual Basic**

```
Public ReadOnly MatTrTensResist As double
```

**Version**

Available since version 3.5.

**II.2.2.59 StrsBend****C++**

```
HRESULT get_StrsBend(double*);
```

**C#**

```
public double StrsBend { get; }
```

**Visual Basic**

```
Public ReadOnly StrsBend As double
```

**Version**

Available since version 3.5.

**II.2.2.60 StrsBendInCompEdgeMY****C++**

```
HRESULT get_StrsBendInCompEdgeMY(double*);
```

**C#**

```
public double StrsBendInCompEdgeMY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsBendInCompEdgeMY As double
```

**Version**

Available since version 3.5.

**II.2.2.61 StrsBendInCompEdgeMZ****C++**

```
HRESULT get_StrsBendInCompEdgeMZ(double*);
```

**C#**

```
public double StrsBendInCompEdgeMZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsBendInCompEdgeMZ As double
```

**Version**

Available since version 3.5.

**II.2.2.62 StrsBendInCurveBeam****C++**

```
HRESULT get_StrsBendInCurveBeam(double*);
```

**C#**

```
public double StrsBendInCurveBeam { get; }
```

**Visual Basic**

```
Public ReadOnly StrsBendInCurveBeam As double
```

**Version**

Available since version 3.5.

**II.2.2.63 StrsComp****C++**

```
HRESULT get_StrsComp(double*);
```

**C#**

```
public double StrsComp { get; }
```

**Visual Basic**

```
Public ReadOnly StrsComp As double
```

**Version**

Available since version 3.5.

**II.2.2.64 StrsFinal****C++**

```
HRESULT get_StrsFinal(double*);
```

**C#**

```
public double StrsFinal { get; }
```

**Visual Basic**

```
Public ReadOnly StrsFinal As double
```

**Version**

Available since version 3.5.

**II.2.2.65 StrsLftEdgeMZ****C++**

```
HRESULT get_StrsLftEdgeMZ(double*);
```

**C#**

```
public double StrsLftEdgeMZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsLftEdgeMZ As double
```

**II.2.2.66 StrsLowEdgeMY****C++**

```
HRESULT get_StrsLowEdgeMY(double*);
```

**C#**

```
public double StrsLowEdgeMY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsLowEdgeMY As double
```

**Version**

Available since version 3.5.

**II.2.2.67 StrsRgtEdgeMZ****C++**

```
HRESULT get_StrsRgtEdgeMZ(double*);
```

**C#**

```
public double StrsRgtEdgeMZ { get; }
```

**Visual Basic**

```
Public ReadOnly StrsRgtEdgeMZ As double
```

**Version**

Available since version 3.5.

**II.2.2.68 StrsShearY****C++**

```
HRESULT get_StrsShearY(double*);
```

**C#**

```
public double StrsShearY { get; }
```

**Visual Basic**

```
Public ReadOnly StrsShearY As double
```

**Version**

Available since version 3.5.

### II.2.2.69 StrsShearZ

#### C++

```
HRESULT get_StrsShearZ(double*);
```

#### C#

```
public double StrsShearZ { get; }
```

#### Visual Basic

```
Public ReadOnly StrsShearZ As double
```

#### Version

Available since version 3.5.

### II.2.2.70 StrsTens

#### C++

```
HRESULT get_StrsTens(double*);
```

#### C#

```
public double StrsTens { get; }
```

#### Visual Basic

```
Public ReadOnly StrsTens As double
```

#### Version

Available since version 3.5.

### II.2.2.71 StrsUprEdgeMY

#### C++

```
HRESULT get_StrsUprEdgeMY(double*);
```

#### C#

```
public double StrsUprEdgeMY { get; }
```

#### Visual Basic

```
Public ReadOnly StrsUprEdgeMY As double
```

#### Version

Available since version 3.5.

### II.2.2.72 TimberType

#### C++

```
HRESULT get_TimberType(short*);
```

#### C#

```
public short TimberType { get; }
```

#### Visual Basic

```
Public ReadOnly TimberType As short
```

#### Version

Available since version 3.5.

## II.3 IRDimBuckDiagramCB71

### C++

```
enum IRDimBuckDiagramCB71;
```

### C#

```
public enum IRDimBuckDiagramCB71;
```

### Visual Basic

```
Public Enum IRDimBuckDiagramCB71
```

### Members

Members	Description
I_BD_CB71_PINNED_PINNED_1_0 = 0	Available since version 3.5.
I_BD_CB71_STIFF_STIFF_0_65 = 1	Available since version 3.5.
I_BD_CB71_STIFF_PINNED_0_8 = 2	Available since version 3.5.
I_BD_CB71_CANTILEVER = 3	Available since version 3.5.
I_BD_CB71_NONE = 4	Available since version 3.5.
I_BD_CB71_INTERNAL_BRACINGS = 5	Available since version 3.5.

### Version

Available since version 3.5.

## II.4 IRDimLaterBuckTypeCB71

### C++

```
enum IRDimLaterBuckTypeCB71;
```

### C#

```
public enum IRDimLaterBuckTypeCB71;
```

### Visual Basic

```
Public Enum IRDimLaterBuckTypeCB71
```

### Members

Members	Description
I_LBT_CB71_PINNED_SUPPORTS = 0	Available since version 3.5.
I_LBT_CB71_FIXED_SUPPORTS = 1	Available since version 3.5.
I_LBT_CB71_CANTILEVER = 2	Available since version 3.5.
I_LBT_CB71_NONE = 3	Available since version 3.5.

### Version

Available since version 3.5.

## II.5 IRDimLoadLevelCB71

### C++

```
enum IRDimLoadLevelCB71;
```

### C#

```
public enum IRDimLoadLevelCB71;
```

**Visual Basic**

```
Public Enum IRDimLoadLevelCB71
```

**Members**

Members	Description
I_LL_CB71_UP = 0	Available since version 3.5.
I_LL_CB71_MIDDLE = 1	Available since version 3.5.
I_LL_CB71_BOTTOM = 2	Available since version 3.5.

**Version**

Available since version 3.5.

**II.6 IRDimLoadTypeCB71****C++**

```
enum IRDimLoadTypeCB71;
```

**C#**

```
public enum IRDimLoadTypeCB71;
```

**Visual Basic**

```
Public Enum IRDimLoadTypeCB71
```

**Members**

Members	Description
I_DLT_CB71_MOMENT_AT_END = 0	Available since version 3.5.
I_DLT_CB71_UNIFORM_LOAD = 1	Available since version 3.5.
I_DLT_CB71_FORCE_AT_DISTANCE = 2	Available since version 3.5.

**Version**

Available since version 3.5.

**II.7 IRDimFireMemberPositionsCB71****C++**

```
enum IRDimFireMemberPositionsCB71;
```

**C#**

```
public enum IRDimFireMemberPositionsCB71;
```

**Visual Basic**

```
Public Enum IRDimFireMemberPositionsCB71
```

**Members**

Members	Description
I_FMP_CB71_AUTOMATIC = 0	Available since version 3.5.
I_FMP_CB71_VERTICAL = 1	Available since version 3.5.
I_FMP_CB71_HORIZONTAL = 2	Available since version 3.5.

**Version**

Available since version 3.5.

## II.8 IRDimFireProtectionTimesCB71

### C++

```
enum IRDimFireProtectionTimesCB71;
```

### C#

```
public enum IRDimFireProtectionTimesCB71;
```

### Visual Basic

```
Public Enum IRDimFireProtectionTimesCB71
```

### Members

Members	Description
I_FPT_CB71_LESS = 0	< 1/4. Available since version 3.5.
I_FPT_CB71_MORE = 1	> 1/4. Available since version 3.5.

### Version

Available since version 3.5.

## III IRDimStreamType

### C++

```
enum IRDimStreamType;
```

### C#

```
public enum IRDimStreamType;
```

### Visual Basic

```
Public Enum IRDimStreamType
```

## IV IRDimStream

### Class Hierarchy

### C++

```
interface IRDimStream : IDispatch;
```

### C#

```
public interface IRDimStream;
```

### Visual Basic

```
Public Interface IRDimStream
```

## IV.1 IRDimStream Members

The following tables list the members exposed by IRDimStream.

## Public Methods

	Name	Description
≡	Clear ( [ see page 1882 )	
≡	ReadDouble ( [ see page 1882 )	
≡	ReadLong ( [ see page 1882 )	
≡	ReadText ( [ see page 1883 )	
≡	SeekSet ( [ see page 1883 )	
≡	Size ( [ see page 1883 )	
≡	WriteDouble ( [ see page 1883 )	
≡	WriteLong ( [ see page 1883 )	
≡	WriteText ( [ see page 1884 )	

## IV.2 IRDimStream Methods

The methods of the IRDimStream class are listed here.

### Public Methods

	Name	Description
≡	Clear ( [ see page 1882 )	
≡	ReadDouble ( [ see page 1882 )	
≡	ReadLong ( [ see page 1882 )	
≡	ReadText ( [ see page 1883 )	
≡	SeekSet ( [ see page 1883 )	
≡	Size ( [ see page 1883 )	
≡	WriteDouble ( [ see page 1883 )	
≡	WriteLong ( [ see page 1883 )	
≡	WriteText ( [ see page 1884 )	

### IV.2.1 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

### IV.2.2 ReadDouble

#### C++

```
HRESULT ReadDouble(double* ret);
```

#### C#

```
public double ReadDouble();
```

#### Visual Basic

```
Public Function ReadDouble() As double
```

### IV.2.3 ReadLong

C++

```
HRESULT ReadLong(long* ret);
```

C#

```
public long ReadLong();
```

Visual Basic

```
Public Function ReadLong() As long
```

### IV.2.4 ReadText

C++

```
HRESULT ReadText(BSTR* ret);
```

C#

```
public String ReadText();
```

Visual Basic

```
Public Function ReadText() As String
```

### IV.2.5 SeekSet

C++

```
HRESULT SeekSet(IRDimStreamType _type, long _pos);
```

C#

```
public void SeekSet(IRDimStreamType _type, long _pos);
```

Visual Basic

```
Public Sub SeekSet(_type As IRDimStreamType, _pos As long)
```

### IV.2.6 Size

C++

```
HRESULT Size(IRDimStreamType _type, long* ret);
```

C#

```
public long Size(IRDimStreamType _type);
```

Visual Basic

```
Public Function Size(_type As IRDimStreamType) As long
```

### IV.2.7 WriteDouble

C++

```
HRESULT WriteDouble(double _val);
```

C#

```
public void WriteDouble(double _val);
```

**Visual Basic**

```
Public Sub WriteDouble(_val As double)
```

**IV.2.8 WriteLong****C++**

```
HRESULT WriteLong(long _val);
```

**C#**

```
public void WriteLong(long _val);
```

**Visual Basic**

```
Public Sub WriteLong(_val As long)
```

**IV.2.9 WriteText****C++**

```
HRESULT WriteText(BSTR _val);
```

**C#**

```
public void WriteText(String _val);
```

**Visual Basic**

```
Public Sub WriteText(_val As String)
```

**V IRDimMembDefType****C++**

```
enum IRDimMembDefType;
```

**C#**

```
public enum IRDimMembDefType;
```

**Visual Basic**

```
Public Enum IRDimMembDefType
```

**VI IRDimMembDefMatType****C++**

```
enum IRDimMembDefMatType;
```

**C#**

```
public enum IRDimMembDefMatType;
```

**Visual Basic**

```
Public Enum IRDimMembDefMatType
```

**Members**

Members	Description
I_DMDMT_CONCRETE = 3	Available since version 7.5.

## VII IRDimMembDefLengthDataType

C++

```
enum IRDimMembDefLengthDataType;
```

C#

```
public enum IRDimMembDefLengthDataType;
```

Visual Basic

```
Public Enum IRDimMembDefLengthDataType
```

## VIII IRDimMembDefBucklingDataType

C++

```
enum IRDimMembDefBucklingDataType;
```

C#

```
public enum IRDimMembDefBucklingDataType;
```

Visual Basic

```
Public Enum IRDimMembDefBucklingDataType
```

## IX IRDimMembDefDispDataType

C++

```
enum IRDimMembDefDispDataType;
```

C#

```
public enum IRDimMembDefDispDataType;
```

Visual Basic

```
Public Enum IRDimMembDefDispDataType
```

## X IRDimMembDefDeflDataType

C++

```
enum IRDimMembDefDeflDataType;
```

C#

```
public enum IRDimMembDefDeflDataType;
```

Visual Basic

```
Public Enum IRDimMembDefDeflDataType
```

## XI IRDimMembDef

Class Hierarchy

**C++**

```
interface IRDimMembDef : IDispatch;
```

**C#**

```
public interface IRDimMembDef;
```

**Visual Basic**

```
Public Interface IRDimMembDef
```

## XI.1 IRDimMembDef Members

The following tables list the members exposed by IRDimMembDef.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	ClientID (see page 1886)	
◆	Length (see page 1886)	
◆	MatType (see page 1887)	
◆	Name (see page 1887)	
◆	Type (see page 1887)	

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	DeflYZRelLimit (see page 1888)	
◆	DispXYRelLimit (see page 1888)	
◆	IsBuckCoefConst (see page 1888)	
◆	IsDeflectionYZ (see page 1888)	
◆	IsDisplacementXY (see page 1888)	
◆	LengthYZUV (see page 1889)	
◆	Retrieve (see page 1889)	
◆	Store (see page 1889)	

## XI.2 IRDimMembDef Fields

The fields of the IRDimMembDef class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	ClientID (see page 1886)	
◆	Length (see page 1886)	
◆	MatType (see page 1887)	
◆	Name (see page 1887)	
◆	Type (see page 1887)	

### XI.2.1 ClientID

**C++**

```
HRESULT get_ClientID(BSTR*);
```

**C#**

```
public String ClientID { get; }
```

**Visual Basic**

```
Public ReadOnly ClientID As String
```

**XI.2.2 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**XI.2.3 MatType****C++**

```
HRESULT get_MatType(IRDimMembDefMatType*);
```

**C#**

```
public IRDimMembDefMatType MatType { get; }
```

**Visual Basic**

```
Public ReadOnly MatType As IRDimMembDefMatType
```

**XI.2.4 Name****C++**

```
HRESULT get_Name(BSTR*);
```

**C#**

```
public String Name { get; }
```

**Visual Basic**

```
Public ReadOnly Name As String
```

**XI.2.5 Type****C++**

```
HRESULT get_Type(IRDimMembDefType*);  
HRESULT put_Type(IRDimMembDefType*);
```

**C#**

```
public IRDimMembDefType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRDimMembDefType
```

**XI.3 IRDimMembDef Methods**

The methods of the IRDimMembDef class are listed here.

## Public Methods

	Name	Description
≡	DeflYZRelLimit ( [ see page 1888] )	
≡	DispIXYRelLimit ( [ see page 1888] )	
≡	IsBuckCoefConst ( [ see page 1888] )	
≡	IsDeflectionYZ ( [ see page 1888] )	
≡	IsDisplacementXY ( [ see page 1888] )	
≡	LengthYZUV ( [ see page 1889] )	
≡	Retrieve ( [ see page 1889] )	
≡	Store ( [ see page 1889] )	

### XI.3.1 DeflYZRelLimit

C++

```
HRESULT DeflYZRelLimit(IRDimMembDefDeflDataType _type, double* ret);
```

C#

```
public double DeflYZRelLimit(IRDimMembDefDeflDataType _type);
```

Visual Basic

```
Public Function DeflYZRelLimit(_type As IRDimMembDefDeflDataType) As double
```

### XI.3.2 DispIXYRelLimit

C++

```
HRESULT DispIXYRelLimit(IRDimMembDefDispDataType _type, double* ret);
```

C#

```
public double DispIXYRelLimit(IRDimMembDefDispDataType _type);
```

Visual Basic

```
Public Function DispIXYRelLimit(_type As IRDimMembDefDispDataType) As double
```

### XI.3.3 IsBuckCoefConst

C++

```
HRESULT IsBuckCoefConst(IRDimMembDefBucklingDataType _type, VARIANT_BOOL* ret);
```

C#

```
public bool IsBuckCoefConst(IRDimMembDefBucklingDataType _type);
```

Visual Basic

```
Public Function IsBuckCoefConst(_type As IRDimMembDefBucklingDataType) As Boolean
```

### XI.3.4 IsDeflectionYZ

C++

```
HRESULT IsDeflectionYZ(IRDimMembDefDeflDataType _type, VARIANT_BOOL* ret);
```

C#

```
public bool IsDeflectionYZ(IRDimMembDefDeflDataType _type);
```

**Visual Basic**

```
Public Function IsDeflectionYZ(_type As IRDimMembDefDef1DataType) As Boolean
```

**XI.3.5 IsDisplacementXY****C++**

```
HRESULT IsDisplacementXY(IRDimMembDefDispDataType _type, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsDisplacementXY(IRDimMembDefDispDataType _type);
```

**Visual Basic**

```
Public Function IsDisplacementXY(_type As IRDimMembDefDispDataType) As Boolean
```

**XI.3.6 LengthYZUV****C++**

```
HRESULT LengthYZUV(IRDimMembDefLengthDataType _type, double* ret);
```

**C#**

```
public double LengthYZUV(IRDimMembDefLengthDataType _type);
```

**Visual Basic**

```
Public Function LengthYZUV(_type As IRDimMembDefLengthDataType) As double
```

**XI.3.7 Retrieve****C++**

```
HRESULT Retrieve(IRDimStream* _str);
```

**C#**

```
public void Retrieve(IRDimStream _str);
```

**Visual Basic**

```
Public Sub Retrieve(ByRef _str As IRDimStream)
```

**XI.3.8 Store****C++**

```
HRESULT Store(IRDimStream* _str);
```

**C#**

```
public void Store(IRDimStream _str);
```

**Visual Basic**

```
Public Sub Store(ByRef _str As IRDimStream)
```

**XII IRDimEffDefParamType****C++**

```
enum IRDimEffDefParamType;
```

**C#**

```
public enum IRDimEffDefParamType;
```

**Visual Basic**

```
Public Enum IRDimEffDefParamType
```

## XIII IRDimEffDefDirType

**C++**

```
enum IRDimEffDefDirType;
```

**C#**

```
public enum IRDimEffDefDirType;
```

**Visual Basic**

```
Public Enum IRDimEffDefDirType
```

## XIV IRDimEffDefIntPsType

**C++**

```
enum IRDimEffDefIntPsType;
```

**C#**

```
public enum IRDimEffDefIntPsType;
```

**Visual Basic**

```
Public Enum IRDimEffDefIntPsType
```

## XV IRDimEffDef

**Class Hierarchy****C++**

```
interface IRDimEffDef : IDispatch;
```

**C#**

```
public interface IRDimEffDef;
```

**Visual Basic**

```
Public Interface IRDimEffDef
```

### XV.1 IRDimEffDef Members

The following tables list the members exposed by IRDimEffDef.

**Public Fields**

	Name	Description
◆	MX (↗ see page 1892)	
◆	MY (↗ see page 1892)	
◆	MZ (↗ see page 1892)	

◆	N ( <a href="#">see page 1892</a> )	
◆	QY ( <a href="#">see page 1893</a> )	
◆	QZ ( <a href="#">see page 1893</a> )	

**Public Methods**

	Name	Description
◆	Clear ( <a href="#">see page 1894</a> )	
◆	Read_IntPsEff_M_1P4L ( <a href="#">see page 1894</a> )	
◆	Read_IntPsEff_M_3P4L ( <a href="#">see page 1894</a> )	
◆	Read_IntPsEff_M_MID ( <a href="#">see page 1895</a> )	
◆	Read_IntPsEff_M1 ( <a href="#">see page 1895</a> )	
◆	Read_IntPsEff_M12 ( <a href="#">see page 1895</a> )	
◆	Read_IntPsEff_M2 ( <a href="#">see page 1895</a> )	
◆	Read_IntPsEff_MN_MAX ( <a href="#">see page 1895</a> )	
◆	Read_IntPsEff_MP_MAX ( <a href="#">see page 1896</a> )	
◆	Read_M_1P4L ( <a href="#">see page 1896</a> )	
◆	Read_M_3P4L ( <a href="#">see page 1896</a> )	
◆	Read_M_MID ( <a href="#">see page 1896</a> )	
◆	Read_M1 ( <a href="#">see page 1896</a> )	
◆	Read_M12 ( <a href="#">see page 1897</a> )	
◆	Read_M2 ( <a href="#">see page 1897</a> )	
◆	Read_MN_MAX ( <a href="#">see page 1897</a> )	
◆	Read_MP_MAX ( <a href="#">see page 1897</a> )	
◆	ReadParam ( <a href="#">see page 1897</a> )	
◆	WriteForces ( <a href="#">see page 1898</a> )	
◆	WriteIntPsEffSet1 ( <a href="#">see page 1898</a> )	
◆	WriteIntPsEffSet2 ( <a href="#">see page 1898</a> )	
◆	WriteParam ( <a href="#">see page 1898</a> )	
◆	WriteValuesSet1 ( <a href="#">see page 1899</a> )	
◆	WriteValuesSet2 ( <a href="#">see page 1899</a> )	

**XV.2 IRDimEffDef Fields**

The fields of the IRDimEffDef class are listed here.

**Public Fields**

	Name	Description
◆	MX ( <a href="#">see page 1892</a> )	
◆	MY ( <a href="#">see page 1892</a> )	
◆	MZ ( <a href="#">see page 1892</a> )	
◆	N ( <a href="#">see page 1892</a> )	
◆	QY ( <a href="#">see page 1893</a> )	

	QZ (see page 1893)	
--	--------------------	--

## XV.2.1 MX

**C++**

```
HRESULT get_MX(double*);
```

**C#**

```
public double MX { get; }
```

**Visual Basic**

```
Public ReadOnly MX As double
```

## XV.2.2 MY

**C++**

```
HRESULT get_MY(double*);
```

**C#**

```
public double MY { get; }
```

**Visual Basic**

```
Public ReadOnly MY As double
```

## XV.2.3 MZ

**C++**

```
HRESULT get_MZ(double*);
```

**C#**

```
public double MZ { get; }
```

**Visual Basic**

```
Public ReadOnly MZ As double
```

## XV.2.4 N

**C++**

```
HRESULT get_N(double*);
```

**C#**

```
public double N { get; }
```

**Visual Basic**

```
Public ReadOnly N As double
```

## XV.2.5 QY

**C++**

```
HRESULT get_QY(double*);
```

**C#**

```
public double QY { get; }
```

**Visual Basic**

```
Public ReadOnly QY As double
```

## XV.2.6 QZ

### C++

```
HRESULT get_QZ(double*);
```

### C#

```
public double QZ { get; }
```

### Visual Basic

```
Public ReadOnly QZ As Double
```

## XV.3 IRDimEffDef Methods

The methods of the IRDimEffDef class are listed here.

### Public Methods

	Name	Description
≡	Clear ( [ see page 1894 )	
≡	Read_IntPsEff_M_1P4L ( [ see page 1894 )	
≡	Read_IntPsEff_M_3P4L ( [ see page 1894 )	
≡	Read_IntPsEff_M_MID ( [ see page 1895 )	
≡	Read_IntPsEff_M1 ( [ see page 1895 )	
≡	Read_IntPsEff_M12 ( [ see page 1895 )	
≡	Read_IntPsEff_M2 ( [ see page 1895 )	
≡	Read_IntPsEff_MN_MAX ( [ see page 1895 )	
≡	Read_IntPsEff_MP_MAX ( [ see page 1896 )	
≡	Read_M_1P4L ( [ see page 1896 )	
≡	Read_M_3P4L ( [ see page 1896 )	
≡	Read_M_MID ( [ see page 1896 )	
≡	Read_M1 ( [ see page 1896 )	
≡	Read_M12 ( [ see page 1897 )	
≡	Read_M2 ( [ see page 1897 )	
≡	Read_MN_MAX ( [ see page 1897 )	
≡	Read_MP_MAX ( [ see page 1897 )	
≡	ReadParam ( [ see page 1897 )	
≡	WriteForces ( [ see page 1898 )	
≡	WriteIntPsEffSet1 ( [ see page 1898 )	
≡	WriteIntPsEffSet2 ( [ see page 1898 )	
≡	WriteParam ( [ see page 1898 )	
≡	WriteValuesSet1 ( [ see page 1899 )	
≡	WriteValuesSet2 ( [ see page 1899 )	

### XV.3.1 Clear

C++

```
HRESULT Clear();
```

C#

```
public void Clear();
```

Visual Basic

```
Public Sub Clear()
```

### XV.3.2 Read\_IntPsEff\_M\_1P4L

C++

```
HRESULT Read_IntPsEff_M_1P4L(IRDimEffDefIntPsType _type, double* ret);
```

C#

```
public double Read_IntPsEff_M_1P4L(IRDimEffDefIntPsType _type);
```

Visual Basic

```
Public Function Read_IntPsEff_M_1P4L(_type As IRDimEffDefIntPsType) As double
```

### XV.3.3 Read\_IntPsEff\_M\_3P4L

C++

```
HRESULT Read_IntPsEff_M_3P4L(IRDimEffDefIntPsType _type, double* ret);
```

C#

```
public double Read_IntPsEff_M_3P4L(IRDimEffDefIntPsType _type);
```

Visual Basic

```
Public Function Read_IntPsEff_M_3P4L(_type As IRDimEffDefIntPsType) As double
```

### XV.3.4 Read\_IntPsEff\_M\_MID

C++

```
HRESULT Read_IntPsEff_M_MID(IRDimEffDefIntPsType _type, double* ret);
```

C#

```
public double Read_IntPsEff_M_MID(IRDimEffDefIntPsType _type);
```

Visual Basic

```
Public Function Read_IntPsEff_M_MID(_type As IRDimEffDefIntPsType) As double
```

### XV.3.5 Read\_IntPsEff\_M1

C++

```
HRESULT Read_IntPsEff_M1(IRDimEffDefIntPsType _type, double* ret);
```

C#

```
public double Read_IntPsEff_M1(IRDimEffDefIntPsType _type);
```

**Visual Basic**

```
Public Function Read_IntPsEff_M1(_type As IRDimEffDefIntPsType) As double
```

**XV.3.6 Read\_IntPsEff\_M12****C++**

```
HRESULT Read_IntPsEff_M12(IRDimEffDefIntPsType _type, double* ret);
```

**C#**

```
public double Read_IntPsEff_M12(IRDimEffDefIntPsType _type);
```

**Visual Basic**

```
Public Function Read_IntPsEff_M12(_type As IRDimEffDefIntPsType) As double
```

**XV.3.7 Read\_IntPsEff\_M2****C++**

```
HRESULT Read_IntPsEff_M2(IRDimEffDefIntPsType _type, double* ret);
```

**C#**

```
public double Read_IntPsEff_M2(IRDimEffDefIntPsType _type);
```

**Visual Basic**

```
Public Function Read_IntPsEff_M2(_type As IRDimEffDefIntPsType) As double
```

**XV.3.8 Read\_IntPsEff\_MN\_MAX****C++**

```
HRESULT Read_IntPsEff_MN_MAX(IRDimEffDefIntPsType _type, double* ret);
```

**C#**

```
public double Read_IntPsEff_MN_MAX(IRDimEffDefIntPsType _type);
```

**Visual Basic**

```
Public Function Read_IntPsEff_MN_MAX(_type As IRDimEffDefIntPsType) As double
```

**XV.3.9 Read\_IntPsEff\_MP\_MAX****C++**

```
HRESULT Read_IntPsEff_MP_MAX(IRDimEffDefIntPsType _type, double* ret);
```

**C#**

```
public double Read_IntPsEff_MP_MAX(IRDimEffDefIntPsType _type);
```

**Visual Basic**

```
Public Function Read_IntPsEff_MP_MAX(_type As IRDimEffDefIntPsType) As double
```

**XV.3.10 Read\_M\_1P4L****C++**

```
HRESULT Read_M_1P4L(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M_1P4L(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M_1P4L(_type As IRDimEffDefDirType) As double
```

**XV.3.11 Read\_M\_3P4L****C++**

```
HRESULT Read_M_3P4L(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M_3P4L(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M_3P4L(_type As IRDimEffDefDirType) As double
```

**XV.3.12 Read\_M\_MID****C++**

```
HRESULT Read_M_MID(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M_MID(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M_MID(_type As IRDimEffDefDirType) As double
```

**XV.3.13 Read\_M1****C++**

```
HRESULT Read_M1(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M1(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M1(_type As IRDimEffDefDirType) As double
```

**XV.3.14 Read\_M12****C++**

```
HRESULT Read_M12(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M12(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M12(_type As IRDimEffDefDirType) As double
```

**XV.3.15 Read\_M2****C++**

```
HRESULT Read_M2(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_M2(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_M2(_type As IRDimEffDefDirType) As double
```

**XV.3.16 Read\_MN\_MAX****C++**

```
HRESULT Read_MN_MAX(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_MN_MAX(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_MN_MAX(_type As IRDimEffDefDirType) As double
```

**XV.3.17 Read\_MP\_MAX****C++**

```
HRESULT Read_MP_MAX(IRDimEffDefDirType _type, double* ret);
```

**C#**

```
public double Read_MP_MAX(IRDimEffDefDirType _type);
```

**Visual Basic**

```
Public Function Read_MP_MAX(_type As IRDimEffDefDirType) As double
```

**XV.3.18 ReadParam****C++**

```
HRESULT ReadParam(IRDimEffDefParamType _type, long* ret);
```

**C#**

```
public long ReadParam(IRDimEffDefParamType _type);
```

**Visual Basic**

```
Public Function ReadParam(_type As IRDimEffDefParamType) As long
```

**XV.3.19 WriteForces****C++**

```
HRESULT WriteForces(double _N, double _QY, double _QZ, double _MX, double _MY, double _MZ);
```

**C#**

```
public void WriteForces(double _N, double _QY, double _QZ, double _MX, double _MY, double _MZ);
```

**Visual Basic**

```
Public Sub WriteForces(_N As double, _QY As double, _QZ As double, _MX As double, _MY As double, _MZ As double)
```

**XV.3.20 WriteIntPsEffSet1****C++**

```
HRESULT WriteIntPsEffSet1(IRDimEffDefIntPsType _type, double _M1, double _M2, double _M12);
```

**C#**

```
public void WriteIntPsEffSet1(IRDimEffDefIntPsType _type, double _M1, double _M2, double _M12);
```

**Visual Basic**

```
Public Sub WriteIntPsEffSet1(_type As IRDimEffDefIntPsType, _M1 As Double, _M2 As Double, _M12 As Double)
```

**XV.3.21 WriteIntPsEffSet2****C++**

```
HRESULT WriteIntPsEffSet2(IRDimEffDefIntPsType _type, double _MP_MAX, double _MN_MAX, double _M_MID, double _M_1P4L, double _M_3P4L);
```

**C#**

```
public void WriteIntPsEffSet2(IRDimEffDefIntPsType _type, double _MP_MAX, double _MN_MAX, double _M_MID, double _M_1P4L, double _M_3P4L);
```

**Visual Basic**

```
Public Sub WriteIntPsEffSet2(_type As IRDimEffDefIntPsType, _MP_MAX As Double, _MN_MAX As Double, _M_MID As Double, _M_1P4L As Double, _M_3P4L As Double)
```

**XV.3.22 WriteParam****C++**

```
HRESULT WriteParam(IRDimEffDefParamType _type, long _val);
```

**C#**

```
public void WriteParam(IRDimEffDefParamType _type, long _val);
```

**Visual Basic**

```
Public Sub WriteParam(_type As IRDimEffDefParamType, _val As Long)
```

**XV.3.23 WriteValuesSet1****C++**

```
HRESULT WriteValuesSet1(IRDimEffDefDirType _type, double _M1, double _M2, double _M12);
```

**C#**

```
public void WriteValuesSet1(IRDimEffDefDirType _type, double _M1, double _M2, double _M12);
```

**Visual Basic**

```
Public Sub WriteValuesSet1(_type As IRDimEffDefDirType, _M1 As Double, _M2 As Double, _M12 As Double)
```

**XV.3.24 WriteValuesSet2****C++**

```
HRESULT WriteValuesSet2(IRDimEffDefDirType _type, double _MP_MAX, double _MN_MAX, double _M_MID, double _M_1P4L, double _M_3P4L);
```

**C#**

```
public void WriteValuesSet2(IRDimEffDefDirType _type, double _MP_MAX, double _MN_MAX,  
double _M_MID, double _M_1P4L, double _M_3P4L);
```

**Visual Basic**

```
Public Sub WriteValuesSet2(_type As IRDimEffDefDirType, _MP_MAX As Double, _MN_MAX As  
Double, _M_MID As Double, _M_1P4L As Double, _M_3P4L As Double)
```

## XVI IRDimMatDefType

**C++**

```
enum IRDimMatDefType;
```

**C#**

```
public enum IRDimMatDefType;
```

**Visual Basic**

```
Public Enum IRDimMatDefType
```

## XVII IRDimMatDefValType

**C++**

```
enum IRDimMatDefValType;
```

**C#**

```
public enum IRDimMatDefValType;
```

**Visual Basic**

```
Public Enum IRDimMatDefValType
```

## XVIII IRDimMatDefLongExValType

**C++**

```
enum IRDimMatDefLongExValType;
```

**C#**

```
public enum IRDimMatDefLongExValType;
```

**Visual Basic**

```
Public Enum IRDimMatDefLongExValType
```

## XIX IRDimMatDefDblExValType

**C++**

```
enum IRDimMatDefDblExValType;
```

**C#**

```
public enum IRDimMatDefDblExValType;
```

## Visual Basic

```
Public Enum IRDimMatDefDblExValType
```

# XX IRDimMatDef

## Class Hierarchy

### C++

```
interface IRDimMatDef : IDispatch;
```

### C#

```
public interface IRDimMatDef;
```

## Visual Basic

```
Public Interface IRDimMatDef
```

## XX.1 IRDimMatDef Members

The following tables list the members exposed by IRDimMatDef.

### Public Fields

	Name	Description
◆	Name ( <a href="#">see page 1901</a> )	
◆	SecondName ( <a href="#">see page 1901</a> )	
◆	Type ( <a href="#">see page 1901</a> )	

### Public Methods

	Name	Description
≡◆	ReadDoubleExtraValue ( <a href="#">see page 1902</a> )	
≡◆	ReadLongExtraValue ( <a href="#">see page 1902</a> )	
≡◆	ReadValue ( <a href="#">see page 1902</a> )	
≡◆	SetNames ( <a href="#">see page 1903</a> )	
≡◆	WriteDoubleExtraValue ( <a href="#">see page 1903</a> )	
≡◆	WriteLongExtraValue ( <a href="#">see page 1903</a> )	
≡◆	WriteValue ( <a href="#">see page 1903</a> )	

## XX.2 IRDimMatDef Fields

The fields of the IRDimMatDef class are listed here.

### Public Fields

	Name	Description
◆	Name ( <a href="#">see page 1901</a> )	
◆	SecondName ( <a href="#">see page 1901</a> )	
◆	Type ( <a href="#">see page 1901</a> )	

## XX.2.1 Name

### C++

```
HRESULT get_Name(BSTR*);
```

### C#

```
public String Name { get; }
```

### Visual Basic

```
Public ReadOnly Name As String
```

## XX.2.2 SecondName

### C++

```
HRESULT get_SecondName(BSTR*);
```

### C#

```
public String SecondName { get; }
```

### Visual Basic

```
Public ReadOnly SecondName As String
```

## XX.2.3 Type

### C++

```
HRESULT get_Type(IRDIMatDefType*);  
HRESULT put_Type(IRDIMatDefType*);
```

### C#

```
public IRDIMatDefType Type { get; set; }
```

### Visual Basic

```
Public Type As IRDIMatDefType
```

## XX.3 IRDimMatDef Methods

The methods of the IRDimMatDef class are listed here.

### Public Methods

	Name	Description
≡	ReadDoubleExtraValue (see page 1902)	
≡	ReadLongExtraValue (see page 1902)	
≡	ReadValue (see page 1902)	
≡	SetNames (see page 1903)	
≡	WriteDoubleExtraValue (see page 1903)	
≡	WriteLongExtraValue (see page 1903)	
≡	WriteValue (see page 1903)	

## XX.3.1 ReadDoubleExtraValue

### C++

```
HRESULT ReadDoubleExtraValue(IRDIMatDefDblExValType _type, double* ret);
```

**C#**

```
public double ReadDoubleExtraValue(IRDimMatDefDblExValType _type);
```

**Visual Basic**

```
Public Function ReadDoubleExtraValue(_type As IRDimMatDefDblExValType) As double
```

**XX.3.2 ReadLongExtraValue****C++**

```
HRESULT ReadLongExtraValue(IRDimMatDefLongExValType _type, long* ret);
```

**C#**

```
public long ReadLongExtraValue(IRDimMatDefLongExValType _type);
```

**Visual Basic**

```
Public Function ReadLongExtraValue(_type As IRDimMatDefLongExValType) As long
```

**XX.3.3 ReadValue****C++**

```
HRESULT ReadValue(IRDimMatDefValType _type, double* ret);
```

**C#**

```
public double ReadValue(IRDimMatDefValType _type);
```

**Visual Basic**

```
Public Function ReadValue(_type As IRDimMatDefValType) As double
```

**XX.3.4 SetNames****C++**

```
HRESULT SetNames(BSTR _name, BSTR _second_name);
```

**C#**

```
public void SetNames(String _name, String _second_name);
```

**Visual Basic**

```
Public Sub SetNames(_name As String, _second_name As String)
```

**XX.3.5 WriteDoubleExtraValue****C++**

```
HRESULT WriteDoubleExtraValue(IRDimMatDefDblExValType _type, double _val);
```

**C#**

```
public void WriteDoubleExtraValue(IRDimMatDefDblExValType _type, double _val);
```

**Visual Basic**

```
Public Sub WriteDoubleExtraValue(_type As IRDimMatDefDblExValType, _val As double)
```

### XX.3.6 WriteLongExtraValue

**C++**

```
HRESULT WriteLongExtraValue(IRDimMatDefLongExValType _type, long _val);
```

**C#**

```
public void WriteLongExtraValue(IRDimMatDefLongExValType _type, long _val);
```

**Visual Basic**

```
Public Sub WriteLongExtraValue(_type As IRDimMatDefLongExValType, _val As long)
```

### XX.3.7 WriteValue

**C++**

```
HRESULT WriteValue(IRDimMatDefValType _type, double _val);
```

**C#**

```
public void WriteValue(IRDimMatDefValType _type, double _val);
```

**Visual Basic**

```
Public Sub WriteValue(_type As IRDimMatDefValType, _val As double)
```

## XXI IRDimProfDefType

**C++**

```
enum IRDimProfDefType;
```

**C#**

```
public enum IRDimProfDefType;
```

**Visual Basic**

```
Public Enum IRDimProfDefType
```

## XXII IRDimProfDefItemType

**C++**

```
enum IRDimProfDefItemType;
```

**C#**

```
public enum IRDimProfDefItemType;
```

**Visual Basic**

```
Public Enum IRDimProfDefItemType
```

## XXIII IRDimProfDefValType

**C++**

```
enum IRDimProfDefValType;
```

**C#**

```
public enum IRDimProfDefValType;
```

**Visual Basic**

```
Public Enum IRDimProfDefValType
```

## XXIV IRDimProfDef

**Class Hierarchy****C++**

```
interface IRDimProfDef : IDispatch;
```

**C#**

```
public interface IRDimProfDef;
```

**Visual Basic**

```
Public Interface IRDimProfDef
```

### XXIV.1 IRDimProfDef Members

The following tables list the members exposed by IRDimProfDef.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	IsVar (↗ see page 1905)	
◆	Name (↗ see page 1905)	
◆	Type (↗ see page 1905)	

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡▼	Clear (↗ see page 1906)	
≡▼	ReadValue (↗ see page 1906)	
≡▼	WriteValue (↗ see page 1906)	

### XXIV.2 IRDimProfDef Fields

The fields of the IRDimProfDef class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	IsVar (↗ see page 1905)	
◆	Name (↗ see page 1905)	
◆	Type (↗ see page 1905)	

#### XXIV.2.1 IsVar

**C++**

```
HRESULT get_IsVar(VARIANT_BOOL*);
```

**C#**

```
public bool IsVar { get; }
```

**Visual Basic**

```
Public ReadOnly IsVar As Boolean
```

**XXIV.2.2 Name****C++**

```
HRESULT get_Name(BSTR*) ;
HRESULT put_Name(BSTR) ;
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**XXIV.2.3 Type****C++**

```
HRESULT get_Type(IRDimProfDefType*) ;
HRESULT put_Type(IRDimProfDefType) ;
```

**C#**

```
public IRDimProfDefType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRDimProfDefType
```

**XXIV.3 IRDimProfDef Methods**

The methods of the IRDimProfDef class are listed here.

**Public Methods**

	Name	Description
≡	Clear (see page 1906)	
≡	ReadValue (see page 1906)	
≡	WriteValue (see page 1906)	

**XXIV.3.1 Clear****C++**

```
HRESULT Clear();
```

**C#**

```
public void Clear();
```

**Visual Basic**

```
Public Sub Clear()
```

**XXIV.3.2 ReadValue****C++**

```
HRESULT ReadValue(IRDimProfDefItemType _item, IRDimProfDefValType _type, double* ret);
```

**C#**

```
public double ReadValue(IRDimProfDefItemType _item, IRDimProfDefValType _type);
```

**Visual Basic**

```
Public Function ReadValue(_item As IRDimProfDefItemType, _type As IRDimProfDefValType) As
double
```

**XXIV.3.3 WriteValue****C++**

```
HRESULT WriteValue(IRDimProfDefItemType _item, IRDimProfDefValType _type, double _val);
```

**C#**

```
public void WriteValue(IRDimProfDefItemType _item, IRDimProfDefValType _type, double _val);
```

**Visual Basic**

```
Public Sub WriteValue(_item As IRDimProfDefItemType, _type As IRDimProfDefValType, _val As
double)
```

**XXV IRDimCalcStateFlagType****C++**

```
enum IRDimCalcStateFlagType;
```

**C#**

```
public enum IRDimCalcStateFlagType;
```

**Visual Basic**

```
Public Enum IRDimCalcStateFlagType
```

**XXVI IRDimCalcStateParamType****C++**

```
enum IRDimCalcStateParamType;
```

**C#**

```
public enum IRDimCalcStateParamType;
```

**Visual Basic**

```
Public Enum IRDimCalcStateParamType
```

**XXVII IRDimCalcStateParamValue****C++**

```
enum IRDimCalcStateParamValue;
```

**C#**

```
public enum IRDimCalcStateParamValue;
```

**Visual Basic**

```
Public Enum IRDimCalcStateParamValue
```

## XXVIII IRDimCalcValueType

### C++

```
enum IRDimCalcValueType;
```

### C#

```
public enum IRDimCalcValueType;
```

### Visual Basic

```
Public Enum IRDimCalcValueType
```

## XXIX IRDimCalcState

### Class Hierarchy

### C++

```
interface IRDimCalcState : IDispatch;
```

### C#

```
public interface IRDimCalcState;
```

### Visual Basic

```
Public Interface IRDimCalcState
```

## XXIX.1 IRDimCalcState Members

The following tables list the members exposed by IRDimCalcState.

### Public Methods

	Name	Description
≡	GetParam (see page 1908)	
≡	GetParamValue (see page 1908)	
≡	IsFlagSet (see page 1909)	
≡	SetFlag (see page 1909)	
≡	SetParam (see page 1909)	
≡	SetParamValue (see page 1909)	

## XXIX.2 IRDimCalcState Methods

The methods of the IRDimCalcState class are listed here.

### Public Methods

	Name	Description
≡	GetParam (see page 1908)	
≡	GetParamValue (see page 1908)	
≡	IsFlagSet (see page 1909)	
≡	SetFlag (see page 1909)	
≡	SetParam (see page 1909)	
≡	SetParamValue (see page 1909)	

## XXIX.2.1 GetParam

**C++**

```
HRESULT GetParam(IRDimCalcStateParamType _type, IRDimCalcStateParamValue* ret);
```

**C#**

```
public IRDimCalcStateParamValue GetParam(IRDimCalcStateParamType _type);
```

**Visual Basic**

```
Public Function GetParam(_type As IRDimCalcStateParamType) As IRDimCalcStateParamValue
```

## XXIX.2.2 GetParamValue

**C++**

```
HRESULT GetParamValue(IRDimCalcStateValueType _type, double* ret);
```

**C#**

```
public double GetParamValue(IRDimCalcStateValueType _type);
```

**Visual Basic**

```
Public Function GetParamValue(_type As IRDimCalcStateValueType) As double
```

## XXIX.2.3 IsFlagSet

**C++**

```
HRESULT IsFlagSet(IRDimCalcStateFlagType _type, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsFlagSet(IRDimCalcStateFlagType _type);
```

**Visual Basic**

```
Public Function IsFlagSet(_type As IRDimCalcStateFlagType) As Boolean
```

## XXIX.2.4 SetFlag

**C++**

```
HRESULT SetFlag(IRDimCalcStateFlagType _type, VARIANT_BOOL _val);
```

**C#**

```
public void SetFlag(IRDimCalcStateFlagType _type, bool _val);
```

**Visual Basic**

```
Public Sub SetFlag(_type As IRDimCalcStateFlagType, _val As Boolean)
```

## XXIX.2.5 SetParam

**C++**

```
HRESULT SetParam(IRDimCalcStateParamType _type, IRDimCalcStateParamValue _val);
```

**C#**

```
public void SetParam(IRDimCalcStateParamType _type, IRDimCalcStateParamValue _val);
```

**Visual Basic**

```
Public Sub SetParam(_type As IRDimCalcStateParamType, _val As IRDimCalcStateParamValue)
```

**XXIX.2.6 SetParamValue****C++**

```
HRESULT SetParamValue(IRDimCalcStateValueType _type, double _val);
```

**C#**

```
public void SetParamValue(IRDimCalcStateValueType _type, double _val);
```

**Visual Basic**

```
Public Sub SetParamValue(_type As IRDimCalcStateValueType, _val As double)
```

**XXX IRDimMembSrv****Class Hierarchy****C++**

```
interface IRDimMembSrv : IDispatch;
```

**C#**

```
public interface IRDimMembSrv;
```

**Visual Basic**

```
Public Interface IRDimMembSrv
```

**XXX.1 IRDimMembSrv Members**

The following tables list the members exposed by IRDimMembSrv.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	CheckLabelName (↗ see page 1910)	
≡	Save (↗ see page 1910)	

**XXX.2 IRDimMembSrv Methods**

The methods of the IRDimMembSrv class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
≡	CheckLabelName (↗ see page 1910)	
≡	Save (↗ see page 1910)	

**XXX.2.1 CheckLabelName****C++**

```
HRESULT CheckLabelName(BSTR _name, bool* _is_defined, bool* _can_be_saved);
```

**C#**

```
public void CheckLabelName(String _name, bool* _is_defined, bool* _can_be_saved);
```

**Visual Basic**

```
Public Sub CheckLabelName(_name As String, ByRef _is_defined As bool*, ByRef _can_be_saved As bool*)
```

**XXX.2.2 Save****C++**

```
HRESULT Save(IRDimMembDef* _val);
```

**C#**

```
public void Save(IRDimMembDef _val);
```

**Visual Basic**

```
Public Sub Save(ByRef _val As IRDimMembDef)
```

**XXXI IRDimMembCalcRetValue****C++**

```
enum IRDimMembCalcRetValue;
```

**C#**

```
public enum IRDimMembCalcRetValue;
```

**Visual Basic**

```
Public Enum IRDimMembCalcRetValue
```

**XXXII IRDimMembCalcBuckType****C++**

```
enum IRDimMembCalcBuckType;
```

**C#**

```
public enum IRDimMembCalcBuckType;
```

**Visual Basic**

```
Public Enum IRDimMembCalcBuckType
```

**XXXIII IRDimMembResTableLineType****C++**

```
enum IRDimMembResTableLineType;
```

**C#**

```
public enum IRDimMembResTableLineType;
```

**Visual Basic**

```
Public Enum IRDimMembResTableLineType
```

## XXXIV IRDimMembResTableComp

### C++

```
enum IRDimMembResTableComp;
```

### C#

```
public enum IRDimMembResTableComp;
```

### Visual Basic

```
Public Enum IRDimMembResTableComp
```

## XXXV IRDimMembRes

### Class Hierarchy

### C++

```
interface IRDimMembRes : IDispatch;
```

### C#

```
public interface IRDimMembRes;
```

### Visual Basic

```
Public Interface IRDimMembRes
```

## XXXV.1 IRDimMembRes Members

The following tables list the members exposed by IRDimMembRes.

### Public Fields

	Name	Description
◆	EffRatio ( <a href="#">see page 1913</a> )	
◆	Language ( <a href="#">see page 1913</a> )	
◆	Ratio ( <a href="#">see page 1913</a> )	
◆	ResOfCalc ( <a href="#">see page 1914</a> )	
◆	RtfFileName ( <a href="#">see page 1914</a> )	
◆	SlendY ( <a href="#">see page 1914</a> )	
◆	SlendZ ( <a href="#">see page 1914</a> )	
◆	Units ( <a href="#">see page 1914</a> )	

### Public Methods

	Name	Description
≡◆	BlockCount ( <a href="#">see page 1915</a> )	
≡◆	CreateResWnd ( <a href="#">see page 1916</a> )	
≡◆	GetEffDefAccess ( <a href="#">see page 1916</a> )	
≡◆	GetLineComponent ( <a href="#">see page 1916</a> )	
≡◆	GetLineType ( <a href="#">see page 1916</a> )	
≡◆	GetMatDefAccess ( <a href="#">see page 1916</a> )	
≡◆	GetMaxLineNo ( <a href="#">see page 1917</a> )	

	GetMembDefAccess (see page 1917)	
	GetProfDefAccess (see page 1917)	
	IsLineActive (see page 1917)	
	IsStatement (see page 1917)	
	RecognizedPQ (see page 1918)	
	RefreshUnits (see page 1918)	
	ReplaceMark (see page 1918)	
	Retrieve (see page 1918)	
	SetEffDef (see page 1918)	
	SetMatDef (see page 1919)	
	SetMembDef (see page 1919)	
	SetProfDef (see page 1919)	

## XXXV.2 IRDimMembRes Fields

The fields of the IRDimMembRes class are listed here.

### Public Fields

	Name	Description
	EffRatio (see page 1913)	
	Language (see page 1913)	
	Ratio (see page 1913)	
	ResOfCalc (see page 1914)	
	RtfFileName (see page 1914)	
	SlendY (see page 1914)	
	SlendZ (see page 1914)	
	Units (see page 1914)	

### XXXV.2.1 EffRatio

#### C++

```
HRESULT get_EffRatio(double*);
```

#### C#

```
public double EffRatio { get; }
```

#### Visual Basic

```
Public ReadOnly EffRatio As Double
```

### XXXV.2.2 Language

#### C++

```
HRESULT get_Language(long*);  
HRESULT put_Language(long);
```

#### C#

```
public long Language { get; set; }
```

#### Visual Basic

```
Public Language As Long
```

### XXXV.2.3 Ratio

**C++**

```
HRESULT get_Ratio(double*);
```

**C#**

```
public double Ratio { get; }
```

**Visual Basic**

```
Public ReadOnly Ratio As double
```

### XXXV.2.4 ResOfCalc

**C++**

```
HRESULT get_ResOfCalc(IRDimMembCalcRetValue*);
```

**C#**

```
public IRDimMembCalcRetValue ResOfCalc { get; }
```

**Visual Basic**

```
Public ReadOnly ResOfCalc As IRDimMembCalcRetValue
```

### XXXV.2.5 RtfFileName

**C++**

```
HRESULT get_RtfFileName(BSTR*);
```

**C#**

```
public String RtfFileName { get; }
```

**Visual Basic**

```
Public ReadOnly RtfFileName As String
```

### XXXV.2.6 SlendY

**C++**

```
HRESULT get_SlendY(double*);
```

**C#**

```
public double SlendY { get; }
```

**Visual Basic**

```
Public ReadOnly SlendY As double
```

### XXXV.2.7 SlendZ

**C++**

```
HRESULT get_SlendZ(double*);
```

**C#**

```
public double SlendZ { get; }
```

**Visual Basic**

```
Public ReadOnly SlendZ As double
```

## XXXV.2.8 Units

### C++

```
HRESULT get_Units(IRDimUnits**);
HRESULT put_Units(IRDimUnits*);
```

### C#

```
public IRDimUnits Units { get; set; }
```

### Visual Basic

```
Public Units As IRDimUnits
```

## XXXV.3 IRDimMembRes Methods

The methods of the IRDimMembRes class are listed here.

### Public Methods

	Name	Description
»	BlockCount ( [ see page 1915) )	
»	CreateResWnd ( [ see page 1916) )	
»	GetEffDefAccess ( [ see page 1916) )	
»	GetLineComponent ( [ see page 1916) )	
»	GetLineType ( [ see page 1916) )	
»	GetMatDefAccess ( [ see page 1916) )	
»	GetMaxLineNo ( [ see page 1917) )	
»	GetMembDefAccess ( [ see page 1917) )	
»	GetProfDefAccess ( [ see page 1917) )	
»	IsLineActive ( [ see page 1917) )	
»	IsStatement ( [ see page 1917) )	
»	RecognizedPQ ( [ see page 1918) )	
»	RefreshUnits ( [ see page 1918) )	
»	ReplaceMark ( [ see page 1918) )	
»	Retrieve ( [ see page 1918) )	
»	SetEffDef ( [ see page 1918) )	
»	SetMatDef ( [ see page 1919) )	
»	SetMembDef ( [ see page 1919) )	
»	SetProfDef ( [ see page 1919) )	

## XXXV.3.1 BlockCount

### C++

```
HRESULT BlockCount(BSTR _name, long* ret);
```

### C#

```
public long BlockCount(String _name);
```

### Visual Basic

```
Public Function BlockCount(_name As String) As long
```

### XXXV.3.2 CreateResWnd

**C++**

```
HRESULT CreateResWnd(long _parent_hwnd, long* ret);
```

**C#**

```
public long CreateResWnd(long _parent_hwnd);
```

**Visual Basic**

```
Public Function CreateResWnd(_parent_hwnd As long) As long
```

### XXXV.3.3 GetEffDefAccess

**C++**

```
HRESULT GetEffDefAccess(IRDimEffDef** ret);
```

**C#**

```
public IRDimEffDef GetEffDefAccess();
```

**Visual Basic**

```
Public Function GetEffDefAccess() As IRDimEffDef
```

### XXXV.3.4 GetLineComponent

**C++**

```
HRESULT GetLineComponent(long _line_no, IRDimMembResTableComp _cmpnt_no, BSTR* ret);
```

**C#**

```
public String GetLineComponent(long _line_no, IRDimMembResTableComp _cmpnt_no);
```

**Visual Basic**

```
Public Function GetLineComponent(_line_no As long, _cmpnt_no As IRDimMembResTableComp) As String
```

### XXXV.3.5 GetLineType

**C++**

```
HRESULT GetLineType(long _line_no, IRDimMembResTableLineType* ret);
```

**C#**

```
public IRDimMembResTableLineType GetLineType(long _line_no);
```

**Visual Basic**

```
Public Function GetLineType(_line_no As long) As IRDimMembResTableLineType
```

### XXXV.3.6 GetMatDefAccess

**C++**

```
HRESULT GetMatDefAccess(IRDimMatDef** ret);
```

**C#**

```
public IRDimMatDef GetMatDefAccess();
```

**Visual Basic**

```
Public Function GetMatDefAccess() As IRDimMatDef
```

**XXXV.3.7 GetMaxLineNo****C++**

```
HRESULT GetMaxLineNo(long* ret);
```

**C#**

```
public long GetMaxLineNo();
```

**Visual Basic**

```
Public Function GetMaxLineNo() As long
```

**XXXV.3.8 GetMembDefAccess****C++**

```
HRESULT GetMembDefAccess(IRDimMembDef** ret);
```

**C#**

```
public IRDimMembDef GetMembDefAccess();
```

**Visual Basic**

```
Public Function GetMembDefAccess() As IRDimMembDef
```

**XXXV.3.9 GetProfDefAccess****C++**

```
HRESULT GetProfDefAccess(IRDimProfDef** ret);
```

**C#**

```
public IRDimProfDef GetProfDefAccess();
```

**Visual Basic**

```
Public Function GetProfDefAccess() As IRDimProfDef
```

**XXXV.3.10 IsLineActive****C++**

```
HRESULT IsLineActive(long _line_no, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsLineActive(long _line_no);
```

**Visual Basic**

```
Public Function IsLineActive(_line_no As long) As Boolean
```

**XXXV.3.11 IsStatement****C++**

```
HRESULT IsStatement(BSTR _name, VARIANT_BOOL* ret);
```

**C#**

```
public bool IsStatement(String _name);
```

**Visual Basic**

```
Public Function IsStatement(_name As String) As Boolean
```

**XXXV.3.12 RecognizedPQ****C++**

```
HRESULT RecognizedPQ(BSTR _name, BSTR _vstr, BSTR _ustr, VARIANT_BOOL* ret);
```

**C#**

```
public bool RecognizedPQ(String _name, String _vstr, String _ustr);
```

**Visual Basic**

```
Public Function RecognizedPQ(_name As String, ByRef _vstr As String, ByRef _ustr As String) As Boolean
```

**XXXV.3.13 RefreshUnits****C++**

```
HRESULT RefreshUnits(VARIANT_BOOL _are_robot_units);
```

**C#**

```
public void RefreshUnits(bool _are_robot_units);
```

**Visual Basic**

```
Public Sub RefreshUnits(_are_robot_units As Boolean)
```

**XXXV.3.14 ReplaceMark****C++**

```
HRESULT ReplaceMark(BSTR _mark, BSTR _mark_out, VARIANT_BOOL* ret);
```

**C#**

```
public bool ReplaceMark(String _mark, String _mark_out);
```

**Visual Basic**

```
Public Function ReplaceMark(_mark As String, ByRef _mark_out As String) As Boolean
```

**XXXV.3.15 Retrieve****C++**

```
HRESULT Retrieve(IRDimStream* _str);
```

**C#**

```
public void Retrieve(IRDimStream _str);
```

**Visual Basic**

```
Public Sub Retrieve(ByRef _str As IRDimStream)
```

**XXXV.3.16 SetEffDef****C++**

```
HRESULT SetEffDef(IRDimEffDef* _eff_def);
```

**C#**

```
public void SetEffDef(IRDimEffDef _eff_def);
```

**Visual Basic**

```
Public Sub SetEffDef(ByRef _eff_def As IRDimEffDef)
```

**XXXV.3.17 SetMatDef****C++**

```
HRESULT SetMatDef(IRDimMatDef* _mat_def);
```

**C#**

```
public void SetMatDef(IRDimMatDef _mat_def);
```

**Visual Basic**

```
Public Sub SetMatDef(ByRef _mat_def As IRDimMatDef)
```

**XXXV.3.18 SetMembDef****C++**

```
HRESULT SetMembDef(IRDimMembDef* _memb_def);
```

**C#**

```
public void SetMembDef(IRDimMembDef _memb_def);
```

**Visual Basic**

```
Public Sub SetMembDef(ByRef _memb_def As IRDimMembDef)
```

**XXXV.3.19 SetProfDef****C++**

```
HRESULT SetProfDef(IRDimProfDef* _prof_def);
```

**C#**

```
public void SetProfDef(IRDimProfDef _prof_def);
```

**Visual Basic**

```
Public Sub SetProfDef(ByRef _prof_def As IRDimProfDef)
```

**XXXVI IRDimMembCalc****Class Hierarchy****C++**

```
interface IRDimMembCalc : IDispatch;
```

**C#**

```
public interface IRDimMembCalc;
```

**Visual Basic**

```
Public Interface IRDimMembCalc
```

## XXXVI.1 IRDimMembCalc Members

The following tables list the members exposed by IRDimMembCalc.

### Public Fields

	Name	Description
◆	IsExtraMomentsSet1 (see page 1920)	
◆	IsExtraMomentsSet2 (see page 1921)	
◆	IsIntPointsMomentsSet1 (see page 1921)	
◆	IsIntPointsMomentsSet2 (see page 1921)	

### Public Methods

	Name	Description
≡	CalculBuckling (see page 1922)	
≡	CalculMember (see page 1922)	
≡	GetRatio (see page 1922)	
≡	GetResultsInterface (see page 1922)	
≡	SetCalcState (see page 1922)	
≡	SetEffDef (see page 1923)	
≡	SetMatDef (see page 1923)	
≡	SetMembDef (see page 1923)	
≡	SetProfDef (see page 1923)	

## XXXVI.2 IRDimMembCalc Fields

The fields of the IRDimMembCalc class are listed here.

### Public Fields

	Name	Description
◆	IsExtraMomentsSet1 (see page 1920)	
◆	IsExtraMomentsSet2 (see page 1921)	
◆	IsIntPointsMomentsSet1 (see page 1921)	
◆	IsIntPointsMomentsSet2 (see page 1921)	

### XXXVI.2.1 IsExtraMomentsSet1

#### C++

```
HRESULT get_IsExtraMomentsSet1(VARIANT_BOOL* );
```

#### C#

```
public bool IsExtraMomentsSet1 { get; }
```

#### Visual Basic

```
Public ReadOnly IsExtraMomentsSet1 As Boolean
```

## XXXVI.2.2 IsExtraMomentsSet2

**C++**

```
HRESULT get_IsExtraMomentsSet2(VARIANT_BOOL* );
```

**C#**

```
public bool IsExtraMomentsSet2 { get; }
```

**Visual Basic**

```
Public ReadOnly IsExtraMomentsSet2 As Boolean
```

## XXXVI.2.3 IsIntPointsMomentsSet1

**C++**

```
HRESULT get_IsIntPointsMomentsSet1(VARIANT_BOOL* );
```

**C#**

```
public bool IsIntPointsMomentsSet1 { get; }
```

**Visual Basic**

```
Public ReadOnly IsIntPointsMomentsSet1 As Boolean
```

## XXXVI.2.4 IsIntPointsMomentsSet2

**C++**

```
HRESULT get_IsIntPointsMomentsSet2(VARIANT_BOOL* );
```

**C#**

```
public bool IsIntPointsMomentsSet2 { get; }
```

**Visual Basic**

```
Public ReadOnly IsIntPointsMomentsSet2 As Boolean
```

## XXXVI.3 IRDimMembCalc Methods

The methods of the IRDimMembCalc class are listed here.

### Public Methods

	Name	Description
ⓘ	CalculBuckling ( ⓘ see page 1922)	
ⓘ	CalculMember ( ⓘ see page 1922)	
ⓘ	GetRatio ( ⓘ see page 1922)	
ⓘ	GetResultsInterface ( ⓘ see page 1922)	
ⓘ	SetCalcState ( ⓘ see page 1922)	
ⓘ	SetEffDef ( ⓘ see page 1923)	
ⓘ	SetMatDef ( ⓘ see page 1923)	
ⓘ	SetMembDef ( ⓘ see page 1923)	
ⓘ	SetProfDef ( ⓘ see page 1923)	

## XXXVI.3.1 CalculBuckling

**C++**

```
HRESULT CalculBuckling();
```

**C#**

```
public void CalculBuckling();
```

**Visual Basic**

```
Public Sub CalculBuckling()
```

**XXXVI.3.2 CalculMember****C++**

```
HRESULT CalculMember(IRDimMembCalcRetValue* ret);
```

**C#**

```
public IRDimMembCalcRetValue CalculMember();
```

**Visual Basic**

```
Public Function CalculMember() As IRDimMembCalcRetValue
```

**XXXVI.3.3 GetRatio****C++**

```
HRESULT GetRatio(double* ret);
```

**C#**

```
public double GetRatio();
```

**Visual Basic**

```
Public Function GetRatio() As double
```

**XXXVI.3.4 GetResultsInterface****C++**

```
HRESULT GetResultsInterface(IRDimMembRes** ret);
```

**C#**

```
public IRDimMembRes GetResultsInterface();
```

**Visual Basic**

```
Public Function GetResultsInterface() As IRDimMembRes
```

**XXXVI.3.5 SetCalcState****C++**

```
HRESULT SetCalcState(IRDimCalcState* _st);
```

**C#**

```
public void SetCalcState(IRDimCalcState _st);
```

**Visual Basic**

```
Public Sub SetCalcState(ByRef _st As IRDimCalcState)
```

**XXXVI.3.6 SetEffDef****C++**

```
HRESULT SetEffDef(IRDimEffDef* _eff_data);
```

**C#**

```
public void SetEffDef(IRDimEffDef _eff_data);
```

**Visual Basic**

```
Public Sub SetEffDef(ByRef _eff_data As IRDimEffDef)
```

**XXXVI.3.7 SetMatDef****C++**

```
HRESULT SetMatDef(IRDimMatDef* _mat_def);
```

**C#**

```
public void SetMatDef(IRDimMatDef _mat_def);
```

**Visual Basic**

```
Public Sub SetMatDef(ByRef _mat_def As IRDimMatDef)
```

**XXXVI.3.8 SetMembDef****C++**

```
HRESULT SetMembDef(IRDimMembDef* _memb_def);
```

**C#**

```
public void SetMembDef(IRDimMembDef _memb_def);
```

**Visual Basic**

```
Public Sub SetMembDef(ByRef _memb_def As IRDimMembDef)
```

**XXXVI.3.9 SetProfDef****C++**

```
HRESULT SetProfDef(IRDimProfDef* _prof_def);
```

**C#**

```
public void SetProfDef(IRDimProfDef _prof_def);
```

**Visual Basic**

```
Public Sub SetProfDef(ByRef _prof_def As IRDimProfDef)
```

**XXXVII IRDimCodeService****Class Hierarchy****C++**

```
interface IRDimCodeService : IDispatch;
```

**C#**

```
public interface IRDimCodeService;
```

**Visual Basic**

```
Public Interface IRDimCodeService
```

## XXXVII.1 IRDimCodeService Members

The following tables list the members exposed by IRDimCodeService.

### Public Methods

	Name	Description
≡	EditMembDef (see page 1924)	
≡	GetDefaultMembDef (see page 1924)	
≡	GetMembCalc (see page 1925)	

## XXXVII.2 IRDimCodeService Methods

The methods of the IRDimCodeService class are listed here.

### Public Methods

	Name	Description
≡	EditMembDef (see page 1924)	
≡	GetDefaultMembDef (see page 1924)	
≡	GetMembCalc (see page 1925)	

### XXXVII.2.1 EditMembDef

#### C++

```
HRESULT EditMembDef(long _language, VARIANT_BOOL _inModalWindow, long _parentHWND,
IRDimMembDef* _val);
```

#### C#

```
public void EditMembDef(long _language, bool _inModalWindow, long _parentHWND, IRDimMembDef
_val);
```

#### Visual Basic

```
Public Sub EditMembDef(_language As long, _inModalWindow As Boolean, _parentHWND As long,
ByRef _val As IRDimMembDef)
```

### XXXVII.2.2 GetDefaultMembDef

#### C++

```
HRESULT GetDefaultMembDef(IRDimMembDefType _type, IRDimMembDef** ret);
```

#### C#

```
public IRDimMembDef GetDefaultMembDef(IRDimMembDefType _type);
```

#### Visual Basic

```
Public Function GetDefaultMembDef(_type As IRDimMembDefType) As IRDimMembDef
```

### XXXVII.2.3 GetMembCalc

#### C++

```
HRESULT GetMembCalc(IRDimMembCalc** ret);
```

#### C#

```
public IRDimMembCalc GetMembCalc();
```

**Visual Basic**

```
Public Function GetMembCalc() As IRDimMembCalc
```

## XXXVIII IRDimClient

**Class Hierarchy****C++**

```
interface IRDimClient : IDispatch;
```

**C#**

```
public interface IRDimClient;
```

**Visual Basic**

```
Public Interface IRDimClient
```

### XXXVIII.1 IRDimClient Members

The following tables list the members exposed by IRDimClient.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetRDimCodeService (see page 1925)	

### XXXVIII.2 IRDimClient Methods

The methods of the IRDimClient class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
	GetRDimCodeService (see page 1925)	

#### XXXVIII.2.1 GetRDimCodeService

**C++**

```
HRESULT GetRDimCodeService(IRDimCodeService** ret);
```

**C#**

```
public IRDimCodeService GetRDimCodeService();
```

**Visual Basic**

```
Public Function GetRDimCodeService() As IRDimCodeService
```

## XXXIX IRDimUnitType

**C++**

```
enum IRDimUnitType;
```

**C#**

```
public enum IRDimUnitType;
```

**Visual Basic**

```
Public Enum IRDimUnitType
```

## XL IRDimUnits

**Class Hierarchy****C++**

```
interface IRDimUnits : IDispatch;
```

**C#**

```
public interface IRDimUnits;
```

**Visual Basic**

```
Public Interface IRDimUnits
```

### XL.1 IRDimUnits Members

The following tables list the members exposed by IRDimUnits.

**Public Fields**

	Name	Description
◆	AreRobotUnits ( <a href="#">see page 1927</a> )	

**Public Methods**

	Name	Description
≡◆	Format ( <a href="#">see page 1927</a> )	
≡◆	ReadToUserCoef ( <a href="#">see page 1927</a> )	
≡◆	ReadUserName ( <a href="#">see page 1928</a> )	
≡◆	Refresh ( <a href="#">see page 1928</a> )	

### XL.2 IRDimUnits Fields

The fields of the IRDimUnits class are listed here.

**Public Fields**

	Name	Description
◆	AreRobotUnits ( <a href="#">see page 1927</a> )	

#### XL.2.1 AreRobotUnits

**C++**

```
HRESULT get_AreRobotUnits(VARIANT_BOOL* );
```

**C#**

```
public bool AreRobotUnits { get; }
```

**Visual Basic**

```
Public ReadOnly AreRobotUnits As Boolean
```

## XL.3 IRDimUnits Methods

The methods of the IRDimUnits class are listed here.

### Public Methods

	Name	Description
♫	Format (see page 1927)	
♫	ReadToUserCoef (see page 1927)	
♫	ReadUserName (see page 1928)	
♫	Refresh (see page 1928)	

### XL.3.1 Format

#### C++

```
HRESULT Format(IRDimUnitType _unit_type, double _val, BSTR* ret);
```

#### C#

```
public String Format(IRDimUnitType _unit_type, double _val);
```

#### Visual Basic

```
Public Function Format(_unit_type As IRDimUnitType, _val As double) As String
```

### XL.3.2 ReadToUserCoef

#### C++

```
HRESULT ReadToUserCoef(IRDimUnitType _unit_type, double* ret);
```

#### C#

```
public double ReadToUserCoef(IRDimUnitType _unit_type);
```

#### Visual Basic

```
Public Function ReadToUserCoef(_unit_type As IRDimUnitType) As double
```

### XL.3.3 ReadUserName

#### C++

```
HRESULT ReadUserName(IRDimUnitType _unit_type, BSTR* ret);
```

#### C#

```
public String ReadUserName(IRDimUnitType _unit_type);
```

#### Visual Basic

```
Public Function ReadUserName(_unit_type As IRDimUnitType) As String
```

### XL.3.4 Refresh

#### C++

```
HRESULT Refresh(VARIANT_BOOL _are_robot_units);
```

#### C#

```
public void Refresh(bool _are_robot_units);
```

**Visual Basic**

```
Public Sub Refresh(_are_robot_units As Boolean)
```

## XLI IRDimServer

**Class Hierarchy****C++**

```
interface IRDimServer : IDispatch;
```

**C#**

```
public interface IRDimServer;
```

**Visual Basic**

```
Public Interface IRDimServer
```

**Description**

Server responsible for design and verification of steel and timber members.

**Version**

Available since version 2.5.

## Member section definition module

Names of interfaces from the module of bar section definition start with the RSect prefix.

**Version**

Available since version 3.

## Connection module

Names of interfaces defined for the connection module begin with the RJoint prefix.

**Notes**

All values described by interfaces of a connection module are given in millimeters.

**Enumerations**

	<b>Name</b>	<b>Description</b>
	IRJointConnectionDefType ( <a href="#">see page 2144</a> )	A method of connection definition.
	IRJointConnectionType ( <a href="#">see page 2144</a> )	Types of connections available in a connections module of Robot.
	IRJointWebFlangeRelativePos ( <a href="#">see page 2147</a> )	Possible types of a beam - column mutual position. .
	IRJointBoltType ( <a href="#">see page 2157</a> )	Available bolt types.
	IRJointWebType ( <a href="#">see page 2157</a> )	Types of reinforcing plates.
	IRJointLoadType ( <a href="#">see page 2167</a> )	
	IRJointAngleUnit ( <a href="#">see page 2167</a> )	
	IRJointExtType ( <a href="#">see page 2167</a> )	

	IRJointComponentType (see page 2168)	
-----------------------------------------------------------------------------------	--------------------------------------	--

## Interfaces

	Name	Description
	IRJointConnectionInfo (see page 2145)	Information about connection..
	IRJointConnection (see page 2147)	A data type describing common features for all connections that can be defined in Robot.
	IRJointWeld (see page 2149)	Description of a weld.
	IRJointPlate (see page 2151)	Plate description.
	IRJointBolts (see page 2153)	Definition of a group of bolts with common parameters.
	IRJointConnectionServer (see page 2157)	The connection server administers all connection defined in the project. Each connection is associated with a number. The numbering of connections may be discontinuous.
	IRJointProfile (see page 2162)	Structure describing a single connection component. .
	IRJointLoad (see page 2165)	Interface describing a load applied to the connection. .

## I Knee connection

### Enumerations

	Name	Description
	IRJointKneeType (see page 1941)	Accessible types of a knee connection.
	IRJointKneeFixType (see page 1941)	Available types of a knee connection as for joint type.
	IRJointKneeReinfType (see page 1941)	Available bracket types for a knee connection.
	IRJointKneeWebStiffType (see page 1955)	Bracket types of a column web for a knee connection. .
	IRJointKneeDiagonalStiffType (see page 1955)	Available slanting bracket types for a column web.
	IRJointKneeMaterials (see page 1960)	

### Interfaces

	Name	Description
	IRJointKneeBolts (see page 1930)	Definition of a bolt group for a knee connection.
	IRJointKneeBracket (see page 1934)	Knee bracket.
	IRJointKneeWebPlate (see page 1938)	Description of a plate stiffening a column web.
	IRJointKnee (see page 1942)	Knee connection.
	IRJointKneeDiagonalStiff (see page 1956)	An inclined stiffener of a column.
	IRJointKneeStiffColumn (see page 1957)	Column bracket in a knee connection. .
	IRJointKneeLoad (see page 1958)	Interface describing a load applied to a knee connection. .

### I.1 IRJointKneeBolts

#### Class Hierarchy

**C++**

```
interface IRJointKneeBolts : IRJointBolts;
```

**C#**

```
public interface IRJointKneeBolts : IRJointBolts;
```

**Visual Basic**

```
Public Interface IRJointKneeBolts
```

**Description**

Definition of a bolt group for a knee connection.

**I.1.1 IRJointKneeBolts Members**

The following tables list the members exposed by IRJointKneeBolts.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Area ( <a href="#">see page 2154</a> )	Bolt section area.
◆	ClassName ( <a href="#">see page 1931</a> )	Bolt class.
◆	Cols ( <a href="#">see page 1932</a> )	Number of bolts in a column.
◆	Diameter ( <a href="#">see page 2155</a> )	Bolt diameter.
◆	DiameterName ( <a href="#">see page 1932</a> )	Bolt diameter.
◆	EqualSpac ( <a href="#">see page 1932</a> )	Bolts are even-spaced if EqualSpac equals 1.
◆	Friction ( <a href="#">see page 2156</a> )	Bolt friction.
◆	Height1 ( <a href="#">see page 1933</a> )	Distance h1.
◆	Rows ( <a href="#">see page 1933</a> )	Number of bolts in a row.
◆	SpacingH ( <a href="#">see page 1933</a> )	Distances between bolts in the horizontal direction.
◆	SpacingV ( <a href="#">see page 1933</a> )	Distances between bolts in the vertical direction.
◆	Symmetry ( <a href="#">see page 1934</a> )	Bolts are centered with respect to the plate if Symmetry equals 1.

**I.1.2 IRJointKneeBolts Fields**

The fields of the IRJointKneeBolts class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	ClassName ( <a href="#">see page 1931</a> )	Bolt class.
◆	Cols ( <a href="#">see page 1932</a> )	Number of bolts in a column.
◆	DiameterName ( <a href="#">see page 1932</a> )	Bolt diameter.
◆	EqualSpac ( <a href="#">see page 1932</a> )	Bolts are even-spaced if EqualSpac equals 1.
◆	Height1 ( <a href="#">see page 1933</a> )	Distance h1.
◆	Rows ( <a href="#">see page 1933</a> )	Number of bolts in a row.
◆	SpacingH ( <a href="#">see page 1933</a> )	Distances between bolts in the horizontal direction.
◆	SpacingV ( <a href="#">see page 1933</a> )	Distances between bolts in the vertical direction.
◆	Symmetry ( <a href="#">see page 1934</a> )	Bolts are centered with respect to the plate if Symmetry equals 1.

**I.1.2.1 ClassName****C++**

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

**C#**

```
public BSTR ClassName { get; set; }
```

**Visual Basic**

```
Public ClassName As BSTR
```

**Description**

Bolt class.

**Version**

Available since version 9.5.

**I.1.2.2 Cols****C++**

```
HRESULT get_Cols(int*);  
HRESULT put_Cols(int);
```

**C#**

```
public int Cols { get; set; }
```

**Visual Basic**

```
Public Cols As int
```

**Description**

Number of bolts in a column.

**Version**

Available since version 9.5.

**I.1.2.3 DiameterName****C++**

```
HRESULT get_DiameterName(BSTR*);  
HRESULT put_DiameterName(BSTR);
```

**C#**

```
public BSTR DiameterName { get; set; }
```

**Visual Basic**

```
Public DiameterName As BSTR
```

**Description**

Bolt diameter.

**Version**

Available since version 9.5.

**I.1.2.4 EqualSpac****C++**

```
HRESULT get_EqualSpac(int*);  
HRESULT put_EqualSpac(int);
```

**C#**

```
public int EqualSpac { get; set; }
```

**Visual Basic**

```
Public EqualSpac As int
```

**Description**

Bolts are even-spaced if EqualSpac equals 1.

**Version**

Available since version 9.5.

**I.1.2.5 Height1****C++**

```
HRESULT get_Height1(double*);  
HRESULT put_Height1(double);
```

**C#**

```
public double Height1 { get; set; }
```

**Visual Basic**

```
Public Height1 As Double
```

**Description**

Distance h1.

**Version**

Available since version 9.5.

**I.1.2.6 Rows****C++**

```
HRESULT get_Rows(int*);  
HRESULT put_Rows(int);
```

**C#**

```
public int Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Integer
```

**Description**

Number of bolts in a row.

**Version**

Available since version 9.5.

**I.1.2.7 SpacingH****C++**

```
HRESULT get_SpacingH(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray SpacingH { get; }
```

**Visual Basic**

```
Public ReadOnly SpacingH As IRobotValuesArray
```

**Description**

Distances between bolts in the horizontal direction.

**I.1.2.8 SpacingV****C++**

```
HRESULT get_SpacingV(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray SpacingV { get; }
```

**Visual Basic**

```
Public ReadOnly SpacingV As IRobotValuesArray
```

**Description**

Distances between bolts in the vertical direction.

**I.1.2.9 Symmetry****C++**

```
HRESULT get_Symmetry(int*);  
HRESULT put_Symmetry(int);
```

**C#**

```
public int Symmetry { get; set; }
```

**Visual Basic**

```
Public Symmetry As Integer
```

**Description**

Bolts are centered with respect to the plate if Symmetry equals 1.

**Version**

Available since version 9.5.

**I.2 IRJointKneeBracket****Class Hierarchy****C++**

```
interface IRJointKneeBracket : IDispatch;
```

**C#**

```
public interface IRJointKneeBracket;
```

**Visual Basic**

```
Public Interface IRJointKneeBracket
```

**Description**

Knee bracket.

**I.2.1 IRJointKneeBracket Members**

The following tables list the members exposed by IRJointKneeBracket.

## Public Fields

	Name	Description
◆	Angle ( [ see page 1935) )	Inclination angle of reinforcement.
◆	Exist ( [ see page 1935) )	Information about a bracket existence.
◆	Height ( [ see page 1936) )	Bracket height.
◆	Length ( [ see page 1936) )	Bracket length.
◆	Material ( [ see page 1936) )	Name of the bracket material .
◆	ThickFlange ( [ see page 1936) )	Thickness of bracket flange .
◆	ThickWeb ( [ see page 1937) )	Thickness of bracket web.
◆	Width ( [ see page 1937) )	Bracket width.

## Public Methods

	Name	Description
◆	AngleExt ( [ see page 1937) )	Angle ( [ see page 1935) ) of bracket inclination, depending on the parameter - in degrees or radians.

## I.2.2 IRJointKneeBracket Fields

The fields of the IRJointKneeBracket class are listed here.

### Public Fields

	Name	Description
◆	Angle ( [ see page 1935) )	Inclination angle of reinforcement.
◆	Exist ( [ see page 1935) )	Information about a bracket existence.
◆	Height ( [ see page 1936) )	Bracket height.
◆	Length ( [ see page 1936) )	Bracket length.
◆	Material ( [ see page 1936) )	Name of the bracket material .
◆	ThickFlange ( [ see page 1936) )	Thickness of bracket flange .
◆	ThickWeb ( [ see page 1937) )	Thickness of bracket web.
◆	Width ( [ see page 1937) )	Bracket width.

### I.2.2.1 Angle

#### C++

```
HRESULT get_Angle(double* );
HRESULT put_Angle(double);
```

#### C#

```
public double Angle { get; set; }
```

#### Visual Basic

```
Public Angle As Double
```

#### Description

Inclination angle of reinforcement.

### I.2.2.2 Exist

#### C++

```
HRESULT get_Exist(VARIANT_BOOL* );
HRESULT put_Exist(VARIANT_BOOL);
```

#### C#

```
public bool Exist { get; set; }
```

**Visual Basic**

```
Public Exist As Boolean
```

**Description**

Information about a bracket existence.

**I.2.2.3 Height****C++**

```
HRESULT get_Height(double*);  
HRESULT put_Height(double);
```

**C#**

```
public double Height { get; set; }
```

**Visual Basic**

```
Public Height As Double
```

**Description**

Bracket height.

**I.2.2.4 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Description**

Bracket length.

**I.2.2.5 Material****C++**

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

**C#**

```
public BSTR Material { get; set; }
```

**Visual Basic**

```
Public Material As BSTR
```

**Description**

Name of the bracket material .

**Version**

Available since version 9.5.

### I.2.2.6 ThickFlange

**C++**

```
HRESULT get_ThickFlange(double* );
HRESULT put_ThickFlange(double);
```

**C#**

```
public double ThickFlange { get; set; }
```

**Visual Basic**

```
Public ThickFlange As double
```

**Description**

Thickness of bracket flange .

### I.2.2.7 ThickWeb

**C++**

```
HRESULT get_ThickWeb(double* );
HRESULT put_ThickWeb(double);
```

**C#**

```
public double ThickWeb { get; set; }
```

**Visual Basic**

```
Public ThickWeb As double
```

**Description**

Thickness of bracket web.

### I.2.2.8 Width

**C++**

```
HRESULT get_Width(double* );
HRESULT put_Width(double);
```

**C#**

```
public double Width { get; set; }
```

**Visual Basic**

```
Public Width As double
```

**Description**

Bracket width.

## I.2.3 IRJointKneeBracket Methods

The methods of the IRJointKneeBracket class are listed here.

### Public Methods

	Name	Description
!	AngleExt ( see page 1937)	Angle ( see page 1935) of bracket inclination, depending on the parameter - in degrees or radians.

### I.2.3.1 AngleExt

#### C++

```
HRESULT AngleExt(IRJointAngleUnit _unit, double* ret);
```

#### C#

```
public double AngleExt(IRJointAngleUnit _unit);
```

#### Visual Basic

```
Public Function AngleExt(_unit As IRJointAngleUnit) As Double
```

#### Description

Angle (see page 1935) of bracket inclination, depending on the parameter - in degrees or radians.

#### Version

Available since version 9.5.

## I.3 IRJointKneeWebPlate

#### Class Hierarchy

#### C++

```
interface IRJointKneeWebPlate : IDispatch;
```

#### C#

```
public interface IRJointKneeWebPlate;
```

#### Visual Basic

```
Public Interface IRJointKneeWebPlate
```

#### Description

Description of a plate stiffening a column web.

### I.3.1 IRJointKneeWebPlate Members

The following tables list the members exposed by IRJointKneeWebPlate.

#### Public Fields

	Name	Description
❖	Bolts (see page 1939)	Bolt group.
❖	Height (see page 1939)	Available since version 1.7.
❖	Material (see page 1939)	Name of the component material.
❖	Thick (see page 1939)	Available since version 1.7.
❖	Type (see page 1940)	
❖	WeldLong (see page 1940)	A longitudinal weld.
❖	WeldTran (see page 1940)	A transversal weld.
❖	Width (see page 1940)	Available since version 1.7.

### I.3.2 IRJointKneeWebPlate Fields

The fields of the IRJointKneeWebPlate class are listed here.

## Public Fields

	Name	Description
◆	Bolts (see page 1939)	Bolt group.
◆	Height (see page 1939)	Available since version 1.7.
◆	Material (see page 1939)	Name of the component material.
◆	Thick (see page 1939)	Available since version 1.7.
◆	Type (see page 1940)	
◆	WeldLong (see page 1940)	A longitudinal weld.
◆	WeldTran (see page 1940)	A transversal weld.
◆	Width (see page 1940)	Available since version 1.7.

### I.3.2.1 Bolts

#### C++

```
HRESULT get_Bolts(IRJointKneeBolts**);
```

#### C#

```
public IRJointKneeBolts Bolts { get; }
```

#### Visual Basic

```
Public ReadOnly Bolts As IRJointKneeBolts
```

#### Description

Bolt group.

### I.3.2.2 Height

#### C++

```
HRESULT get_Height(double*);  
HRESULT put_Height(double);
```

#### C#

```
public double Height { get; set; }
```

#### Visual Basic

```
Public Height As Double
```

#### Description

Available since version 1.7.

### I.3.2.3 Material

#### C++

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

#### C#

```
public BSTR Material { get; set; }
```

#### Visual Basic

```
Public Material As BSTR
```

#### Description

Name of the component material.

## Version

Available since version 9.5.

### I.3.2.4 Thick

#### C++

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

#### C#

```
public double Thick { get; set; }
```

#### Visual Basic

```
Public Thick As Double
```

#### Description

Available since version 1.7.

### I.3.2.5 Type

#### C++

```
HRESULT get_Type(IRJointWebType*);  
HRESULT put_Type(IRJointWebType);
```

#### C#

```
public IRJointWebType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRJointWebType
```

### I.3.2.6 WeldLong

#### C++

```
HRESULT get_WeldLong(IRJointWeld**);
```

#### C#

```
public IRJointWeld WeldLong { get; }
```

#### Visual Basic

```
Public ReadOnly WeldLong As IRJointWeld
```

#### Description

A longitudinal weld.

### I.3.2.7 WeldTran

#### C++

```
HRESULT get_WeldTran(IRJointWeld**);
```

#### C#

```
public IRJointWeld WeldTran { get; }
```

#### Visual Basic

```
Public ReadOnly WeldTran As IRJointWeld
```

#### Description

A transversal weld.

### I.3.2.8 Width

#### C++

```
HRESULT get_Width(double* );
HRESULT put_Width(double);
```

#### C#

```
public double Width { get; set; }
```

#### Visual Basic

```
Public Width As Double
```

#### Description

Available since version 1.7.

## I.4 IRJointKneeType

#### C++

```
enum IRJointKneeType;
```

#### C#

```
public enum IRJointKneeType;
```

#### Visual Basic

```
Public Enum IRJointKneeType
```

#### Members

Members	Description
I_JKT_BEAM2BEAM = 0	Beam-to-beam connection.
I_JKT_BEAM2COLUMN = 1	A frame knee connection.
I_JKT_BEAM2COLUMN_CONTINUE = 2	A beam-to-column connection.

#### Description

Accessible types of a knee connection.

## I.5 IRJointKneeFixType

#### C++

```
enum IRJointKneeFixType;
```

#### C#

```
public enum IRJointKneeFixType;
```

#### Visual Basic

```
Public Enum IRJointKneeFixType
```

#### Description

Available types of a knee connection as for joint type.

## I.6 IRJointKneeReinfType

### C++

```
enum IRJointKneeReinfType;
```

### C#

```
public enum IRJointKneeReinfType;
```

### Visual Basic

```
Public Enum IRJointKneeReinfType
```

### Description

Available bracket types for a knee connection.

## I.7 IRJointKnee

### Class Hierarchy

### C++

```
interface IRJointKnee : IRJointConnection;
```

### C#

```
public interface IRJointKnee : IRJointConnection;
```

### Visual Basic

```
Public Interface IRJointKnee
```

### Description

Knee connection.

### I.7.1 IRJointKnee Members

The following tables list the members exposed by IRJointKnee.

#### Public Fields

	Name	Description
◆	Beam (see page 1945)	Beam section .
◆	Bolts (see page 1945)	A group of connecting bolts.
◆	BracketLow (see page 1945)	Lower bracket of a beam.
◆	BracketUp (see page 1946)	Upper bracket of a beam.
◆	Column (see page 1946)	Column section .
◆	DefComponentMaterial (see page 1946)	Name of the default material of components.
◆	FixType (see page 1946)	Out-of-date parameter - retained to ensure compatibility with the previous version.
◆	FlanPlateLower (see page 1947)	Label of the lower bracket of the column flange.
◆	FlanPlatePosLower (see page 1947)	Lower bracket of the column flange - ebd parameter.
◆	FlanPlatePosUpper (see page 1947)	Upper bracket of the column flange - ebu parameter.
◆	FlanPlateUpper (see page 1948)	Label of the upper bracket of the column flange.
◆	IsBolted (see page 1948)	The connection is bolted if IsBolted equals 1, otherwise, the connection is welded.

◆	IsDefComponentMaterialSet (see page 1948)	All components are assigned the same material if IsDefComponentMaterialSet equals 1.
◆	KneeType (see page 1948)	A type of a knee connection.
◆	Material (see page 1949)	Available since version ARSA 2010.
◆	MaterPlates (see page 1949)	
◆	Plate (see page 1949)	Main plate.
◆	PlatePosition (see page 1949)	Distance between the top edge of the slab and the top edge of the adjoining element.
◆	PlatePositionLow (see page 1950)	Distance between bottom plate edge and the bottom edge of the adjoining element .
◆	StiffDiag (see page 1950)	Inclined column bracket.
◆	StiffLow (see page 1950)	Bottom column bracing.
◆	StiffTypeLow (see page 1951)	A type of lower bracket.
◆	StiffTypeUp (see page 1951)	A type of upper bracket.
◆	StiffUp (see page 1951)	Top column bracing.
◆	TensionPlateLow (see page 1951)	Bottom plate reinforcing the beam.
◆	TensionPlateUp (see page 1952)	Top plate reinforcing the beam.
◆	Type (see page 2148)	A connection type.
◆	VStiffLow (see page 1952)	A bottom, vertical bracing of a beam.
◆	VStiffUp (see page 1952)	A top, vertical bracing of a beam.
◆	WebBeamStiffenerVLower (see page 1952)	Label of the lower vertical stiffener of a beam.
◆	WebBeamStiffenerVUpper (see page 1953)	Label of the upper vertical stiffener of a beam.
◆	WebColumnStiffenerHLower (see page 1953)	Label of the lower horizontal stiffener of a column.
◆	WebColumnStiffenerHUpper (see page 1953)	Label of the upper horizontal stiffener of a column.
◆	WebPlate (see page 1954)	A plate reinforcing a column web.
◆	WebStiffType (see page 1954)	A bracket type of a column web.
◆	WeldBracketDownFlange (see page 1954)	Thickness of the weld connecting the flange of lower bracket with the column.
◆	WeldBracketUpFlange (see page 1954)	Thickness of the weld connecting the flange of upper bracket with the column .
◆	WeldFlange (see page 1955)	
◆	WeldStiff (see page 1955)	
◆	WeldWeb (see page 1955)	
◆	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

## Public Methods

	Name	Description
◆	GetFromRobot (see page 2149)	The function gets connection parameters connection parameters from Robot and fills an object with the data.
◆	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

## I.7.2 IRJointKnee Fields

The fields of the IRJointKnee class are listed here.

### Public Fields

	Name	Description
◆	Beam (see page 1945)	Beam section .

◆	Bolts ( <a href="#">see page 1945</a> )	A group of connecting bolts.
◆	BracketLow ( <a href="#">see page 1945</a> )	Lower bracket of a beam.
◆	BracketUp ( <a href="#">see page 1946</a> )	Upper bracket of a beam.
◆	Column ( <a href="#">see page 1946</a> )	Column section .
◆	DefComponentMaterial ( <a href="#">see page 1946</a> )	Name of the default material of components.
◆	FixType ( <a href="#">see page 1946</a> )	Out-of-date parameter - retained to ensure compatibility with the previous version.
◆	FlanPlateLower ( <a href="#">see page 1947</a> )	Label of the lower bracket of the column flange.
◆	FlanPlatePosLower ( <a href="#">see page 1947</a> )	Lower bracket of the column flange - ebd parameter.
◆	FlanPlatePosUpper ( <a href="#">see page 1947</a> )	Upper bracket of the column flange - ebu parameter.
◆	FlanPlateUpper ( <a href="#">see page 1948</a> )	Label of the upper bracket of the column flange.
◆	IsBolted ( <a href="#">see page 1948</a> )	The connection is bolted if IsBolted equals 1, otherwise, the connection is welded.
◆	IsDefComponentMaterialSet ( <a href="#">see page 1948</a> )	All components are assigned the same material if IsDefComponentMaterialSet equals 1.
◆	KneeType ( <a href="#">see page 1948</a> )	A type of a knee connection.
◆	Material ( <a href="#">see page 1949</a> )	Available since version ARSA 2010.
◆	MaterPlates ( <a href="#">see page 1949</a> )	
◆	Plate ( <a href="#">see page 1949</a> )	Main plate.
◆	PlatePosition ( <a href="#">see page 1949</a> )	Distance between the top edge of the slab and the top edge of the adjoining element.
◆	PlatePositionLow ( <a href="#">see page 1950</a> )	Distance between bottom plate edge and the bottom edge of the adjoining element .
◆	StiffDiag ( <a href="#">see page 1950</a> )	Inclined column bracket.
◆	StiffLow ( <a href="#">see page 1950</a> )	Bottom column bracing.
◆	StiffTypeLow ( <a href="#">see page 1951</a> )	A type of lower bracket.
◆	StiffTypeUp ( <a href="#">see page 1951</a> )	A type of upper bracket.
◆	StiffUp ( <a href="#">see page 1951</a> )	Top column bracing.
◆	TensionPlateLow ( <a href="#">see page 1951</a> )	Bottom plate reinforcing the beam.
◆	TensionPlateUp ( <a href="#">see page 1952</a> )	Top plate reinforcing the beam.
◆	VStiffLow ( <a href="#">see page 1952</a> )	A bottom, vertical bracing of a beam.
◆	VStiffUp ( <a href="#">see page 1952</a> )	A top, vertical bracing of a beam.
◆	WebBeamStiffenerVLower ( <a href="#">see page 1952</a> )	Label of the lower vertical stiffener of a beam.
◆	WebBeamStiffenerVUpper ( <a href="#">see page 1953</a> )	Label of the upper vertical stiffener of a beam.
◆	WebColumnStiffenerHLower ( <a href="#">see page 1953</a> )	Label of the lower horizontal stiffener of a column.
◆	WebColumnStiffenerHUpper ( <a href="#">see page 1953</a> )	Label of the upper horizontal stiffener of a column.
◆	WebPlate ( <a href="#">see page 1954</a> )	A plate reinforcing a column web.
◆	WebStiffType ( <a href="#">see page 1954</a> )	A bracket type of a column web.
◆	WeldBracketDownFlange ( <a href="#">see page 1954</a> )	Thickness of the weld connecting the flange of lower bracket with the column.
◆	WeldBracketUpFlange ( <a href="#">see page 1954</a> )	Thickness of the weld connecting the flange of upper bracket with the column .
◆	WeldFlange ( <a href="#">see page 1955</a> )	
◆	WeldStiff ( <a href="#">see page 1955</a> )	
◆	WeldWeb ( <a href="#">see page 1955</a> )	

### I.7.2.1 Beam

**C++**

```
HRESULT get_Beam(IRJointProfile**);
```

**C#**

```
public IRJointProfile Beam { get; }
```

**Visual Basic**

```
Public ReadOnly Beam As IRJointProfile
```

**Description**

Beam section .

### I.7.2.2 Bolts

**C++**

```
HRESULT get_Bolts(IRJointKneeBolts**);
```

**C#**

```
public IRJointKneeBolts Bolts { get; }
```

**Visual Basic**

```
Public ReadOnly Bolts As IRJointKneeBolts
```

**Description**

A group of connecting bolts.

### I.7.2.3 BracketLow

**C++**

```
HRESULT get_BracketLow(IRJointKneeBracket**);
```

**C#**

```
public IRJointKneeBracket BracketLow { get; }
```

**Visual Basic**

```
Public ReadOnly BracketLow As IRJointKneeBracket
```

**Description**

Lower bracket of a beam.

### I.7.2.4 BracketUp

**C++**

```
HRESULT get_BracketUp(IRJointKneeBracket**);
```

**C#**

```
public IRJointKneeBracket BracketUp { get; }
```

**Visual Basic**

```
Public ReadOnly BracketUp As IRJointKneeBracket
```

**Description**

Upper bracket of a beam.

### I.7.2.5 Column

**C++**

```
HRESULT get_Column( IRJointProfile**);
```

**C#**

```
public IRJointProfile Column { get; }
```

**Visual Basic**

```
Public ReadOnly Column As IRJointProfile
```

**Description**

Column section .

### I.7.2.6 DefComponentMaterial

**C++**

```
HRESULT get_DefComponentMaterial( BSTR*);  
HRESULT put_DefComponentMaterial( BSTR);
```

**C#**

```
public BSTR DefComponentMaterial { get; set; }
```

**Visual Basic**

```
Public DefComponentMaterial As BSTR
```

**Description**

Name of the default material of components.

**Version**

Available since version 9.5.

### I.7.2.7 FixType

**C++**

```
HRESULT get_FixType( IRJointKneeFixType*);  
HRESULT put_FixType( IRJointKneeFixType);
```

**C#**

```
public IRJointKneeFixType FixType { get; set; }
```

**Visual Basic**

```
Public FixType As IRJointKneeFixType
```

**Description**

Out-of-date parameter - retained to ensure compatibility with the previous version.

### I.7.2.8 FlanPlateLower

**C++**

```
HRESULT get_FlanPlateLower( IRJointPlate**);
```

**C#**

```
public IRJointPlate FlanPlateLower { get; }
```

**Visual Basic**

```
Public ReadOnly FlanPlateLower As IRJointPlate
```

**Description**

Label of the lower bracket of the column flange.

**Version**

Available since version 9.5.

**I.7.2.9 FlanPlatePosLower****C++**

```
HRESULT get_FlanPlatePosLower(double*);  
HRESULT put_FlanPlatePosLower(double);
```

**C#**

```
public double FlanPlatePosLower { get; set; }
```

**Visual Basic**

```
Public FlanPlatePosLower As Double
```

**Description**

Lower bracket of the column flange - ebd parameter.

**Version**

Available since version 9.5.

**I.7.2.10 FlanPlatePosUpper****C++**

```
HRESULT get_FlanPlatePosUpper(double*);  
HRESULT put_FlanPlatePosUpper(double);
```

**C#**

```
public double FlanPlatePosUpper { get; set; }
```

**Visual Basic**

```
Public FlanPlatePosUpper As Double
```

**Description**

Upper bracket of the column flange - ebu parameter.

**Version**

Available since version 9.5.

**I.7.2.11 FlanPlateUpper****C++**

```
HRESULT get_FlanPlateUpper(IRJointPlate**);
```

**C#**

```
public IRJointPlate FlanPlateUpper { get; }
```

**Visual Basic**

```
Public ReadOnly FlanPlateUpper As IRJointPlate
```

**Description**

Label of the upper bracket of the column flange.

**Version**

Available since version 9.5.

**I.7.2.12 IsBolted****C++**

```
HRESULT get_IsBolted(int*);  
HRESULT put_IsBolted(int);
```

**C#**

```
public int IsBolted { get; set; }
```

**Visual Basic**

```
Public IsBolted As Integer
```

**Description**

The connection is bolted if IsBolted equals 1, otherwise, the connection is welded.

**Version**

Available since version 9.5.

**I.7.2.13 IsDefComponentMaterialSet****C++**

```
HRESULT get_IsDefComponentMaterialSet(int*);  
HRESULT put_IsDefComponentMaterialSet(int);
```

**C#**

```
public int IsDefComponentMaterialSet { get; set; }
```

**Visual Basic**

```
Public IsDefComponentMaterialSet As Integer
```

**Description**

All components are assigned the same material if IsDefComponentMaterialSet equals 1.

**Version**

Available since version 9.5.

**I.7.2.14 KneeType****C++**

```
HRESULT get_KneeType(IRJointKneeType*);  
HRESULT put_KneeType(IRJointKneeType);
```

**C#**

```
public IRJointKneeType KneeType { get; set; }
```

**Visual Basic**

```
Public KneeType As IRJointKneeType
```

**Description**

A type of a knee connection.

### I.7.2.15 Material

**C++**

```
HRESULT get_Material(IRJointKneeMaterials* );
HRESULT put_Material(IRJointKneeMaterials);
```

**C#**

```
public IRJointKneeMaterials Material { get; set; }
```

**Visual Basic**

```
Public Material As IRJointKneeMaterials
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### I.7.2.16 MaterPlates

**C++**

```
HRESULT get_MaterPlates(IRobotMaterialData** );
```

**C#**

```
public IRobotMaterialData MaterPlates { get; }
```

**Visual Basic**

```
Public ReadOnly MaterPlates As IRobotMaterialData
```

### I.7.2.17 Plate

**C++**

```
HRESULT get_Plate(IRJointPlate** );
```

**C#**

```
public IRJointPlate Plate { get; }
```

**Visual Basic**

```
Public ReadOnly Plate As IRJointPlate
```

**Description**

Main plate.

### I.7.2.18 PlatePosition

**C++**

```
HRESULT get_PlatePosition(double* );
HRESULT put_PlatePosition(double);
```

**C#**

```
public double PlatePosition { get; set; }
```

**Visual Basic**

```
Public PlatePosition As Double
```

**Description**

Distance between the top edge of the slab and the top edge of the adjoining element.

## Version

Available since version 9.5.

### I.7.2.19 PlatePositionLow

#### C++

```
HRESULT get_PlatePositionLow(double*);  
HRESULT put_PlatePositionLow(double);
```

#### C#

```
public double PlatePositionLow { get; set; }
```

#### Visual Basic

```
Public PlatePositionLow As Double
```

#### Description

Distance between bottom plate edge and the bottom edge of the adjoining element .

### I.7.2.20 StiffDiag

#### C++

```
HRESULT get_StiffDiag(IRJointKneeDiagonalStiff**);
```

#### C#

```
public IRJointKneeDiagonalStiff StiffDiag { get; }
```

#### Visual Basic

```
Public ReadOnly StiffDiag As IRJointKneeDiagonalStiff
```

#### Description

Inclined column bracket.

### I.7.2.21 StiffLow

#### C++

```
HRESULT get_StiffLow(IRJointKneeStiffColumn**);
```

#### C#

```
public IRJointKneeStiffColumn StiffLow { get; }
```

#### Visual Basic

```
Public ReadOnly StiffLow As IRJointKneeStiffColumn
```

#### Description

Bottom column bracing.

### I.7.2.22 StiffTypeLow

#### C++

```
HRESULT get_StiffTypeLow(IRJointKneeReinfType*);  
HRESULT put_StiffTypeLow(IRJointKneeReinfType);
```

#### C#

```
public IRJointKneeReinfType StiffTypeLow { get; set; }
```

#### Visual Basic

```
Public StiffTypeLow As IRJointKneeReinfType
```

**Description**

A type of lower bracket.

**I.7.2.23 StiffTypeUp****C++**

```
HRESULT get_StiffTypeUp(IRJointKneeReinfType* );
HRESULT put_StiffTypeUp(IRJointKneeReinfType);
```

**C#**

```
public IRJointKneeReinfType StiffTypeUp { get; set; }
```

**Visual Basic**

```
Public StiffTypeUp As IRJointKneeReinfType
```

**Description**

A type of upper bracket.

**I.7.2.24 StiffUp****C++**

```
HRESULT get_StiffUp(IRJointKneeStiffColumn** );
```

**C#**

```
public IRJointKneeStiffColumn StiffUp { get; }
```

**Visual Basic**

```
Public ReadOnly StiffUp As IRJointKneeStiffColumn
```

**Description**

Top column bracing.

**I.7.2.25 TensionPlateLow****C++**

```
HRESULT get_TensionPlateLow(IRJointPlate** );
```

**C#**

```
public IRJointPlate TensionPlateLow { get; }
```

**Visual Basic**

```
Public ReadOnly TensionPlateLow As IRJointPlate
```

**Description**

Bottom plate reinforcing the beam.

**I.7.2.26 TensionPlateUp****C++**

```
HRESULT get_TensionPlateUp(IRJointPlate** );
```

**C#**

```
public IRJointPlate TensionPlateUp { get; }
```

**Visual Basic**

```
Public ReadOnly TensionPlateUp As IRJointPlate
```

**Description**

Top plate reinforcing the beam.

**I.7.2.27 VStiffLow****C++**

```
HRESULT get_VStiffLow(IRJointPlate**);
```

**C#**

```
public IRJointPlate VStiffLow { get; }
```

**Visual Basic**

```
Public ReadOnly VStiffLow As IRJointPlate
```

**Description**

A bottom, vertical bracing of a beam.

**I.7.2.28 VStiffUp****C++**

```
HRESULT get_VStiffUp(IRJointPlate**);
```

**C#**

```
public IRJointPlate VStiffUp { get; }
```

**Visual Basic**

```
Public ReadOnly VStiffUp As IRJointPlate
```

**Description**

A top, vertical bracing of a beam.

**I.7.2.29 WebBeamStiffenerVLower****C++**

```
HRESULT get_WebBeamStiffenerVLower(IRJointPlate**);
```

**C#**

```
public IRJointPlate WebBeamStiffenerVLower { get; }
```

**Visual Basic**

```
Public ReadOnly WebBeamStiffenerVLower As IRJointPlate
```

**Description**

Label of the lower vertical stiffener of a beam.

**Version**

Available since version 9.5.

**I.7.2.30 WebBeamStiffenerVUpper****C++**

```
HRESULT get_WebBeamStiffenerVUpper(IRJointPlate**);
```

**C#**

```
public IRJointPlate WebBeamStiffenerVUpper { get; }
```

**Visual Basic**

```
Public ReadOnly WebBeamStiffenerVUpper As IRJointPlate
```

**Description**

Label of the upper vertical stiffener of a beam.

**Version**

Available since version 9.5.

**I.7.2.31 WebColumnStiffenerHLower****C++**

```
HRESULT get_WebColumnStiffenerHLower(IRJointPlate**);
```

**C#**

```
public IRJointPlate WebColumnStiffenerHLower { get; }
```

**Visual Basic**

```
Public ReadOnly WebColumnStiffenerHLower As IRJointPlate
```

**Description**

Label of the lower horizontal stiffener of a column.

**Version**

Available since version 9.5.

**I.7.2.32 WebColumnStiffenerHUpper****C++**

```
HRESULT get_WebColumnStiffenerHUpper(IRJointPlate**);
```

**C#**

```
public IRJointPlate WebColumnStiffenerHUpper { get; }
```

**Visual Basic**

```
Public ReadOnly WebColumnStiffenerHUpper As IRJointPlate
```

**Description**

Label of the upper horizontal stiffener of a column.

**Version**

Available since version 9.5.

**I.7.2.33 WebPlate****C++**

```
HRESULT get_WebPlate(IRJointKneeWebPlate**);
```

**C#**

```
public IRJointKneeWebPlate WebPlate { get; }
```

**Visual Basic**

```
Public ReadOnly WebPlate As IRJointKneeWebPlate
```

**Description**

A plate reinforcing a column web.

### I.7.2.34 WebStiffType

**C++**

```
HRESULT get_WebStiffType(IRJointKneeWebStiffType* );
HRESULT put_WebStiffType(IRJointKneeWebStiffType);
```

**C#**

```
public IRJointKneeWebStiffType WebStiffType { get; set; }
```

**Visual Basic**

```
Public WebStiffType As IRJointKneeWebStiffType
```

**Description**

A bracket type of a column web.

### I.7.2.35 WeldBracketDownFlange

**C++**

```
HRESULT get_WeldBracketDownFlange(IRJointWeld** );
```

**C#**

```
public IRJointWeld WeldBracketDownFlange { get; }
```

**Visual Basic**

```
Public ReadOnly WeldBracketDownFlange As IRJointWeld
```

**Description**

Thickness of the weld connecting the flange of lower bracket with the column.

### I.7.2.36 WeldBracketUpFlange

**C++**

```
HRESULT get_WeldBracketUpFlange(IRJointWeld** );
```

**C#**

```
public IRJointWeld WeldBracketUpFlange { get; }
```

**Visual Basic**

```
Public ReadOnly WeldBracketUpFlange As IRJointWeld
```

**Description**

Thickness of the weld connecting the flange of upper bracket with the column .

**Version**

Available since version 9.5.

### I.7.2.37 WeldFlange

**C++**

```
HRESULT get_WeldFlange(IRJointWeld** );
```

**C#**

```
public IRJointWeld WeldFlange { get; }
```

**Visual Basic**

```
Public ReadOnly WeldFlange As IRJointWeld
```

### I.7.2.38 WeldStiff

C++

```
HRESULT get_WeldStiff(IRJointWeld**);
```

C#

```
public IRJointWeld WeldStiff { get; }
```

Visual Basic

```
Public ReadOnly WeldStiff As IRJointWeld
```

### I.7.2.39 WeldWeb

C++

```
HRESULT get_WeldWeb(IRJointWeld**);
```

C#

```
public IRJointWeld WeldWeb { get; }
```

Visual Basic

```
Public ReadOnly WeldWeb As IRJointWeld
```

## I.8 IRJointKneeWebStiffType

C++

```
enum IRJointKneeWebStiffType;
```

C#

```
public enum IRJointKneeWebStiffType;
```

Visual Basic

```
Public Enum IRJointKneeWebStiffType
```

Description

Bracket types of a column web for a knee connection. .

## I.9 IRJointKneeDiagonalStiffType

C++

```
enum IRJointKneeDiagonalStiffType;
```

C#

```
public enum IRJointKneeDiagonalStiffType;
```

Visual Basic

```
Public Enum IRJointKneeDiagonalStiffType
```

Description

Available slanting bracket types for a column web.

## I.10 IRJointKneeDiagonalStiff

Class Hierarchy

**C++**

```
interface IRJointKneeDiagonalStiff : IDispatch;
```

**C#**

```
public interface IRJointKneeDiagonalStiff;
```

**Visual Basic**

```
Public Interface IRJointKneeDiagonalStiff
```

**Description**

An inclined stiffener of a column.

**I.10.1 IRJointKneeDiagonalStiff Members**

The following tables list the members exposed by IRJointKneeDiagonalStiff.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Exist (❑ see page 1956)	
❖	Material (❑ see page 1957)	Name of the component material.
❖	Thick (❑ see page 1957)	Thickness.
❖	Type (❑ see page 1957)	A stiffener type.

**I.10.2 IRJointKneeDiagonalStiff Fields**

The fields of the IRJointKneeDiagonalStiff class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Exist (❑ see page 1956)	
❖	Material (❑ see page 1957)	Name of the component material.
❖	Thick (❑ see page 1957)	Thickness.
❖	Type (❑ see page 1957)	A stiffener type.

**I.10.2.1 Exist****C++**

```
HRESULT get_Exist(VARIANT_BOOL* );
```

**C#**

```
public bool Exist { get; }
```

**Visual Basic**

```
Public ReadOnly Exist As Boolean
```

**I.10.2.2 Material****C++**

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

**C#**

```
public BSTR Material { get; set; }
```

**Visual Basic**

```
Public Material As BSTR
```

**Description**

Name of the component material.

**Version**

Available since version 9.5.

### I.10.2.3 Thick

**C++**

```
HRESULT get_Thickness(double*);  
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thickness As Double
```

**Description**

Thickness.

### I.10.2.4 Type

**C++**

```
HRESULT get_Type(IRJointKneeDiagonalStiffType*);  
HRESULT put_Type(IRJointKneeDiagonalStiffType);
```

**C#**

```
public IRJointKneeDiagonalStiffType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointKneeDiagonalStiffType
```

**Description**

A stiffener type.

## I.11 IRJointKneeStiffColumn

**Class Hierarchy****C++**

```
interface IRJointKneeStiffColumn : IDispatch;
```

**C#**

```
public interface IRJointKneeStiffColumn;
```

**Visual Basic**

```
Public Interface IRJointKneeStiffColumn
```

**Description**

Column bracket in a knee connection. .

## I.11.1 IRJointKneeStiffColumn Members

The following tables list the members exposed by IRJointKneeStiffColumn.

### Public Fields

	Name	Description
◆	Thick (see page 1958)	Thickness.

## I.11.2 IRJointKneeStiffColumn Fields

The fields of the IRJointKneeStiffColumn class are listed here.

### Public Fields

	Name	Description
◆	Thick (see page 1958)	Thickness.

### I.11.2.1 Thick

#### C++

```
HRESULT get_Thick(double* );
HRESULT put_Thick(double);
```

#### C#

```
public double Thick { get; set; }
```

#### Visual Basic

```
Public Thick As Double
```

#### Description

Thickness.

## I.12 IRJointKneeLoad

### Class Hierarchy

#### C++

```
interface IRJointKneeLoad : IRJointLoad;
```

#### C#

```
public interface IRJointKneeLoad : IRJointLoad;
```

#### Visual Basic

```
Public Interface IRJointKneeLoad
```

#### Description

Interface describing a load applied to a knee connection. .

## I.12.1 IRJointKneeLoad Members

The following tables list the members exposed by IRJointKneeLoad.

### Public Fields

	Name	Description
◆	Cases (see page 2166)	Selection of load cases, defined for the structure..
◆	M (see page 1959)	

❖	N (see page 1959)	
❖	Q (see page 1959)	
❖	Type (see page 2166)	Load type (defined manually or taken from Robot).

## I.12.2 IRJointKneeLoad Fields

The fields of the IRJointKneeLoad class are listed here.

### Public Fields

	Name	Description
❖	M (see page 1959)	
❖	N (see page 1959)	
❖	Q (see page 1959)	

### I.12.2.1 M

#### C++

```
HRESULT get_M(double* );
HRESULT put_M(double);
```

#### C#

```
public double M { get; set; }
```

#### Visual Basic

```
Public M As double
```

### I.12.2.2 N

#### C++

```
HRESULT get_N(double* );
HRESULT put_N(double);
```

#### C#

```
public double N { get; set; }
```

#### Visual Basic

```
Public N As double
```

### I.12.2.3 Q

#### C++

```
HRESULT get_Q(double* );
HRESULT put_Q(double);
```

#### C#

```
public double Q { get; set; }
```

#### Visual Basic

```
Public Q As double
```

## I.13 IRJointKneeMaterials

#### C++

```
enum IRJointKneeMaterials;
```

#### C#

```
public enum IRJointKneeMaterials;
```

## Visual Basic

```
Public Enum IRJointKneeMaterials
```

### Members

Members	Description
I_JKM_PLATE = 0	Available since version 9.7.
I_JKM_PLATE_LEFT = 1	Available since version 9.7.
I_JKM_FLAN_PLATE_UPPER = 2	Available since version 9.7.
I_JKM_FLAN_PLATE_LOWER = 3	Available since version 9.7.
I_JKM_WEB_COLUMN_STIFF_H_UPPER = 4	Available since version 9.7.
I_JKM_WEB_COLUMN_STIFF_H_LOWER = 5	Available since version 9.7.
I_JKM_WEB_BEAM_STIFF_V_UPPER = 6	Available since version 9.7.
I_JKM_WEB_BEAM_STIFF_V_LOWER = 7	Available since version 9.7.
I_JKM_TENSION_PLATE_UPPER = 8	Available since version 9.7.
I_JKM_TENSION_PLATE_LOWER = 9	Available since version 9.7.

### Version

Available since version 9.7.

## II Column base connections

### Enumerations

	Name	Description
☞	IRJointAnchorType (☞ see page 1993)	Available anchorage types.
☞	IRJointFootPlateType (☞ see page 1999)	Available connection types - plate/angle.
☞	IRJointWedgeType (☞ see page 2003)	Available wedge types.
☞	IRJointAnchorPlateType (☞ see page 2009)	Types of plates near anchor.
☞	IRJointFootStiffType (☞ see page 2019)	

### Interfaces

	Name	Description
☞	IRJointAnchorBolt (☞ see page 1991)	Anchorage bolt.
☞	IRJointAnchorPlate (☞ see page 1992)	Plate of an anchor bolt.
☞	IRJointAnchor (☞ see page 1994)	Anchorage definition.
☞	IRJointFootBolts (☞ see page 1996)	Bolt group of identical parameters for pinned column base. .
☞	IRJointFootPlate (☞ see page 1999)	
☞	IRJointWedge (☞ see page 2001)	
☞	IRJointBearingPlate (☞ see page 2003)	
☞	IRJointFootWelds (☞ see page 2004)	Description of connection element welds.
☞	IRJointColumnBracket (☞ see page 2006)	Column stiffener.

	IRJointFootStiffenerVert (see page 2009)	
	IRJointFootStiffenerHoriz (see page 2011)	
	IRJointFootMaterials (see page 2012)	
	IRJointColumnSquare (see page 2014)	
	IRJointFootStiffenerSimple (see page 2015)	
	IRJointFootStiffenerComplex (see page 2017)	

## II.1 Pinned column base

### Enumerations

	Name	Description
	IRJointPinnedColumnBaseStiffType (see page 1961)	

### Interfaces

	Name	Description
	IRJointPinnedColumnBase (see page 1962)	Pinned column base.
	IRJointPinnedLoad (see page 1967)	

### II.1.1 IRJointPinnedColumnBaseStiffType

#### C++

```
enum IRJointPinnedColumnBaseStiffType;
```

#### C#

```
public enum IRJointPinnedColumnBaseStiffType;
```

#### Visual Basic

```
Public Enum IRJointPinnedColumnBaseStiffType
```

#### Members

Members	Description
I_JPCBST_LRDIAG = 0	Available since version 1.7.
I_JPCBST_LDIAG = 1	Available since version 1.7.
I_JPCBST_RDIAG = 2	Available since version 1.7.
I_JPCBST_LRI = 3	Available since version 1.7.
I_JPCBST_LI = 4	Available since version 1.7.
I_JPCBST_RI = 5	Available since version 1.7.
I_JPCBST_NONE = 6	Available since version 1.7.

### II.1.2 IRJointPinnedColumnBase

#### Class Hierarchy

#### C++

```
interface IRJointPinnedColumnBase : IRJointConnection;
```

**C#**

```
public interface IRJointPinnedColumnBase : IRJointConnection;
```

**Visual Basic**

```
Public Interface IRJointPinnedColumnBase
```

**Description**

Pinned column base.

**II.1.2.1 IRJointPinnedColumnBase Members**

The following tables list the members exposed by IRJointPinnedColumnBase.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Anchor (see page 1963)	Anchorage Available since version 1.7.
❖	Base (see page 1963)	Base Available since version 1.7.
❖	BasePlateMaterial (see page 1964)	
❖	Bearing (see page 1964)	Bearing Available since version 1.7.
❖	BearingPlate (see page 1964)	Bearing (see page 1964) plate Available since version 1.7.
❖	Bolts (see page 1964)	Group of connecting bolts Available since version 1.7.
❖	FootPlate (see page 1965)	Plate Available since version 1.7.
❖	Materials (see page 1965)	Available since version 1.7.
❖	NodeNumber (see page 1965)	
❖	Profile (see page 1965)	Column section Available since version 1.7.
❖	Square (see page 1966)	Available since version 1.7.
❖	StiffHoriz (see page 1966)	Available since version 1.7.
❖	StiffType (see page 1966)	Available since version 1.7.
❖	StiffVert (see page 1966)	Available since version 1.7.
❖	Type (see page 2148)	A connection type.
❖	Washer (see page 1967)	Washer Available since version 1.7.
❖	Wedge (see page 1967)	Wedge Available since version 1.7.
❖	Welds (see page 1967)	Welds in a connection Available since version 1.7.
❖	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
❖	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

**II.1.2.2 IRJointPinnedColumnBase Fields**

The fields of the IRJointPinnedColumnBase class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Anchor (see page 1963)	Anchorage Available since version 1.7.
❖	Base (see page 1963)	Base Available since version 1.7.
❖	BasePlateMaterial (see page 1964)	

◆	Bearing (see page 1964)	Bearing Available since version 1.7.
◆	BearingPlate (see page 1964)	Bearing (see page 1964) plate Available since version 1.7.
◆	Bolts (see page 1964)	Group of connecting bolts Available since version 1.7.
◆	FootPlate (see page 1965)	Plate Available since version 1.7.
◆	Materials (see page 1965)	Available since version 1.7.
◆	NodeNumber (see page 1965)	
◆	Profile (see page 1965)	Column section Available since version 1.7.
◆	Square (see page 1966)	Available since version 1.7.
◆	StiffHoriz (see page 1966)	Available since version 1.7.
◆	StiffType (see page 1966)	Available since version 1.7.
◆	StiffVert (see page 1966)	Available since version 1.7.
◆	Washer (see page 1967)	Washer Available since version 1.7.
◆	Wedge (see page 1967)	Wedge Available since version 1.7.
◆	Welds (see page 1967)	Welds in a connection Available since version 1.7.

### II.1.2.2.1 Anchor

#### C++

```
HRESULT get_Anchor( IRJointAnchor** );
```

#### C#

```
public IRJointAnchor Anchor { get; }
```

#### Visual Basic

```
Public ReadOnly Anchor As IRJointAnchor
```

#### Description

Anchorage Available since version 1.7.

### II.1.2.2.2 Base

#### C++

```
HRESULT get_Base( IRJointPlate** );
```

#### C#

```
public IRJointPlate Base { get; }
```

#### Visual Basic

```
Public ReadOnly Base As IRJointPlate
```

#### Description

Base Available since version 1.7.

### II.1.2.2.3 BasePlateMaterial

#### C++

```
HRESULT get_BasePlateMaterial( IRobotMaterialData** );
```

#### C#

```
public IRobotMaterialData BasePlateMaterial { get; }
```

#### Visual Basic

```
Public ReadOnly BasePlateMaterial As IRobotMaterialData
```

**Version**

Available since version 4.5.

**II.1.2.2.4 Bearing****C++**

```
HRESULT get_Bearing(IRJointPlate**);
```

**C#**

```
public IRJointPlate Bearing { get; }
```

**Visual Basic**

```
Public ReadOnly Bearing As IRJointPlate
```

**Description**

Bearing Available since version 1.7.

**II.1.2.2.5 BearingPlate****C++**

```
HRESULT get_BearingPlate(IRJointBearingPlate**);
```

**C#**

```
public IRJointBearingPlate BearingPlate { get; }
```

**Visual Basic**

```
Public ReadOnly BearingPlate As IRJointBearingPlate
```

**Description**

Bearing (see page 1964) plate Available since version 1.7.

**II.1.2.2.6 Bolts****C++**

```
HRESULT get_Bolts(IRJointFootBolts**);
```

**C#**

```
public IRJointFootBolts Bolts { get; }
```

**Visual Basic**

```
Public ReadOnly Bolts As IRJointFootBolts
```

**Description**

Group of connecting bolts Available since version 1.7.

**II.1.2.2.7 FootPlate****C++**

```
HRESULT get_FootPlate(IRJointFootPlate**);
```

**C#**

```
public IRJointFootPlate FootPlate { get; }
```

**Visual Basic**

```
Public ReadOnly FootPlate As IRJointFootPlate
```

**Description**

Plate Available since version 1.7.

**II.1.2.2.8 Materials****C++**

```
HRESULT get_Materials(IRJointFootMaterials**);
```

**C#**

```
public IRJointFootMaterials Materials { get; }
```

**Visual Basic**

```
Public ReadOnly Materials As IRJointFootMaterials
```

**Description**

Available since version 1.7.

**II.1.2.2.9 NodeNumber****C++**

```
HRESULT get_NodeNumber(long*);  
HRESULT put_NodeNumber(long);
```

**C#**

```
public long NodeNumber { get; set; }
```

**Visual Basic**

```
Public NodeNumber As long
```

**Version**

Available since version 4.5.

**II.1.2.2.10 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Column section Available since version 1.7.

**II.1.2.2.11 Square****C++**

```
HRESULT get_Square(IRJointColumnSquare**);
```

**C#**

```
public IRJointColumnSquare Square { get; }
```

**Visual Basic**

```
Public ReadOnly Square As IRJointColumnSquare
```

**Description**

Available since version 1.7.

**II.1.2.2.12 StiffHoriz****C++**

```
HRESULT get_StiffHoriz(IRJointFootStiffenerHoriz**);
```

**C#**

```
public IRJointFootStiffenerHoriz StiffHoriz { get; }
```

**Visual Basic**

```
Public ReadOnly StiffHoriz As IRJointFootStiffenerHoriz
```

**Description**

Available since version 1.7.

**II.1.2.2.13 StiffType****C++**

```
HRESULT get_StiffType(IRJointPinnedColumnBaseStiffType*);  
HRESULT put_StiffType(IRJointPinnedColumnBaseStiffType);
```

**C#**

```
public IRJointPinnedColumnBaseStiffType StiffType { get; set; }
```

**Visual Basic**

```
Public StiffType As IRJointPinnedColumnBaseStiffType
```

**Description**

Available since version 1.7.

**II.1.2.2.14 StiffVert****C++**

```
HRESULT get_StiffVert(IRJointFootStiffenerVert**);
```

**C#**

```
public IRJointFootStiffenerVert StiffVert { get; }
```

**Visual Basic**

```
Public ReadOnly StiffVert As IRJointFootStiffenerVert
```

**Description**

Available since version 1.7.

**II.1.2.2.15 Washer****C++**

```
HRESULT get_Washer(IRJointPlate**);
```

**C#**

```
public IRJointPlate Washer { get; }
```

**Visual Basic**

```
Public ReadOnly Washer As IRJointPlate
```

**Description**

Washer Available since version 1.7.

**II.1.2.2.16 Wedge****C++**

```
HRESULT get_Wedge( IRJointWedge** );
```

**C#**

```
public IRJointWedge Wedge { get; }
```

**Visual Basic**

```
Public ReadOnly Wedge As IRJointWedge
```

**Description**

Wedge Available since version 1.7.

**II.1.2.2.17 Welds****C++**

```
HRESULT get_Welds( IRJointFootWelds** );
```

**C#**

```
public IRJointFootWelds Welds { get; }
```

**Visual Basic**

```
Public ReadOnly Welds As IRJointFootWelds
```

**Description**

Welds in a connection Available since version 1.7.

**II.1.3 IRJointPinnedLoad****Class Hierarchy****C++**

```
interface IRJointPinnedLoad : IRJointLoad;
```

**C#**

```
public interface IRJointPinnedLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointPinnedLoad
```

**II.1.3.1 IRJointPinnedLoad Members**

The following tables list the members exposed by IRJointPinnedLoad.

**Public Fields**

	Name	Description
❖	Cases ( <a href="#">see page 2166</a> )	Selection of load cases, defined for the structure..
❖	Nc ( <a href="#">see page 1968</a> )	Available since version 1.7.
❖	Nt ( <a href="#">see page 1968</a> )	Available since version 1.7.
❖	NTy ( <a href="#">see page 1969</a> )	Available since version 1.7.
❖	NTz ( <a href="#">see page 1969</a> )	Available since version 1.7.

◆	Ty (see page 1969)	Available since version 1.7.
◆	Type (see page 2166)	Load type (defined manually or taken from Robot).
◆	Tz (see page 1969)	Available since version 1.7.

### II.1.3.2 IRJointPinnedLoad Fields

The fields of the IRJointPinnedLoad class are listed here.

#### Public Fields

	Name	Description
◆	Nc (see page 1968)	Available since version 1.7.
◆	Nt (see page 1968)	Available since version 1.7.
◆	NTy (see page 1969)	Available since version 1.7.
◆	NTz (see page 1969)	Available since version 1.7.
◆	Ty (see page 1969)	Available since version 1.7.
◆	Tz (see page 1969)	Available since version 1.7.

#### II.1.3.2.1 Nc

##### C++

```
HRESULT get_Nc(double* );
HRESULT put_Nc(double );
```

##### C#

```
public double Nc { get; set; }
```

##### Visual Basic

```
Public Nc As double
```

#### Description

Available since version 1.7.

#### II.1.3.2.2 Nt

##### C++

```
HRESULT get_Nt(double* );
HRESULT put_Nt(double );
```

##### C#

```
public double Nt { get; set; }
```

##### Visual Basic

```
Public Nt As double
```

#### Description

Available since version 1.7.

#### II.1.3.2.3 NTy

##### C++

```
HRESULT get_NTy(double* );
HRESULT put_NTy(double );
```

##### C#

```
public double NTy { get; set; }
```

**Visual Basic**

```
Public NTy As double
```

**Description**

Available since version 1.7.

**II.1.3.2.4 NTz****C++**

```
HRESULT get_NTz(double*);  
HRESULT put_NTz(double);
```

**C#**

```
public double NTz { get; set; }
```

**Visual Basic**

```
Public NTz As double
```

**Description**

Available since version 1.7.

**II.1.3.2.5 Ty****C++**

```
HRESULT get_Ty(double*);  
HRESULT put_Ty(double);
```

**C#**

```
public double Ty { get; set; }
```

**Visual Basic**

```
Public Ty As double
```

**Description**

Available since version 1.7.

**II.1.3.2.6 Tz****C++**

```
HRESULT get_Tz(double*);  
HRESULT put_Tz(double);
```

**C#**

```
public double Tz { get; set; }
```

**Visual Basic**

```
Public Tz As double
```

**Description**

Available since version 1.7.

## II.2 Fixed column base

### Enumerations

	Name	Description
	IRJointFixedColumnBaseStiffType ( <a href="#">see page 1979</a> )	

### Interfaces

	Name	Description
	IRJointFixedLoad ( <a href="#">see page 1970</a> )	
	IRJointFixedColumnBase ( <a href="#">see page 1972</a> )	
	IRJointFixedFootWelds ( <a href="#">see page 1977</a> )	

### II.2.1 IRJointFixedLoad

#### Class Hierarchy

#### C++

```
interface IRJointFixedLoad : IRJointLoad;
```

#### C#

```
public interface IRJointFixedLoad : IRJointLoad;
```

#### Visual Basic

```
Public Interface IRJointFixedLoad
```

#### II.2.1.1 IRJointFixedLoad Members

The following tables list the members exposed by IRJointFixedLoad.

#### Public Fields

	Name	Description
	Cases ( <a href="#">see page 2166</a> )	Selection of load cases, defined for the structure. .
	Fx ( <a href="#">see page 1971</a> )	Available since version 1.7.
	Fy ( <a href="#">see page 1971</a> )	Available since version 1.7.
	Fz ( <a href="#">see page 1971</a> )	Available since version 1.7.
	My ( <a href="#">see page 1972</a> )	Available since version 1.7.
	Mz ( <a href="#">see page 1972</a> )	
	Type ( <a href="#">see page 2166</a> )	Load type (defined manually or taken from Robot).

#### II.2.1.2 IRJointFixedLoad Fields

The fields of the IRJointFixedLoad class are listed here.

#### Public Fields

	Name	Description
	Fx ( <a href="#">see page 1971</a> )	Available since version 1.7.
	Fy ( <a href="#">see page 1971</a> )	Available since version 1.7.
	Fz ( <a href="#">see page 1971</a> )	Available since version 1.7.
	My ( <a href="#">see page 1972</a> )	Available since version 1.7.
	Mz ( <a href="#">see page 1972</a> )	

### II.2.1.2.1 Fx

**C++**

```
HRESULT get_Fx(double*);  
HRESULT put_Fx(double);
```

**C#**

```
public double Fx { get; set; }
```

**Visual Basic**

```
Public Fx As double
```

**Description**

Available since version 1.7.

### II.2.1.2.2 Fy

**C++**

```
HRESULT get_Fy(double*);  
HRESULT put_Fy(double);
```

**C#**

```
public double Fy { get; set; }
```

**Visual Basic**

```
Public Fy As double
```

**Description**

Available since version 1.7.

### II.2.1.2.3 Fz

**C++**

```
HRESULT get_Fz(double*);  
HRESULT put_Fz(double);
```

**C#**

```
public double Fz { get; set; }
```

**Visual Basic**

```
Public Fz As double
```

**Description**

Available since version 1.7.

### II.2.1.2.4 My

**C++**

```
HRESULT get_My(double*);  
HRESULT put_My(double);
```

**C#**

```
public double My { get; set; }
```

**Visual Basic**

```
Public My As double
```

## Description

Available since version 1.7.

### II.2.1.2.5 Mz

#### C++

```
HRESULT get_Mz( double* );
HRESULT put_Mz( double );
```

#### C#

```
public double Mz { get; set; }
```

#### Visual Basic

```
Public Mz As Double
```

#### Version

Available since version 7.5.

## II.2.2 IRJointFixedColumnBase

#### Class Hierarchy

#### C++

```
interface IRJointFixedColumnBase : IRJointConnection;
```

#### C#

```
public interface IRJointFixedColumnBase : IRJointConnection;
```

#### Visual Basic

```
Public Interface IRJointFixedColumnBase
```

### II.2.2.1 IRJointFixedColumnBase Members

The following tables list the members exposed by IRJointFixedColumnBase.

#### Public Fields

	Name	Description
◆	Anchor ( [ see page 1973) )	Available since version 1.7.
◆	Base ( [ see page 1974) )	Available since version 1.7.
◆	BasePlateMaterial ( [ see page 1974) )	
◆	Bolts ( [ see page 1974) )	Available since version 1.7.
◆	ComplexStiff ( [ see page 1974) )	Available since version 1.7.
◆	FootPlate ( [ see page 1975) )	Available since version 1.7.
◆	Materials ( [ see page 1975) )	Available since version 1.7.
◆	NodeNumber ( [ see page 1975) )	
◆	Profile ( [ see page 1975) )	Available since version 1.7.
◆	SimpleStiff ( [ see page 1976) )	Available since version 1.7.
◆	StiffType ( [ see page 1976) )	Available since version 1.7.
◆	Type ( [ see page 2148) )	A connection type.
◆	Washer ( [ see page 1976) )	Available since version 1.7.
◆	Wedge ( [ see page 1976) )	Available since version 1.7.
◆	Welds ( [ see page 1977) )	Available since version 1.7.
◆	WFRelPos ( [ see page 2148) )	A type of relative position of a beam and a column.

## Public Methods

	Name	Description
💡	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
💡	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

## II.2.2.2 IRJointFixedColumnBase Fields

The fields of the IRJointFixedColumnBase class are listed here.

### Public Fields

	Name	Description
💡	Anchor (see page 1973)	Available since version 1.7.
💡	Base (see page 1974)	Available since version 1.7.
💡	BasePlateMaterial (see page 1974)	
💡	Bolts (see page 1974)	Available since version 1.7.
💡	ComplexStiff (see page 1974)	Available since version 1.7.
💡	FootPlate (see page 1975)	Available since version 1.7.
💡	Materials (see page 1975)	Available since version 1.7.
💡	NodeNumber (see page 1975)	
💡	Profile (see page 1975)	Available since version 1.7.
💡	SimpleStiff (see page 1976)	Available since version 1.7.
💡	StiffType (see page 1976)	Available since version 1.7.
💡	Washer (see page 1976)	Available since version 1.7.
💡	Wedge (see page 1976)	Available since version 1.7.
💡	Welds (see page 1977)	Available since version 1.7.

### II.2.2.2.1 Anchor

#### C++

```
HRESULT get_Anchor(IRJointAnchor**);
```

#### C#

```
public IRJointAnchor Anchor { get; }
```

#### Visual Basic

```
Public ReadOnly Anchor As IRJointAnchor
```

#### Description

Available since version 1.7.

### II.2.2.2.2 Base

#### C++

```
HRESULT get_Base(IRJointPlate**);
```

#### C#

```
public IRJointPlate Base { get; }
```

#### Visual Basic

```
Public ReadOnly Base As IRJointPlate
```

**Description**

Available since version 1.7.

**II.2.2.2.3 BasePlateMaterial****C++**

```
HRESULT get_BasePlateMaterial(IRobotMaterialData**);
```

**C#**

```
public IRobotMaterialData BasePlateMaterial { get; }
```

**Visual Basic**

```
Public ReadOnly BasePlateMaterial As IRobotMaterialData
```

**Version**

Available since version 4.5.

**II.2.2.2.4 Bolts****C++**

```
HRESULT get_Bolts(IRJointFootBolts**);
```

**C#**

```
public IRJointFootBolts Bolts { get; }
```

**Visual Basic**

```
Public ReadOnly Bolts As IRJointFootBolts
```

**Description**

Available since version 1.7.

**II.2.2.2.5 ComplexStiff****C++**

```
HRESULT get_ComplexStiff(IRJointFootStiffenerComplex**);
```

**C#**

```
public IRJointFootStiffenerComplex ComplexStiff { get; }
```

**Visual Basic**

```
Public ReadOnly ComplexStiff As IRJointFootStiffenerComplex
```

**Description**

Available since version 1.7.

**II.2.2.2.6 FootPlate****C++**

```
HRESULT get_FootPlate(IRJointFootPlate**);
```

**C#**

```
public IRJointFootPlate FootPlate { get; }
```

**Visual Basic**

```
Public ReadOnly FootPlate As IRJointFootPlate
```

**Description**

Available since version 1.7.

**II.2.2.7 Materials****C++**

```
HRESULT get_Materials(IRJointFootMaterials**);
```

**C#**

```
public IRJointFootMaterials Materials { get; }
```

**Visual Basic**

```
Public ReadOnly Materials As IRJointFootMaterials
```

**Description**

Available since version 1.7.

**II.2.2.8 NodeNumber****C++**

```
HRESULT get_NodeNumber(long*);  
HRESULT put_NodeNumber(long);
```

**C#**

```
public long NodeNumber { get; set; }
```

**Visual Basic**

```
Public NodeNumber As long
```

**Version**

Available since version 4.5.

**II.2.2.9 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available since version 1.7.

**II.2.2.10 SimpleStiff****C++**

```
HRESULT get_SimpleStiff(IRJointFootStiffenerSimple**);
```

**C#**

```
public IRJointFootStiffenerSimple SimpleStiff { get; }
```

**Visual Basic**

```
Public ReadOnly SimpleStiff As IRJointFootStiffenerSimple
```

**Description**

Available since version 1.7.

**II.2.2.2.11 StiffType****C++**

```
HRESULT get_StiffType(IRJointFixedColumnBaseStiffType* );
HRESULT put_StiffType(IRJointFixedColumnBaseStiffType);
```

**C#**

```
public IRJointFixedColumnBaseStiffType StiffType { get; set; }
```

**Visual Basic**

```
Public StiffType As IRJointFixedColumnBaseStiffType
```

**Description**

Available since version 1.7.

**II.2.2.2.12 Washer****C++**

```
HRESULT get_Washer(IRJointPlate** );
```

**C#**

```
public IRJointPlate Washer { get; }
```

**Visual Basic**

```
Public ReadOnly Washer As IRJointPlate
```

**Description**

Available since version 1.7.

**II.2.2.2.13 Wedge****C++**

```
HRESULT get_Wedge(IRJointWedge** );
```

**C#**

```
public IRJointWedge Wedge { get; }
```

**Visual Basic**

```
Public ReadOnly Wedge As IRJointWedge
```

**Description**

Available since version 1.7.

**II.2.2.2.14 Welds****C++**

```
HRESULT get_Welds(IRJointFixedFootWelds** );
```

**C#**

```
public IRJointFixedFootWelds Welds { get; }
```

**Visual Basic**

```
Public ReadOnly Welds As IRJointFixedFootWelds
```

## Description

Available since version 1.7.

### II.2.3 IRJointFixedFootWelds

#### Class Hierarchy

#### C++

```
interface IRJointFixedFootWelds : IDispatch;
```

#### C#

```
public interface IRJointFixedFootWelds;
```

#### Visual Basic

```
Public Interface IRJointFixedFootWelds
```

#### II.2.3.1 IRJointFixedFootWelds Members

The following tables list the members exposed by IRJointFixedFootWelds.

#### Public Fields

	Name	Description
◆	FootPlate ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Stiff ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Washer ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Wedge ( <a href="#">see page 1978</a> )	Available since version 1.7.

#### II.2.3.2 IRJointFixedFootWelds Fields

The fields of the IRJointFixedFootWelds class are listed here.

#### Public Fields

	Name	Description
◆	FootPlate ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Stiff ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Washer ( <a href="#">see page 1978</a> )	Available since version 1.7.
◆	Wedge ( <a href="#">see page 1978</a> )	Available since version 1.7.

#### II.2.3.2.1 FootPlate

#### C++

```
HRESULT get_FootPlate(IRJointWeld**);
```

#### C#

```
public IRJointWeld FootPlate { get; }
```

#### Visual Basic

```
Public ReadOnly FootPlate As IRJointWeld
```

#### Description

Available since version 1.7.

#### II.2.3.2.2 Stiff

#### C++

```
HRESULT get_Stiff(IRJointWeld**);
```

**C#**

```
public IRJointWeld Stiff { get; }
```

**Visual Basic**

```
Public ReadOnly Stiff As IRJointWeld
```

**Description**

Available since version 1.7.

**II.2.3.2.3 Washer****C++**

```
HRESULT get_Washer(IRJointWeld**);
```

**C#**

```
public IRJointWeld Washer { get; }
```

**Visual Basic**

```
Public ReadOnly Washer As IRJointWeld
```

**Description**

Available since version 1.7.

**II.2.3.2.4 Wedge****C++**

```
HRESULT get_Wedge(IRJointWeld**);
```

**C#**

```
public IRJointWeld Wedge { get; }
```

**Visual Basic**

```
Public ReadOnly Wedge As IRJointWeld
```

**Description**

Available since version 1.7.

**II.2.4 IRJointFixedColumnBaseStiffType****C++**

```
enum IRJointFixedColumnBaseStiffType;
```

**C#**

```
public enum IRJointFixedColumnBaseStiffType;
```

**Visual Basic**

```
Public Enum IRJointFixedColumnBaseStiffType
```

**Members**

Members	Description
I_JFCBST_NONE = 0	Available since version 1.7.
I_JFCBST_SIMPLE = 1	Available since version 1.7.
I_JFCBST_COMPLEX = 2	Available since version 1.7.

## II.3 Concrete column base

### Enumerations

	Name	Description
	IRJointSpreadFootingType (see page 1983)	
	IRJointColumnBasePlateCalcModel (see page 1988)	

### Interfaces

	Name	Description
	IRJointConcreteMaterials (see page 1979)	
	IRJointConcreteColumn (see page 1981)	
	IRJointConcreteColumnFoundation (see page 1984)	
	IRJointConcreteColumnLoad (see page 1988)	

### II.3.1 IRJointConcreteMaterials

#### Class Hierarchy

#### C++

```
interface IRJointConcreteMaterials : IDispatch;
```

#### C#

```
public interface IRJointConcreteMaterials;
```

#### Visual Basic

```
Public Interface IRJointConcreteMaterials
```

#### II.3.1.1 IRJointConcreteMaterials Members

The following tables list the members exposed by IRJointConcreteMaterials.

#### Public Fields

	Name	Description
	CoeffBA (see page 1980)	Available since version 1.7.
	Dosage (see page 1980)	Available since version 1.7.
	Sigma (see page 1980)	Available since version 1.7.

#### II.3.1.2 IRJointConcreteMaterials Fields

The fields of the IRJointConcreteMaterials class are listed here.

#### Public Fields

	Name	Description
	CoeffBA (see page 1980)	Available since version 1.7.
	Dosage (see page 1980)	Available since version 1.7.
	Sigma (see page 1980)	Available since version 1.7.

### II.3.1.2.1 CoeffBA

#### C++

```
HRESULT get_CoeffBA(double*);  
HRESULT put_CoeffBA(double);
```

#### C#

```
public double CoeffBA { get; set; }
```

#### Visual Basic

```
Public CoeffBA As double
```

#### Description

Available since version 1.7.

### II.3.1.2.2 Dosage

#### C++

```
HRESULT get_Dosage(double*);  
HRESULT put_Dosage(double);
```

#### C#

```
public double Dosage { get; set; }
```

#### Visual Basic

```
Public Dosage As double
```

#### Description

Available since version 1.7.

### II.3.1.2.3 Sigma

#### C++

```
HRESULT get_Sigma(double*);  
HRESULT put_Sigma(double);
```

#### C#

```
public double Sigma { get; set; }
```

#### Visual Basic

```
Public Sigma As double
```

#### Description

Available since version 1.7.

## II.3.2 IRJointConcreteColumn

### Class Hierarchy

#### C++

```
interface IRJointConcreteColumn : IRJointConnection;
```

#### C#

```
public interface IRJointConcreteColumn : IRJointConnection;
```

#### Visual Basic

```
Public Interface IRJointConcreteColumn
```

### II.3.2.1 IRJointConcreteColumn Members

The following tables list the members exposed by IRJointConcreteColumn.

#### Public Fields

	Name	Description
◆	Base ( <a href="#">see page 1982</a> )	
◆	CalcModel ( <a href="#">see page 1982</a> )	
◆	Depth ( <a href="#">see page 1982</a> )	
◆	Foundation ( <a href="#">see page 1982</a> )	
◆	Materials ( <a href="#">see page 1983</a> )	
◆	Profile ( <a href="#">see page 1983</a> )	
◆	SpreadFootingType ( <a href="#">see page 1983</a> )	
◆	Type ( <a href="#">see page 2148</a> )	A connection type.
◆	WFRelPos ( <a href="#">see page 2148</a> )	A type of relative position of a beam and a column.

#### Public Methods

	Name	Description
◆	GetFromRobot ( <a href="#">see page 2149</a> )	The function gets connection parameters from Robot and fills an object with the data.
◆	SetToRobot ( <a href="#">see page 2149</a> )	The function sets and saves connection parameter in Robot.

### II.3.2.2 IRJointConcreteColumn Fields

The fields of the IRJointConcreteColumn class are listed here.

#### Public Fields

	Name	Description
◆	Base ( <a href="#">see page 1982</a> )	
◆	CalcModel ( <a href="#">see page 1982</a> )	
◆	Depth ( <a href="#">see page 1982</a> )	
◆	Foundation ( <a href="#">see page 1982</a> )	
◆	Materials ( <a href="#">see page 1983</a> )	
◆	Profile ( <a href="#">see page 1983</a> )	
◆	SpreadFootingType ( <a href="#">see page 1983</a> )	

#### II.3.2.2.1 Base

##### C++

```
HRESULT get_Base( IRJointPlate\*\* );
```

##### C#

```
public IRJointPlate Base { get; }
```

##### Visual Basic

```
Public ReadOnly Base As IRJointPlate
```

#### Version

Available since version 7.5.

### II.3.2.2.2 CalcModel

#### C++

```
HRESULT get_CalcModel(IRJointColumnBasePlateCalcModel**);  
HRESULT put_CalcModel(IRJointColumnBasePlateCalcModel);
```

#### C#

```
public IRJointColumnBasePlateCalcModel CalcModel { get; set; }
```

#### Visual Basic

```
Public CalcModel As IRJointColumnBasePlateCalcModel
```

#### Version

Available since version 7.5.

### II.3.2.2.3 Depth

#### C++

```
HRESULT get_Depth(double*);  
HRESULT put_Depth(double);
```

#### C#

```
public double Depth { get; set; }
```

#### Visual Basic

```
Public Depth As Double
```

### II.3.2.2.4 Foundation

#### C++

```
HRESULT get_Foundation(IRJointConcreteColumnFoundation**);
```

#### C#

```
public IRJointConcreteColumnFoundation Foundation { get; }
```

#### Visual Basic

```
Public ReadOnly Foundation As IRJointConcreteColumnFoundation
```

#### Version

Available since version 7.5.

### II.3.2.2.5 Materials

#### C++

```
HRESULT get_Materials(IRJointConcreteMaterials**);
```

#### C#

```
public IRJointConcreteMaterials Materials { get; }
```

#### Visual Basic

```
Public ReadOnly Materials As IRJointConcreteMaterials
```

#### Version

Available since version 7.5.

### II.3.2.6 Profile

#### C++

```
HRESULT get_Profile(IRJointProfile**);
```

#### C#

```
public IRJointProfile Profile { get; }
```

#### Visual Basic

```
Public ReadOnly Profile As IRJointProfile
```

### II.3.2.7 SpreadFootimeType

#### C++

```
HRESULT get_SpreadFootimeType(IRJointSpreadFootimeType*);  
HRESULT put_SpreadFootimeType(IRJointSpreadFootimeType);
```

#### C#

```
public IRJointSpreadFootimeType SpreadFootimeType { get; set; }
```

#### Visual Basic

```
Public SpreadFootimeType As IRJointSpreadFootimeType
```

### II.3.3 IRJointSpreadFootimeType

#### C++

```
enum IRJointSpreadFootimeType;
```

#### C#

```
public enum IRJointSpreadFootimeType;
```

#### Visual Basic

```
Public Enum IRJointSpreadFootimeType
```

#### Members

Members	Description
I_JSFT_RECT_CROSS_SECT = 1	Available since version 7.5.
I_JSFT_ABRUPT_CHANGES_HEIGHT = 2	Available since version 7.5.
I_JSFT_TAPERED = 3	Available since version 7.5.

#### Version

Available since version 7.5.

### II.3.4 IRJointConcreteColumnFoundation

#### Class Hierarchy

#### C++

```
interface IRJointConcreteColumnFoundation : IDispatch;
```

#### C#

```
public interface IRJointConcreteColumnFoundation;
```

#### Visual Basic

```
Public Interface IRJointConcreteColumnFoundation
```

## Version

Available since version 7.5.

### II.3.4.1 IRJointConcreteColumnFoundation Members

The following tables list the members exposed by IRJointConcreteColumnFoundation.

#### Public Fields

	Name	Description
◆	par_a1 ( [ see page 1985)	
◆	par_b ( [ see page 1985)	
◆	par_b1 ( [ see page 1985)	
◆	par_b2 ( [ see page 1986)	
◆	par_h ( [ see page 1986)	
◆	par_h1 ( [ see page 1986)	
◆	par_h2 ( [ see page 1986)	
◆	par_l ( [ see page 1987)	
◆	par_l1 ( [ see page 1987)	
◆	par_l2 ( [ see page 1987)	
◆	par_o1 ( [ see page 1987)	
◆	par_o2 ( [ see page 1988)	

### II.3.4.2 IRJointConcreteColumnFoundation Fields

The fields of the IRJointConcreteColumnFoundation class are listed here.

#### Public Fields

	Name	Description
◆	par_a1 ( [ see page 1985)	
◆	par_b ( [ see page 1985)	
◆	par_b1 ( [ see page 1985)	
◆	par_b2 ( [ see page 1986)	
◆	par_h ( [ see page 1986)	
◆	par_h1 ( [ see page 1986)	
◆	par_h2 ( [ see page 1986)	
◆	par_l ( [ see page 1987)	
◆	par_l1 ( [ see page 1987)	
◆	par_l2 ( [ see page 1987)	
◆	par_o1 ( [ see page 1987)	
◆	par_o2 ( [ see page 1988)	

### II.3.4.2.1 par\_a1

#### C++

```
HRESULT get_par_a1(double* );
HRESULT put_par_a1(double);
```

#### C#

```
public double par_a1 { get; set; }
```

#### Visual Basic

```
Public par_a1 As Double
```

**Version**

Available since version 7.5.

**II.3.4.2.2 par\_b****C++**

```
HRESULT get_par_b(double*);  
HRESULT put_par_b(double);
```

**C#**

```
public double par_b { get; set; }
```

**Visual Basic**

```
Public par_b As double
```

**Version**

Available since version 7.5.

**II.3.4.2.3 par\_b1****C++**

```
HRESULT get_par_b1(double*);  
HRESULT put_par_b1(double);
```

**C#**

```
public double par_b1 { get; set; }
```

**Visual Basic**

```
Public par_b1 As double
```

**Version**

Available since version 7.5.

**II.3.4.2.4 par\_b2****C++**

```
HRESULT get_par_b2(double*);  
HRESULT put_par_b2(double);
```

**C#**

```
public double par_b2 { get; set; }
```

**Visual Basic**

```
Public par_b2 As double
```

**Version**

Available since version 7.5.

**II.3.4.2.5 par\_h****C++**

```
HRESULT get_par_h(double*);  
HRESULT put_par_h(double);
```

**C#**

```
public double par_h { get; set; }
```

**Visual Basic**

```
Public par_h As double
```

**Version**

Available since version 7.5.

**II.3.4.2.6 par\_h1****C++**

```
HRESULT get_par_h1(double*);  
HRESULT put_par_h1(double);
```

**C#**

```
public double par_h1 { get; set; }
```

**Visual Basic**

```
Public par_h1 As double
```

**Version**

Available since version 7.5.

**II.3.4.2.7 par\_h2****C++**

```
HRESULT get_par_h2(double*);  
HRESULT put_par_h2(double);
```

**C#**

```
public double par_h2 { get; set; }
```

**Visual Basic**

```
Public par_h2 As double
```

**Version**

Available since version 7.5.

**II.3.4.2.8 par\_l****C++**

```
HRESULT get_par_l(double*);  
HRESULT put_par_l(double);
```

**C#**

```
public double par_l { get; set; }
```

**Visual Basic**

```
Public par_l As double
```

**Version**

Available since version 7.5.

**II.3.4.2.9 par\_l1****C++**

```
HRESULT get_par_l1(double*);  
HRESULT put_par_l1(double);
```

**C#**

```
public double par_l1 { get; set; }
```

**Visual Basic**

```
Public par_l1 As Double
```

**Version**

Available since version 7.5.

**II.3.4.2.10 par\_l2****C++**

```
HRESULT get_par_l2(double*);  
HRESULT put_par_l2(double);
```

**C#**

```
public double par_l2 { get; set; }
```

**Visual Basic**

```
Public par_l2 As Double
```

**Version**

Available since version 7.5.

**II.3.4.2.11 par\_o1****C++**

```
HRESULT get_par_o1(double*);  
HRESULT put_par_o1(double);
```

**C#**

```
public double par_o1 { get; set; }
```

**Visual Basic**

```
Public par_o1 As Double
```

**Version**

Available since version 7.5.

**II.3.4.2.12 par\_o2****C++**

```
HRESULT get_par_o2(double*);  
HRESULT put_par_o2(double);
```

**C#**

```
public double par_o2 { get; set; }
```

**Visual Basic**

```
Public par_o2 As Double
```

**Version**

Available since version 7.5.

## II.3.5 IRJointColumnBasePlateCalcModel

### C++

```
enum IRJointColumnBasePlateCalcModel;
```

### C#

```
public enum IRJointColumnBasePlateCalcModel;
```

### Visual Basic

```
Public Enum IRJointColumnBasePlateCalcModel
```

### Members

Members	Description
I_JCBPCM_PLASTIC = 0	Available since version 7.5.
I_JCBPCM_ELASTIC = 1	Available since version 7.5.

### Version

Available since version 7.5.

## II.3.6 IRJointConcreteColumnLoad

### Class Hierarchy

### C++

```
interface IRJointConcreteColumnLoad : IRJointLoad;
```

### C#

```
public interface IRJointConcreteColumnLoad : IRJointLoad;
```

### Visual Basic

```
Public Interface IRJointConcreteColumnLoad
```

### Version

Available since version 7.5.

### II.3.6.1 IRJointConcreteColumnLoad Members

The following tables list the members exposed by IRJointConcreteColumnLoad.

### Public Fields

	Name	Description
❖	Cases (see page 2166)	Selection of load cases, defined for the structure. .
❖	Fx (see page 1989)	
❖	FxAssemb (see page 1989)	
❖	Fy (see page 1990)	
❖	Fz (see page 1990)	
❖	My (see page 1990)	
❖	Mz (see page 1990)	
❖	Type (see page 2166)	Load type (defined manually or taken from Robot).

### II.3.6.2 IRJointConcreteColumnLoad Fields

The fields of the IRJointConcreteColumnLoad class are listed here.

## Public Fields

	Name	Description
◆	Fx (see page 1989)	
◆	FxAssemb (see page 1989)	
◆	Fy (see page 1990)	
◆	Fz (see page 1990)	
◆	My (see page 1990)	
◆	Mz (see page 1990)	

### II.3.6.2.1 Fx

#### C++

```
HRESULT get_Fx(double* );
HRESULT put_Fx(double);
```

#### C#

```
public double Fx { get; set; }
```

#### Visual Basic

```
Public Fx As double
```

#### Version

Available since version 7.5.

### II.3.6.2.2 FxAssemb

#### C++

```
HRESULT get_FxAssemb(double* );
HRESULT put_FxAssemb(double);
```

#### C#

```
public double FxAssemb { get; set; }
```

#### Visual Basic

```
Public FxAssemb As double
```

#### Version

Available since version 7.5.

### II.3.6.2.3 Fy

#### C++

```
HRESULT get_Fy(double* );
HRESULT put_Fy(double);
```

#### C#

```
public double Fy { get; set; }
```

#### Visual Basic

```
Public Fy As double
```

#### Version

Available since version 7.5.

### II.3.6.2.4 Fz

#### C++

```
HRESULT get_Fz(double*);  
HRESULT put_Fz(double);
```

#### C#

```
public double Fz { get; set; }
```

#### Visual Basic

```
Public Fz As double
```

#### Version

Available since version 7.5.

### II.3.6.2.5 My

#### C++

```
HRESULT get_My(double*);  
HRESULT put_My(double);
```

#### C#

```
public double My { get; set; }
```

#### Visual Basic

```
Public My As double
```

#### Version

Available since version 7.5.

### II.3.6.2.6 Mz

#### C++

```
HRESULT get_Mz(double*);  
HRESULT put_Mz(double);
```

#### C#

```
public double Mz { get; set; }
```

#### Visual Basic

```
Public Mz As double
```

#### Version

Available since version 7.5.

## II.4 IRJointAnchorBolt

### Class Hierarchy

#### C++

```
interface IRJointAnchorBolt : IDispatch;
```

#### C#

```
public interface IRJointAnchorBolt;
```

**Visual Basic**

```
Public Interface IRJointAnchorBolt
```

**Description**

Anchorage bolt.

**II.4.1 IRJointAnchorBolt Members**

The following tables list the members exposed by IRJointAnchorBolt.

**Public Fields**

	Name	Description
◆	Length1 (↗ see page 1991)	Available since version 1.7.
◆	Length2 (↗ see page 1992)	Available since version 1.7.
◆	Length3 (↗ see page 1992)	Available since version 1.7.
◆	Length4 (↗ see page 1992)	Available since version 1.7.

**II.4.2 IRJointAnchorBolt Fields**

The fields of the IRJointAnchorBolt class are listed here.

**Public Fields**

	Name	Description
◆	Length1 (↗ see page 1991)	Available since version 1.7.
◆	Length2 (↗ see page 1992)	Available since version 1.7.
◆	Length3 (↗ see page 1992)	Available since version 1.7.
◆	Length4 (↗ see page 1992)	Available since version 1.7.

**II.4.2.1 Length1****C++**

```
HRESULT get_Length1(double* );
HRESULT put_Length1(double);
```

**C#**

```
public double Length1 { get; set; }
```

**Visual Basic**

```
Public Length1 As double
```

**Description**

Available since version 1.7.

**II.4.2.2 Length2****C++**

```
HRESULT get_Length2(double* );
HRESULT put_Length2(double);
```

**C#**

```
public double Length2 { get; set; }
```

**Visual Basic**

```
Public Length2 As double
```

**Description**

Available since version 1.7.

**II.4.2.3 Length3****C++**

```
HRESULT get_Length3(double*);  
HRESULT put_Length3(double);
```

**C#**

```
public double Length3 { get; set; }
```

**Visual Basic**

```
Public Length3 As Double
```

**Description**

Available since version 1.7.

**II.4.2.4 Length4****C++**

```
HRESULT get_Length4(double*);  
HRESULT put_Length4(double);
```

**C#**

```
public double Length4 { get; set; }
```

**Visual Basic**

```
Public Length4 As Double
```

**Description**

Available since version 1.7.

**II.5 IRJointAnchorPlate****Class Hierarchy****C++**

```
interface IRJointAnchorPlate : IRJointPlate;
```

**C#**

```
public interface IRJointAnchorPlate : IRJointPlate;
```

**Visual Basic**

```
Public Interface IRJointAnchorPlate
```

**Description**

Plate of an anchor bolt.

**II.5.1 IRJointAnchorPlate Members**

The following tables list the members exposed by IRJointAnchorPlate.

## Public Fields

	Name	Description
◆	Exist ( [ see page 2152)	Information about a plate existance .
◆	Length ( [ see page 2152)	Plate length .
◆	Material ( [ see page 2152)	Name of the component material.
◆	Thick ( [ see page 2152)	Plate thickness.
◆	Type ( [ see page 1993)	Shape of a plate near anchor Available since version 1.7.
◆	Width ( [ see page 2153)	Plate width.

## Public Methods

	Name	Description
◆	AngleExt ( [ see page 2153)	Angle of bracket inclination, depending on the parameter - in degrees or radians.

## II.5.2 IRJointAnchorPlate Fields

The fields of the IRJointAnchorPlate class are listed here.

### Public Fields

	Name	Description
◆	Type ( [ see page 1993)	Shape of a plate near anchor Available since version 1.7.

### II.5.2.1 Type

#### C++

```
HRESULT get_Type(IRJointAnchorPlateType* );
HRESULT put_Type(IRJointAnchorPlateType);
```

#### C#

```
public IRJointAnchorPlateType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRJointAnchorPlateType
```

#### Description

Shape of a plate near anchor Available since version 1.7.

## II.6 IRJointAnchorType

#### C++

```
enum IRJointAnchorType;
```

#### C#

```
public enum IRJointAnchorType;
```

#### Visual Basic

```
Public Enum IRJointAnchorType
```

#### Members

Members	Description
I_JAT_ROD = 0	Available since version 1.7.
I_JAT_SIMPLE = 1	Available since version 1.7.
I_JAT_PIN = 2	Available since version 1.7.

I\_JAT\_PLATE = 3

Available since version 1.7.

**Description**

Available anchorage types.

## II.7 IRJointAnchor

**Class Hierarchy****C++**

```
interface IRJointAnchor : IDispatch;
```

**C#**

```
public interface IRJointAnchor;
```

**Visual Basic**

```
Public Interface IRJointAnchor
```

**Description**

Anchorage definition.

### II.7.1 IRJointAnchor Members

The following tables list the members exposed by IRJointAnchor.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AnchorPlate (see page 1995)	Plate of an anchor bolt Available since version 1.7.
❖	Bolts (see page 1995)	Arrangement and type of anchor bolts Available since version 1.7.
❖	Tige (see page 1995)	Parameters of anchor bolt Available since version 1.7.
❖	Type (see page 1995)	Anchorage type Available since version 1.7.

### II.7.2 IRJointAnchor Fields

The fields of the IRJointAnchor class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	AnchorPlate (see page 1995)	Plate of an anchor bolt Available since version 1.7.
❖	Bolts (see page 1995)	Arrangement and type of anchor bolts Available since version 1.7.
❖	Tige (see page 1995)	Parameters of anchor bolt Available since version 1.7.
❖	Type (see page 1995)	Anchorage type Available since version 1.7.

#### II.7.2.1 AnchorPlate

**C++**

```
HRESULT get_AnchorPlate(IRJointAnchorPlate**);
```

**C#**

```
public IRJointAnchorPlate AnchorPlate { get; }
```

**Visual Basic**

```
Public ReadOnly AnchorPlate As IRJointAnchorPlate
```

**Description**

Plate of an anchor bolt Available since version 1.7.

**II.7.2.2 Bolts****C++**

```
HRESULT get_Bolts(IRJointFootBolts**);
```

**C#**

```
public IRJointFootBolts Bolts { get; }
```

**Visual Basic**

```
Public ReadOnly Bolts As IRJointFootBolts
```

**Description**

Arrangement and type of anchor bolts Available since version 1.7.

**II.7.2.3 Tige****C++**

```
HRESULT get_Tige(IRJointAnchorBolt**);
```

**C#**

```
public IRJointAnchorBolt Tige { get; }
```

**Visual Basic**

```
Public ReadOnly Tige As IRJointAnchorBolt
```

**Description**

Parameters of anchor bolt Available since version 1.7.

**II.7.2.4 Type****C++**

```
HRESULT get_Type(IRJointAnchorType*);  
HRESULT put_Type(IRJointAnchorType*);
```

**C#**

```
public IRJointAnchorType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointAnchorType
```

**Description**

Anchorage type Available since version 1.7.

**II.8 IRJointFootBolts****Class Hierarchy****C++**

```
interface IRJointFootBolts : IDispatch;
```

**C#**

```
public interface IRJointFootBolts;
```

## Visual Basic

```
Public Interface IRJointFootBolts
```

### Description

Bolt group of identical parameters for pinned column base. .

## II.8.1 IRJointFootBolts Members

The following tables list the members exposed by IRJointFootBolts.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 1997</a> )	Available since version 1.7.
◆	ClassName ( <a href="#">see page 1997</a> )	Name of the bolt class Available since version 1.7.
◆	Diameter ( <a href="#">see page 1997</a> )	Bolt diameter Available since version 1.7.
◆	DiameterName ( <a href="#">see page 1997</a> )	Available since version 1.7.
◆	Distance ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	Friction ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	Rows ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	SpacingH ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	SpacingV ( <a href="#">see page 1999</a> )	Available since version 1.7.

## II.8.2 IRJointFootBolts Fields

The fields of the IRJointFootBolts class are listed here.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 1997</a> )	Available since version 1.7.
◆	ClassName ( <a href="#">see page 1997</a> )	Name of the bolt class Available since version 1.7.
◆	Diameter ( <a href="#">see page 1997</a> )	Bolt diameter Available since version 1.7.
◆	DiameterName ( <a href="#">see page 1997</a> )	Available since version 1.7.
◆	Distance ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	Friction ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	Rows ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	SpacingH ( <a href="#">see page 1998</a> )	Available since version 1.7.
◆	SpacingV ( <a href="#">see page 1999</a> )	Available since version 1.7.

### II.8.2.1 Area

#### C++

```
HRESULT get_Area(double*);
```

#### C#

```
public double Area { get; }
```

#### Visual Basic

```
Public ReadOnly Area As Double
```

### Description

Available since version 1.7.

### II.8.2.2 ClassName

#### C++

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

#### C#

```
public String ClassName { get; set; }
```

#### Visual Basic

```
Public ClassName As String
```

#### Description

Name of the bolt class Available since version 1.7.

### II.8.2.3 Diameter

#### C++

```
HRESULT get_Diameter(long* );
```

#### C#

```
public long Diameter { get; }
```

#### Visual Basic

```
Public ReadOnly Diameter As long
```

#### Description

Bolt diameter Available since version 1.7.

### II.8.2.4 DiameterName

#### C++

```
HRESULT get_DiameterName(BSTR* );
HRESULT put_DiameterName(BSTR);
```

#### C#

```
public String DiameterName { get; set; }
```

#### Visual Basic

```
Public DiameterName As String
```

#### Description

Available since version 1.7.

### II.8.2.5 Distance

#### C++

```
HRESULT get_Distance(double* );
HRESULT put_Distance(double);
```

#### C#

```
public double Distance { get; set; }
```

#### Visual Basic

```
Public Distance As double
```

**Description**

Available since version 1.7.

**II.8.2.6 Friction****C++**

```
HRESULT get_Friction(double*);  
HRESULT put_Friction(double);
```

**C#**

```
public double Friction { get; set; }
```

**Visual Basic**

```
Public Friction As Double
```

**Description**

Available since version 1.7.

**II.8.2.7 Rows****C++**

```
HRESULT get_Rows(long*);  
HRESULT put_Rows(long);
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Long
```

**Description**

Available since version 1.7.

**II.8.2.8 SpacingH****C++**

```
HRESULT get_SpacingH(double*);  
HRESULT put_SpacingH(double);
```

**C#**

```
public double SpacingH { get; set; }
```

**Visual Basic**

```
Public SpacingH As Double
```

**Description**

Available since version 1.7.

**II.8.2.9 SpacingV****C++**

```
HRESULT get_SpacingV(double*);  
HRESULT put_SpacingV(double);
```

**C#**

```
public double SpacingV { get; set; }
```

**Visual Basic**

```
Public SpacingV As double
```

**Description**

Available since version 1.7.

**II.9 IRJointFootPlateType****C++**

```
enum IRJointFootPlateType;
```

**C#**

```
public enum IRJointFootPlateType;
```

**Visual Basic**

```
Public Enum IRJointFootPlateType
```

**Members**

Members	Description
I_JFPT_NONE = 0	Available since version 1.7.
I_JFPT_WELD = 1	Available since version 1.7.
I_JFPT_ANGLE = 2	Available since version 1.7.

**Description**

Available connection types - plate/angle.

**II.10 IRJointFootPlate****Class Hierarchy****C++**

```
interface IRJointFootPlate : IRJointPlate;
```

**C#**

```
public interface IRJointFootPlate : IRJointPlate;
```

**Visual Basic**

```
Public Interface IRJointFootPlate
```

**II.10.1 IRJointFootPlate Members**

The following tables list the members exposed by IRJointFootPlate.

**Public Fields**

	Name	Description
◆	Diameter (↗ see page 2000)	Opening diameters in a plate for bolts Available since version 1.7.
◆	Exist (↗ see page 2152)	Information about a plate existance .
◆	Length (↗ see page 2152)	Plate length .
◆	Material (↗ see page 2152)	Name of the component material.
◆	Thick (↗ see page 2152)	Plate thickness.
◆	Type (↗ see page 2000)	Connection type - plate/angle Available since version 1.7.
◆	Width (↗ see page 2153)	Plate width.

## Public Methods

	Name	Description
AngleExt (see page 2153)		Angle of bracket inclination, depending on the parameter - in degrees or radians.

## II.10.2 IRJointFootPlate Fields

The fields of the IRJointFootPlate class are listed here.

### Public Fields

	Name	Description
Diameter (see page 2000)		Opening diameters in a plate for bolts Available since version 1.7.
Type (see page 2000)		Connection type - plate/angle Available since version 1.7.

### II.10.2.1 Diameter

#### C++

```
HRESULT get_Diameter(double* );
HRESULT put_Diameter(double);
```

#### C#

```
public double Diameter { get; set; }
```

#### Visual Basic

```
Public Diameter As Double
```

#### Description

Opening diameters in a plate for bolts Available since version 1.7.

### II.10.2.2 Type

#### C++

```
HRESULT get_Type(IRJointFootPlateType* );
HRESULT put_Type(IRJointFootPlateType);
```

#### C#

```
public IRJointFootPlateType Type { get; set; }
```

#### Visual Basic

```
Public Type As IRJointFootPlateType
```

#### Description

Connection type - plate/angle Available since version 1.7.

## II.11 IRJointWedge

### Class Hierarchy

#### C++

```
interface IRJointWedge : IDispatch;
```

#### C#

```
public interface IRJointWedge;
```

**Visual Basic**

```
Public Interface IRJointWedge
```

**II.11.1 IRJointWedge Members**

The following tables list the members exposed by IRJointWedge.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Length ( <a href="#">see page 2001</a> )	Available since version 1.7.
◆	Profile ( <a href="#">see page 2002</a> )	Wedge section Available since version 1.7.
◆	Thick ( <a href="#">see page 2002</a> )	Available since version 1.7.
◆	Type ( <a href="#">see page 2002</a> )	Wedge type Available since version 1.7.
◆	Width ( <a href="#">see page 2002</a> )	Available since version 1.7.
◆	XTypeMaterial ( <a href="#">see page 2003</a> )	Available since version 1.7.

**II.11.2 IRJointWedge Fields**

The fields of the IRJointWedge class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Length ( <a href="#">see page 2001</a> )	Available since version 1.7.
◆	Profile ( <a href="#">see page 2002</a> )	Wedge section Available since version 1.7.
◆	Thick ( <a href="#">see page 2002</a> )	Available since version 1.7.
◆	Type ( <a href="#">see page 2002</a> )	Wedge type Available since version 1.7.
◆	Width ( <a href="#">see page 2002</a> )	Available since version 1.7.
◆	XTypeMaterial ( <a href="#">see page 2003</a> )	Available since version 1.7.

**II.11.2.1 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Description**

Available since version 1.7.

**II.11.2.2 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Wedge section Available since version 1.7.

**II.11.2.3 Thick****C++**

```
HRESULT get_Thickness(double*);  
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thickness As Double
```

**Description**

Available since version 1.7.

**II.11.2.4 Type****C++**

```
HRESULT get_Type(IRJointWedgeType*);  
HRESULT put_Type(IRJointWedgeType);
```

**C#**

```
public IRJointWedgeType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointWedgeType
```

**Description**

Wedge type Available since version 1.7.

**II.11.2.5 Width****C++**

```
HRESULT get_Width(double*);  
HRESULT put_Width(double);
```

**C#**

```
public double Width { get; set; }
```

**Visual Basic**

```
Public Width As Double
```

**Description**

Available since version 1.7.

**II.11.2.6 XTypeMaterial****C++**

```
HRESULT get_XTypeMaterial(BSTR*);  
HRESULT put_XTypeMaterial(BSTR);
```

**C#**

```
public String XTypeMaterial { get; set; }
```

**Visual Basic**

```
Public XTypeMaterial As String
```

**Description**

Available since version 1.7.

**II.12 IRJointWedgeType****C++**

```
enum IRJointWedgeType;
```

**C#**

```
public enum IRJointWedgeType;
```

**Visual Basic**

```
Public Enum IRJointWedgeType
```

**Members**

Members	Description
I_JWT_NONE = 0	Available since version 1.7.
I_JWT_I = 1	Available since version 1.7.
I_JWT_L = 2	Available since version 1.7.
I_JWT_T = 3	Available since version 1.7.
I_JWT_X = 4	Available since version 1.7.

**Description**

Available wedge types.

**II.13 IRJointBearingPlate****Class Hierarchy****C++**

```
interface IRJointBearingPlate : IRJointPlate;
```

**C#**

```
public interface IRJointBearingPlate : IRJointPlate;
```

**Visual Basic**

```
Public Interface IRJointBearingPlate
```

**II.13.1 IRJointBearingPlate Members**

The following tables list the members exposed by IRJointBearingPlate.

**Public Fields**

	Name	Description
◆	Exist ( <a href="#">see page 2152</a> )	Information about a plate existance .
◆	Length ( <a href="#">see page 2152</a> )	Plate length .
◆	Material ( <a href="#">see page 2152</a> )	Name of the component material.
◆	Thick ( <a href="#">see page 2152</a> )	Plate thickness.
◆	ThickBearingBar ( <a href="#">see page 2004</a> )	Thickness of limit element Available since version 1.7.

	Width ( <a href="#">see page 2153</a> )	Plate width.
-----------------------------------------------------------------------------------	-----------------------------------------	--------------

## Public Methods

	Name	Description
	AngleExt ( <a href="#">see page 2153</a> )	Angle of bracket inclination, depending on the parameter - in degrees or radians.

## II.13.2 IRJointBearingPlate Fields

The fields of the IRJointBearingPlate class are listed here.

### Public Fields

	Name	Description
	ThickBearingBar ( <a href="#">see page 2004</a> )	Thickness of limit element Available since version 1.7.

### II.13.2.1 ThickBearingBar

#### C++

```
HRESULT get_ThickBearingBar(double* );
HRESULT put_ThickBearingBar(double);
```

#### C#

```
public double ThickBearingBar { get; set; }
```

#### Visual Basic

```
Public ThickBearingBar As Double
```

#### Description

Thickness of limit element Available since version 1.7.

## II.14 IRJointFootWelds

### Class Hierarchy

#### C++

```
interface IRJointFootWelds : IDispatch;
```

#### C#

```
public interface IRJointFootWelds;
```

#### Visual Basic

```
Public Interface IRJointFootWelds
```

#### Description

Description of connection element welds.

### II.14.1 IRJointFootWelds Members

The following tables list the members exposed by IRJointFootWelds.

#### Public Fields

	Name	Description
	Bearing ( <a href="#">see page 2005</a> )	Available since version 1.7.
	FootPlate ( <a href="#">see page 2005</a> )	Available since version 1.7.
	Stiff ( <a href="#">see page 2006</a> )	Available since version 1.7.

◆	Washer (see page 2006)	Available since version 1.7.
◆	Wedge (see page 2006)	Available since version 1.7.

## II.14.2 IRJointFootWelds Fields

The fields of the IRJointFootWelds class are listed here.

### Public Fields

	Name	Description
◆	Bearing (see page 2005)	Available since version 1.7.
◆	FootPlate (see page 2005)	Available since version 1.7.
◆	Stiff (see page 2006)	Available since version 1.7.
◆	Washer (see page 2006)	Available since version 1.7.
◆	Wedge (see page 2006)	Available since version 1.7.

### II.14.2.1 Bearing

#### C++

```
HRESULT get_Bearing(IRJointWeld**);
```

#### C#

```
public IRJointWeld Bearing { get; }
```

#### Visual Basic

```
Public ReadOnly Bearing As IRJointWeld
```

#### Description

Available since version 1.7.

### II.14.2.2 FootPlate

#### C++

```
HRESULT get_FootPlate(IRJointWeld**);
```

#### C#

```
public IRJointWeld FootPlate { get; }
```

#### Visual Basic

```
Public ReadOnly FootPlate As IRJointWeld
```

#### Description

Available since version 1.7.

### II.14.2.3 Stiff

#### C++

```
HRESULT get_Stiff(IRJointWeld**);
```

#### C#

```
public IRJointWeld Stiff { get; }
```

#### Visual Basic

```
Public ReadOnly Stiff As IRJointWeld
```

#### Description

Available since version 1.7.

## II.14.2.4 Washer

### C++

```
HRESULT get_Washer( IRJointWeld** );
```

### C#

```
public IRJointWeld Washer { get; }
```

### Visual Basic

```
Public ReadOnly Washer As IRJointWeld
```

### Description

Available since version 1.7.

## II.14.2.5 Wedge

### C++

```
HRESULT get_Wedge( IRJointWeld** );
```

### C#

```
public IRJointWeld Wedge { get; }
```

### Visual Basic

```
Public ReadOnly Wedge As IRJointWeld
```

### Description

Available since version 1.7.

## II.15 IRJointColumnBracket

### Class Hierarchy

### C++

```
interface IRJointColumnBracket : IDispatch;
```

### C#

```
public interface IRJointColumnBracket;
```

### Visual Basic

```
Public Interface IRJointColumnBracket
```

### Description

Column stiffener.

## II.15.1 IRJointColumnBracket Members

The following tables list the members exposed by IRJointColumnBracket.

### Public Fields

	Name	Description
◆	Height ( <a href="#">see page 2007</a> )	Available since version 1.7.
◆	Length ( <a href="#">see page 2008</a> )	Available since version 1.7.
◆	Thick ( <a href="#">see page 2008</a> )	Available since version 1.7.
◆	VThick ( <a href="#">see page 2008</a> )	Available since version 1.7.
◆	Width ( <a href="#">see page 2008</a> )	Available since version 1.7.

## Public Methods

	Name	Description
💡	Exist (see page 2009)	Function returns information about the stiffener existence. Available since version 1.7.

## II.15.2 IRJointColumnBracket Fields

The fields of the IRJointColumnBracket class are listed here.

### Public Fields

	Name	Description
💡	Height (see page 2007)	Available since version 1.7.
💡	Length (see page 2008)	Available since version 1.7.
💡	Thick (see page 2008)	Available since version 1.7.
💡	VThick (see page 2008)	Available since version 1.7.
💡	Width (see page 2008)	Available since version 1.7.

### II.15.2.1 Height

#### C++

```
HRESULT get_Height(double*);  
HRESULT put_Height(double);
```

#### C#

```
public double Height { get; set; }
```

#### Visual Basic

```
Public Height As double
```

#### Description

Available since version 1.7.

### II.15.2.2 Length

#### C++

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

#### C#

```
public double Length { get; set; }
```

#### Visual Basic

```
Public Length As double
```

#### Description

Available since version 1.7.

### II.15.2.3 Thick

#### C++

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

#### C#

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As double
```

**Description**

Available since version 1.7.

**II.15.2.4 VThick****C++**

```
HRESULT get_VThick(double*);  
HRESULT put_VThick(double);
```

**C#**

```
public double VThick { get; set; }
```

**Visual Basic**

```
Public VThick As double
```

**Description**

Available since version 1.7.

**II.15.2.5 Width****C++**

```
HRESULT get_Width(double*);  
HRESULT put_Width(double);
```

**C#**

```
public double Width { get; set; }
```

**Visual Basic**

```
Public Width As double
```

**Description**

Available since version 1.7.

**II.15.3 IRJointColumnBracket Methods**

The methods of the IRJointColumnBracket class are listed here.

**Public Methods**

	Name	Description
💡	Exist (↗ see page 2009)	Function returns information about the stiffener existence. Available since version 1.7.

**II.15.3.1 Exist****C++**

```
HRESULT Exist(VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist();
```

**Visual Basic**

```
Public Function Exist() As Boolean
```

## Description

Function returns information about the stiffener existence. Available since version 1.7.

## II.16 IRJointAnchorPlateType

### C++

```
enum IRJointAnchorPlateType;
```

### C#

```
public enum IRJointAnchorPlateType;
```

### Visual Basic

```
Public Enum IRJointAnchorPlateType
```

### Members

Members	Description
I_JAPT_RECT = 0	Available since version 1.7.
I_JAPT_TUBE = 1	Available since version 1.7.

### Description

Types of plates near anchor.

## II.17 IRJointFootStiffenerVert

### Class Hierarchy

### C++

```
interface IRJointFootStiffenerVert : IDispatch;
```

### C#

```
public interface IRJointFootStiffenerVert;
```

### Visual Basic

```
Public Interface IRJointFootStiffenerVert
```

### II.17.1 IRJointFootStiffenerVert Members

The following tables list the members exposed by IRJointFootStiffenerVert.

#### Public Fields

	Name	Description
◆	Length (see page 2010)	Available since version 1.7.
◆	Thick (see page 2010)	Available since version 1.7.
◆	WidthSpacing (see page 2011)	Available since version 1.7.

### II.17.2 IRJointFootStiffenerVert Fields

The fields of the IRJointFootStiffenerVert class are listed here.

#### Public Fields

	Name	Description
◆	Length (see page 2010)	Available since version 1.7.
◆	Thick (see page 2010)	Available since version 1.7.



WidthSpacing (see page 2011)

Available since version 1.7.

### II.17.2.1 Length

**C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Description**

Available since version 1.7.

### II.17.2.2 Thick

**C++**

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As double
```

**Description**

Available since version 1.7.

### II.17.2.3 WidthSpacing

**C++**

```
HRESULT get_WidthSpacing(double*);  
HRESULT put_WidthSpacing(double);
```

**C#**

```
public double WidthSpacing { get; set; }
```

**Visual Basic**

```
Public WidthSpacing As double
```

**Description**

Available since version 1.7.

## II.18 IRJointFootStiffenerHoriz

**Class Hierarchy****C++**

```
interface IRJointFootStiffenerHoriz : IDispatch;
```

**C#**

```
public interface IRJointFootStiffenerHoriz;
```

**Visual Basic**

```
Public Interface IRJointFootStiffenerHoriz
```

**II.18.1 IRJointFootStiffenerHoriz Members**

The following tables list the members exposed by IRJointFootStiffenerHoriz.

**Public Fields**

	Name	Description
◆	Thick (see page 2011)	Available since version 1.7.

**II.18.2 IRJointFootStiffenerHoriz Fields**

The fields of the IRJointFootStiffenerHoriz class are listed here.

**Public Fields**

	Name	Description
◆	Thick (see page 2011)	Available since version 1.7.

**II.18.2.1 Thick****C++**

```
HRESULT get_Thickness(double*);  
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thick As Double
```

**Description**

Available since version 1.7.

**II.19 IRJointFootMaterials****Class Hierarchy****C++**

```
interface IRJointFootMaterials : IDispatch;
```

**C#**

```
public interface IRJointFootMaterials;
```

**Visual Basic**

```
Public Interface IRJointFootMaterials
```

**II.19.1 IRJointFootMaterials Members**

The following tables list the members exposed by IRJointFootMaterials.

**Public Fields**

	Name	Description
◆	CementAmount (see page 2012)	Available since version 1.7.
◆	ConcrClass (see page 2013)	Available since version 1.7.
◆	ConcrSteelCoeff (see page 2013)	Available since version 1.7.

◆	PlateSigma (see page 2013)	Available since version 1.7.
◆	PlateYoung (see page 2013)	Available since version 1.7.

## II.19.2 IRJointFootMaterials Fields

The fields of the IRJointFootMaterials class are listed here.

### Public Fields

	Name	Description
◆	CementAmount (see page 2012)	Available since version 1.7.
◆	ConcrClass (see page 2013)	Available since version 1.7.
◆	ConcrSteelCoeff (see page 2013)	Available since version 1.7.
◆	PlateSigma (see page 2013)	Available since version 1.7.
◆	PlateYoung (see page 2013)	Available since version 1.7.

### II.19.2.1 CementAmount

#### C++

```
HRESULT get_CementAmount(double*);  
HRESULT put_CementAmount(double);
```

#### C#

```
public double CementAmount { get; set; }
```

#### Visual Basic

```
Public CementAmount As double
```

#### Description

Available since version 1.7.

### II.19.2.2 ConcrClass

#### C++

```
HRESULT get_ConcrClass(double*);  
HRESULT put_ConcrClass(double);
```

#### C#

```
public double ConcrClass { get; set; }
```

#### Visual Basic

```
Public ConcrClass As double
```

#### Description

Available since version 1.7.

### II.19.2.3 ConcrSteelCoeff

#### C++

```
HRESULT get_ConcrSteelCoeff(double*);  
HRESULT put_ConcrSteelCoeff(double);
```

#### C#

```
public double ConcrSteelCoeff { get; set; }
```

#### Visual Basic

```
Public ConcrSteelCoeff As double
```

**Description**

Available since version 1.7.

**II.19.2.4 PlateSigma****C++**

```
HRESULT get_PlateSigma(double* );
HRESULT put_PlateSigma(double);
```

**C#**

```
public double PlateSigma { get; set; }
```

**Visual Basic**

```
Public PlateSigma As Double
```

**Description**

Available since version 1.7.

**II.19.2.5 PlateYoung****C++**

```
HRESULT get_PlateYoung(double* );
HRESULT put_PlateYoung(double);
```

**C#**

```
public double PlateYoung { get; set; }
```

**Visual Basic**

```
Public PlateYoung As Double
```

**Description**

Available since version 1.7.

**II.20 IRJointColumnSquare****Class Hierarchy****C++**

```
interface IRJointColumnSquare : IDispatch;
```

**C#**

```
public interface IRJointColumnSquare;
```

**Visual Basic**

```
Public Interface IRJointColumnSquare
```

**II.20.1 IRJointColumnSquare Members**

The following tables list the members exposed by IRJointColumnSquare.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Diameter (↗ see page 2014)	Available since version 1.7.
◆	Length (↗ see page 2015)	Available since version 1.7.
◆	Profile (↗ see page 2015)	Available since version 1.7.

## II.20.2 IRJointColumnSquare Fields

The fields of the IRJointColumnSquare class are listed here.

### Public Fields

	Name	Description
◆	Diameter (see page 2014)	Available since version 1.7.
◆	Length (see page 2015)	Available since version 1.7.
◆	Profile (see page 2015)	Available since version 1.7.

### II.20.2.1 Diameter

#### C++

```
HRESULT get_Diameter(double* );
HRESULT put_Diameter(double);
```

#### C#

```
public double Diameter { get; set; }
```

#### Visual Basic

```
Public Diameter As double
```

#### Description

Available since version 1.7.

### II.20.2.2 Length

#### C++

```
HRESULT get_Length(double* );
HRESULT put_Length(double);
```

#### C#

```
public double Length { get; set; }
```

#### Visual Basic

```
Public Length As double
```

#### Description

Available since version 1.7.

### II.20.2.3 Profile

#### C++

```
HRESULT get_Profile(IRJointProfile** );
```

#### C#

```
public IRJointProfile Profile { get; }
```

#### Visual Basic

```
Public ReadOnly Profile As IRJointProfile
```

#### Description

Available since version 1.7.

## II.21 IRJointFootStiffenerSimple

### Class Hierarchy

#### C++

```
interface IRJointFootStiffenerSimple : IDispatch;
```

#### C#

```
public interface IRJointFootStiffenerSimple;
```

### Visual Basic

```
Public Interface IRJointFootStiffenerSimple
```

## II.21.1 IRJointFootStiffenerSimple Members

The following tables list the members exposed by IRJointFootStiffenerSimple.

### Public Fields

	Name	Description
◆	Height (↗ see page 2016)	Available since version 1.7.
◆	Length (↗ see page 2016)	Available since version 1.7.
◆	Thick (↗ see page 2016)	Available since version 1.7.
◆	Type (↗ see page 2017)	Available since version 1.7.
◆	Width (↗ see page 2017)	Available since version 1.7.

## II.21.2 IRJointFootStiffenerSimple Fields

The fields of the IRJointFootStiffenerSimple class are listed here.

### Public Fields

	Name	Description
◆	Height (↗ see page 2016)	Available since version 1.7.
◆	Length (↗ see page 2016)	Available since version 1.7.
◆	Thick (↗ see page 2016)	Available since version 1.7.
◆	Type (↗ see page 2017)	Available since version 1.7.
◆	Width (↗ see page 2017)	Available since version 1.7.

### II.21.2.1 Height

#### C++

```
HRESULT get_Height(double* );
HRESULT put_Height(double);
```

#### C#

```
public double Height { get; set; }
```

### Visual Basic

```
Public Height As Double
```

### Description

Available since version 1.7.

## II.21.2.2 Length

### C++

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

### C#

```
public double Length { get; set; }
```

### Visual Basic

```
Public Length As double
```

### Description

Available since version 1.7.

## II.21.2.3 Thick

### C++

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

### C#

```
public double Thick { get; set; }
```

### Visual Basic

```
Public Thick As double
```

### Description

Available since version 1.7.

## II.21.2.4 Type

### C++

```
HRESULT get_Type(IRJointFootStiffType*);  
HRESULT put_Type(IRJointFootStiffType);
```

### C#

```
public IRJointFootStiffType Type { get; set; }
```

### Visual Basic

```
Public Type As IRJointFootStiffType
```

### Description

Available since version 1.7.

## II.21.2.5 Width

### C++

```
HRESULT get_Width(double*);  
HRESULT put_Width(double);
```

### C#

```
public double Width { get; set; }
```

### Visual Basic

```
Public Width As double
```

**Description**

Available since version 1.7.

## II.22 IRJointFootStiffenerComplex

**Class Hierarchy****C++**

```
interface IRJointFootStiffenerComplex : IDispatch;
```

**C#**

```
public interface IRJointFootStiffenerComplex;
```

**Visual Basic**

```
Public Interface IRJointFootStiffenerComplex
```

### II.22.1 IRJointFootStiffenerComplex Members

The following tables list the members exposed by IRJointFootStiffenerComplex.

**Public Fields**

	Name	Description
◆	Height (see page 2018)	Available since version 1.7.
◆	Length (see page 2018)	Available since version 1.7.
◆	ThickPlateHor (see page 2018)	Available since version 1.7.
◆	ThickStiff (see page 2019)	Available since version 1.7.

### II.22.2 IRJointFootStiffenerComplex Fields

The fields of the IRJointFootStiffenerComplex class are listed here.

**Public Fields**

	Name	Description
◆	Height (see page 2018)	Available since version 1.7.
◆	Length (see page 2018)	Available since version 1.7.
◆	ThickPlateHor (see page 2018)	Available since version 1.7.
◆	ThickStiff (see page 2019)	Available since version 1.7.

#### II.22.2.1 Height

**C++**

```
HRESULT get_Height(double* );
HRESULT put_Height(double);
```

**C#**

```
public double Height { get; set; }
```

**Visual Basic**

```
Public Height As Double
```

**Description**

Available since version 1.7.

### II.22.2.2 Length

**C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Description**

Available since version 1.7.

### II.22.2.3 ThickPlateHor

**C++**

```
HRESULT get_ThickPlateHor(double*);  
HRESULT put_ThickPlateHor(double);
```

**C#**

```
public double ThickPlateHor { get; set; }
```

**Visual Basic**

```
Public ThickPlateHor As double
```

**Description**

Available since version 1.7.

### II.22.2.4 ThickStiff

**C++**

```
HRESULT get_ThickStiff(double*);  
HRESULT put_ThickStiff(double);
```

**C#**

```
public double ThickStiff { get; set; }
```

**Visual Basic**

```
Public ThickStiff As double
```

**Description**

Available since version 1.7.

## II.23 IRJointFootStiffType

**C++**

```
enum IRJointFootStiffType;
```

**C#**

```
public enum IRJointFootStiffType;
```

**Visual Basic**

```
Public Enum IRJointFootStiffType
```

## Members

Members	Description
I_JFST_1 = 1	Available since version 1.7.
I_JFST_2 = 2	Available since version 1.7.
I_JFST_3 = 3	Available since version 1.7.
I_JFST_4 = 4	Available since version 1.7.
I_JFST_5 = 5	Available since version 1.7.

## III Angle connection

### Enumerations

	Name	Description
☒	IRJointAngleType (see page 2025)	
☒	IRJointAngleProfilePosition (see page 2030)	

### Interfaces

	Name	Description
↪	IRJointAngleBolts (see page 2020)	
↪	IRJointAngle (see page 2023)	
↪	IRJointBeamCut (see page 2025)	Definition of beam cut-outs. .
↪	IRJointWithAngles (see page 2027)	Angle (see page 2028) connection.
↪	IRJointAngleLoad (see page 2030)	

### III.1 IRJointAngleBolts

#### Class Hierarchy

#### C++

```
interface IRJointAngleBolts : IDispatch;
```

#### C#

```
public interface IRJointAngleBolts;
```

#### Visual Basic

```
Public Interface IRJointAngleBolts
```

### III.1.1 IRJointAngleBolts Members

The following tables list the members exposed by IRJointAngleBolts.

#### Public Fields

	Name	Description
❖	Area (see page 2021)	Available since version 1.7.
❖	ClassName (see page 2021)	Available since version 1.7.
❖	Diameter (see page 2021)	Available since version 1.7.
❖	DiameterName (see page 2021)	Available since version 1.7.

◆	DistFromUpperEdge ( [ see page 2022) )	Available since version 1.7.
◆	DistFromWeb ( [ see page 2022) )	Available since version 1.7.
◆	Friction ( [ see page 2022) )	Available since version 1.7.
◆	Rows ( [ see page 2022) )	Available since version 1.7.
◆	Spacing ( [ see page 2023) )	Available since version 1.7.

### III.1.2 IRJointAngleBolts Fields

The fields of the IRJointAngleBolts class are listed here.

#### Public Fields

	Name	Description
◆	Area ( [ see page 2021) )	Available since version 1.7.
◆	ClassName ( [ see page 2021) )	Available since version 1.7.
◆	Diameter ( [ see page 2021) )	Available since version 1.7.
◆	DiameterName ( [ see page 2021) )	Available since version 1.7.
◆	DistFromUpperEdge ( [ see page 2022) )	Available since version 1.7.
◆	DistFromWeb ( [ see page 2022) )	Available since version 1.7.
◆	Friction ( [ see page 2022) )	Available since version 1.7.
◆	Rows ( [ see page 2022) )	Available since version 1.7.
◆	Spacing ( [ see page 2023) )	Available since version 1.7.

#### III.1.2.1 Area

##### C++

```
HRESULT get_Area(double*);
```

##### C#

```
public double Area { get; }
```

##### Visual Basic

```
Public ReadOnly Area As Double
```

##### Description

Available since version 1.7.

#### III.1.2.2 ClassName

##### C++

```
HRESULT get_ClassName(BSTR*);  
HRESULT put_ClassName(BSTR);
```

##### C#

```
public String ClassName { get; set; }
```

##### Visual Basic

```
Public ClassName As String
```

##### Description

Available since version 1.7.

### III.1.2.3 Diameter

**C++**

```
HRESULT get_Diameter(long*);
```

**C#**

```
public long Diameter { get; }
```

**Visual Basic**

```
Public ReadOnly Diameter As long
```

**Description**

Available since version 1.7.

### III.1.2.4 DiameterName

**C++**

```
HRESULT get_DiameterName(BSTR*);  
HRESULT put_DiameterName(BSTR);
```

**C#**

```
public String DiameterName { get; set; }
```

**Visual Basic**

```
Public DiameterName As String
```

**Description**

Available since version 1.7.

### III.1.2.5 DistFromUpperEdge

**C++**

```
HRESULT get_DistFromUpperEdge(double*);  
HRESULT put_DistFromUpperEdge(double);
```

**C#**

```
public double DistFromUpperEdge { get; set; }
```

**Visual Basic**

```
Public DistFromUpperEdge As double
```

**Description**

Available since version 1.7.

### III.1.2.6 DistFromWeb

**C++**

```
HRESULT get_DistFromWeb(double*);  
HRESULT put_DistFromWeb(double);
```

**C#**

```
public double DistFromWeb { get; set; }
```

**Visual Basic**

```
Public DistFromWeb As double
```

**Description**

Available since version 1.7.

**III.1.2.7 Friction****C++**

```
HRESULT get_Friction(double*);  
HRESULT put_Friction(double);
```

**C#**

```
public double Friction { get; set; }
```

**Visual Basic**

```
Public Friction As Double
```

**Description**

Available since version 1.7.

**III.1.2.8 Rows****C++**

```
HRESULT get_Rows(long*);  
HRESULT put_Rows(long);
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Long
```

**Description**

Available since version 1.7.

**III.1.2.9 Spacing****C++**

```
HRESULT get_Spacing(double*);  
HRESULT put_Spacing(double);
```

**C#**

```
public double Spacing { get; set; }
```

**Visual Basic**

```
Public Spacing As Double
```

**Description**

Available since version 1.7.

**III.2 IRJointAngle****Class Hierarchy****C++**

```
interface IRJointAngle : IDispatch;
```

**C#**

```
public interface IRJointAngle;
```

**Visual Basic**

```
Public Interface IRJointAngle
```

**III.2.1 IRJointAngle Members**

The following tables list the members exposed by IRJointAngle.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DistFromUpperBeamEdge (↗ see page 2024)	Distance from the top beam edge Available since version 1.7.
◆	Length (↗ see page 2024)	Angle height Available since version 1.7.
◆	Profile (↗ see page 2024)	Available since version 1.7.
◆	ProfilePosition (↗ see page 2025)	Available since version 1.7.
◆	Type (↗ see page 2025)	Available since version 1.7.

**III.2.2 IRJointAngle Fields**

The fields of the IRJointAngle class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	DistFromUpperBeamEdge (↗ see page 2024)	Distance from the top beam edge Available since version 1.7.
◆	Length (↗ see page 2024)	Angle height Available since version 1.7.
◆	Profile (↗ see page 2024)	Available since version 1.7.
◆	ProfilePosition (↗ see page 2025)	Available since version 1.7.
◆	Type (↗ see page 2025)	Available since version 1.7.

**III.2.2.1 DistFromUpperBeamEdge****C++**

```
HRESULT get_DistFromUpperBeamEdge(double* );
HRESULT put_DistFromUpperBeamEdge(double);
```

**C#**

```
public double DistFromUpperBeamEdge { get; set; }
```

**Visual Basic**

```
Public DistFromUpperBeamEdge As Double
```

**Description**

Distance from the top beam edge Available since version 1.7.

**III.2.2.2 Length****C++**

```
HRESULT get_Length(double* );
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Description**

Angle height Available since version 1.7.

**III.2.2.3 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available since version 1.7.

**III.2.2.4 ProfilePosition****C++**

```
HRESULT get_ProfilePosition(IRJointAngleProfilePosition*);  
HRESULT put_ProfilePosition(IRJointAngleProfilePosition);
```

**C#**

```
public IRJointAngleProfilePosition ProfilePosition { get; set; }
```

**Visual Basic**

```
Public ProfilePosition As IRJointAngleProfilePosition
```

**Description**

Available since version 1.7.

**III.2.2.5 Type****C++**

```
HRESULT get_Type(IRJointAngleType*);  
HRESULT put_Type(IRJointAngleType);
```

**C#**

```
public IRJointAngleType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointAngleType
```

**Description**

Available since version 1.7.

**III.3 IRJointAngleType****C++**

```
enum IRJointAngleType;
```

**C#**

```
public enum IRJointAngleType;
```

**Visual Basic**

```
Public Enum IRJointAngleType
```

**Members**

Members	Description
I_JST_ALONG_BEARER = 0	Available since version 1.7.
I_JST_ALONG_SECOND = 1	Available since version 1.7.

## III.4 IRJointBeamCut

**Class Hierarchy****C++**

```
interface IRJointBeamCut : IDispatch;
```

**C#**

```
public interface IRJointBeamCut;
```

**Visual Basic**

```
Public Interface IRJointBeamCut
```

**Description**

Definition of beam cut-outs. .

### III.4.1 IRJointBeamCut Members

The following tables list the members exposed by IRJointBeamCut.

**Public Fields**

	Name	Description
◆	HeightLower (see page 2026)	Length (see page 2027) of a beam cut-out (bottom) Available since version 1.7.
◆	HeightUpper (see page 2026)	Length (see page 2027) of a beam cut-out (top) Available since version 1.7.
◆	Length (see page 2027)	Cut-out length Available since version 1.7.

### III.4.2 IRJointBeamCut Fields

The fields of the IRJointBeamCut class are listed here.

**Public Fields**

	Name	Description
◆	HeightLower (see page 2026)	Length (see page 2027) of a beam cut-out (bottom) Available since version 1.7.
◆	HeightUpper (see page 2026)	Length (see page 2027) of a beam cut-out (top) Available since version 1.7.
◆	Length (see page 2027)	Cut-out length Available since version 1.7.

#### III.4.2.1 HeightLower

**C++**

```
HRESULT get_HeightLower(double*);
```

```

HRESULT put_HeightLower(double);

C#
public double HeightLower { get; set; }

```

**Visual Basic**

```
Public HeightLower As double
```

**Description**

Length (█ see page 2027) of a beam cut-out (bottom) Available since version 1.7.

**III.4.2.2 HeightUpper****C++**

```

HRESULT get_HeightUpper(double*);
HRESULT put_HeightUpper(double);

```

**C#**

```
public double HeightUpper { get; set; }
```

**Visual Basic**

```
Public HeightUpper As double
```

**Description**

Length (█ see page 2027) of a beam cut-out (top) Available since version 1.7.

**III.4.2.3 Length****C++**

```

HRESULT get_Length(double*);
HRESULT put_Length(double);

```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Description**

Cut-out length Available since version 1.7.

**III.5 IRJointWithAngles****Class Hierarchy****C++**

```
interface IRJointWithAngles : IRJointConnection;
```

**C#**

```
public interface IRJointWithAngles : IRJointConnection;
```

**Visual Basic**

```
Public Interface IRJointWithAngles
```

**Description**

Angle (█ see page 2028) connection.

### III.5.1 IRJointWithAngles Members

The following tables list the members exposed by IRJointWithAngles.

#### Public Fields

	Name	Description
❖	Angle ( <a href="#">see page 2028</a> )	Angle Available since version 1.7.
❖	BeamBolts ( <a href="#">see page 2028</a> )	Bolt group on the beam side Available since version 1.7.
❖	BeamCut ( <a href="#">see page 2029</a> )	Beam cut-out Available since version 1.7.
❖	BeamProfile ( <a href="#">see page 2029</a> )	Beam section Available since version 1.7.
❖	ColumnBolts ( <a href="#">see page 2029</a> )	Bolt group on the column side Available since version 1.7.
❖	ColumnProfile ( <a href="#">see page 2029</a> )	Column section Available since version 1.7.
❖	Distance ( <a href="#">see page 2030</a> )	Distance between a column and beam Available since version 1.7.
❖	Type ( <a href="#">see page 2148</a> )	A connection type.
❖	WFRelPos ( <a href="#">see page 2148</a> )	A type of relative position of a beam and a column.

#### Public Methods

	Name	Description
❖	GetFromRobot ( <a href="#">see page 2149</a> )	The function gets connection parameters from Robot and fills an object with the data.
❖	SetToRobot ( <a href="#">see page 2149</a> )	The function sets and saves connection parameter in Robot.

### III.5.2 IRJointWithAngles Fields

The fields of the IRJointWithAngles class are listed here.

#### Public Fields

	Name	Description
❖	Angle ( <a href="#">see page 2028</a> )	Angle Available since version 1.7.
❖	BeamBolts ( <a href="#">see page 2028</a> )	Bolt group on the beam side Available since version 1.7.
❖	BeamCut ( <a href="#">see page 2029</a> )	Beam cut-out Available since version 1.7.
❖	BeamProfile ( <a href="#">see page 2029</a> )	Beam section Available since version 1.7.
❖	ColumnBolts ( <a href="#">see page 2029</a> )	Bolt group on the column side Available since version 1.7.
❖	ColumnProfile ( <a href="#">see page 2029</a> )	Column section Available since version 1.7.
❖	Distance ( <a href="#">see page 2030</a> )	Distance between a column and beam Available since version 1.7.

#### III.5.2.1 Angle

##### C++

```
HRESULT get_Angle(IRJointAngle\*\*);
```

##### C#

```
public IRJointAngle Angle { get; }
```

##### Visual Basic

```
Public ReadOnly Angle As IRJointAngle
```

##### Description

Angle Available since version 1.7.

#### III.5.2.2 BeamBolts

##### C++

```
HRESULT get_BeamBolts(IRJointAngleBolts\*\*);
```

**C#**

```
public IRJointAngleBolts BeamBolts { get; }
```

**Visual Basic**

```
Public ReadOnly BeamBolts As IRJointAngleBolts
```

**Description**

Bolt group on the beam side Available since version 1.7.

### III.5.2.3 BeamCut

**C++**

```
HRESULT get_BeamCut(IRJointBeamCut**);
```

**C#**

```
public IRJointBeamCut BeamCut { get; }
```

**Visual Basic**

```
Public ReadOnly BeamCut As IRJointBeamCut
```

**Description**

Beam cut-out Available since version 1.7.

### III.5.2.4 BeamProfile

**C++**

```
HRESULT get_BeamProfile(IRJointProfile**);
```

**C#**

```
public IRJointProfile BeamProfile { get; }
```

**Visual Basic**

```
Public ReadOnly BeamProfile As IRJointProfile
```

**Description**

Beam section Available since version 1.7.

### III.5.2.5 ColumnBolts

**C++**

```
HRESULT get_ColumnBolts(IRJointAngleBolts**);
```

**C#**

```
public IRJointAngleBolts ColumnBolts { get; }
```

**Visual Basic**

```
Public ReadOnly ColumnBolts As IRJointAngleBolts
```

**Description**

Bolt group on the column side Available since version 1.7.

### III.5.2.6 ColumnProfile

**C++**

```
HRESULT get_ColumnProfile(IRJointProfile**);
```

**C#**

```
public IRJointProfile ColumnProfile { get; }
```

**Visual Basic**

```
Public ReadOnly ColumnProfile As IRJointProfile
```

**Description**

Column section Available since version 1.7.

**III.5.2.7 Distance****C++**

```
HRESULT get_Distance(double* );
HRESULT put_Distance(double);
```

**C#**

```
public double Distance { get; set; }
```

**Visual Basic**

```
Public Distance As Double
```

**Description**

Distance between a column and beam Available since version 1.7.

**III.6 IRJointAngleProfilePosition****C++**

```
enum IRJointAngleProfilePosition;
```

**C#**

```
public enum IRJointAngleProfilePosition;
```

**Visual Basic**

```
Public Enum IRJointAngleProfilePosition
```

**Members**

Members	Description
I_JAPP_LONG_FLANGE_TO_COLUMN = 0	Available since version 1.7.
I_JAPP_SHORT_FLANGE_TO_COLUMN = 1	Available since version 1.7.

**III.7 IRJointAngleLoad****Class Hierarchy****C++**

```
interface IRJointAngleLoad : IRJointLoad;
```

**C#**

```
public interface IRJointAngleLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointAngleLoad
```

### III.7.1 IRJointAngleLoad Members

The following tables list the members exposed by IRJointAngleLoad.

#### Public Fields

	Name	Description
❖	Cases ( [ see page 2166)	Selection of load cases, defined for the structure. .
❖	Fz ( [ see page 2031)	Available since version 1.7.
❖	Type ( [ see page 2166)	Load type (defined manually or taken from Robot).

### III.7.2 IRJointAngleLoad Fields

The fields of the IRJointAngleLoad class are listed here.

#### Public Fields

	Name	Description
❖	Fz ( [ see page 2031)	Available since version 1.7.

#### III.7.2.1 Fz

##### C++

```
HRESULT get_Fz(double* );
HRESULT put_Fz(double);
```

##### C#

```
public double Fz { get; set; }
```

##### Visual Basic

```
Public Fz As double
```

##### Description

Available since version 1.7.

## IV Tube connection

#### Enumerations

	Name	Description
❖	IRJointTubeType ( [ see page 2036)	
❖	IRJointTubeProfileType ( [ see page 2036)	

#### Interfaces

	Name	Description
↪	IRJointTube ( [ see page 2031)	
↪	IRJointTubeFlangeProfile ( [ see page 2036)	
↪	IRJointTubeDiagProfile ( [ see page 2039)	
↪	IRJointTubePostProfile ( [ see page 2042)	
↪	IRJointTubeLoad ( [ see page 2045)	

## IV.1 IRJointTube

### Class Hierarchy

#### C++

```
interface IRJointTube : IRJointConnection;
```

#### C#

```
public interface IRJointTube : IRJointConnection;
```

### Visual Basic

```
Public Interface IRJointTube
```

### IV.1.1 IRJointTube Members

The following tables list the members exposed by IRJointTube.

#### Public Fields

	Name	Description
◆	DiagLeftLower (see page 2033)	Available since version 1.7.
◆	DiagLeftUpper (see page 2033)	Section of a first diagonal Available since version 1.7.
◆	DiagRightUpper (see page 2033)	Available since version 1.7.
◆	Flange (see page 2033)	Chord section Available since version 1.7.
◆	PostUpper (see page 2034)	Available since version 1.7.
◆	StiffHBracketMaterial (see page 2034)	Available since version ARSA 2010.
◆	StiffHoriz (see page 2034)	Available since version 1.7.
◆	StiffLateral (see page 2035)	Available since version 1.7.
◆	TubeType (see page 2035)	Type (see page 2148) of tube connection Available since version 1.7.
◆	Type (see page 2148)	A connection type.
◆	WeldDiag (see page 2035)	Available since version 1.7.
◆	WeldStiff (see page 2035)	Available since version 1.7.
◆	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

#### Public Methods

	Name	Description
◆	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
◆	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

### IV.1.2 IRJointTube Fields

The fields of the IRJointTube class are listed here.

#### Public Fields

	Name	Description
◆	DiagLeftLower (see page 2033)	Available since version 1.7.
◆	DiagLeftUpper (see page 2033)	Section of a first diagonal Available since version 1.7.
◆	DiagRightUpper (see page 2033)	Available since version 1.7.
◆	Flange (see page 2033)	Chord section Available since version 1.7.
◆	PostUpper (see page 2034)	Available since version 1.7.
◆	StiffHBracketMaterial (see page 2034)	Available since version ARSA 2010.

◆	StiffHoriz (see page 2034)	Available since version 1.7.
◆	StiffLateral (see page 2035)	Available since version 1.7.
◆	TubeType (see page 2035)	Type (see page 2148) of tube connection Available since version 1.7.
◆	WeldDiag (see page 2035)	Available since version 1.7.
◆	WeldStiff (see page 2035)	Available since version 1.7.

#### IV.1.2.1 DiagLeftLower

##### C++

```
HRESULT get_DiagLeftLower(IRJointTubeDiagProfile**);
```

##### C#

```
public IRJointTubeDiagProfile DiagLeftLower { get; }
```

##### Visual Basic

```
Public ReadOnly DiagLeftLower As IRJointTubeDiagProfile
```

##### Description

Available since version 1.7.

#### IV.1.2.2 DiagLeftUpper

##### C++

```
HRESULT get_DiagLeftUpper(IRJointTubeDiagProfile**);
```

##### C#

```
public IRJointTubeDiagProfile DiagLeftUpper { get; }
```

##### Visual Basic

```
Public ReadOnly DiagLeftUpper As IRJointTubeDiagProfile
```

##### Description

Section of a first diagonal Available since version 1.7.

#### IV.1.2.3 DiagRightUpper

##### C++

```
HRESULT get_DiagRightUpper(IRJointTubeDiagProfile**);
```

##### C#

```
public IRJointTubeDiagProfile DiagRightUpper { get; }
```

##### Visual Basic

```
Public ReadOnly DiagRightUpper As IRJointTubeDiagProfile
```

##### Description

Available since version 1.7.

#### IV.1.2.4 Flange

##### C++

```
HRESULT get_Flange(IRJointTubeFlangeProfile**);
```

##### C#

```
public IRJointTubeFlangeProfile Flange { get; }
```

**Visual Basic**

```
Public ReadOnly Flange As IRJointTubeFlangeProfile
```

**Description**

Chord section Available since version 1.7.

**IV.1.2.5 PostUpper****C++**

```
HRESULT get_PostUpper(IRJointTubePostProfile**);
```

**C#**

```
public IRJointTubePostProfile PostUpper { get; }
```

**Visual Basic**

```
Public ReadOnly PostUpper As IRJointTubePostProfile
```

**Description**

Available since version 1.7.

**IV.1.2.6 StiffHBracketMaterial****C++**

```
HRESULT get_StiffHBracketMaterial( BSTR* );
HRESULT put_StiffHBracketMaterial( BSTR );
```

**C#**

```
public BSTR StiffHBracketMaterial { get; set; }
```

**Visual Basic**

```
Public StiffHBracketMaterial As BSTR
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**IV.1.2.7 StiffHoriz****C++**

```
HRESULT get_StiffHoriz(IRJointPlate**);
```

**C#**

```
public IRJointPlate StiffHoriz { get; }
```

**Visual Basic**

```
Public ReadOnly StiffHoriz As IRJointPlate
```

**Description**

Available since version 1.7.

**IV.1.2.8 StiffLateral****C++**

```
HRESULT get_StiffLateral(IRJointPlate**);
```

**C#**

```
public IRJointPlate StiffLateral { get; }
```

**Visual Basic**

```
Public ReadOnly StiffLateral As IRJointPlate
```

**Description**

Available since version 1.7.

**IV.1.2.9 TubeType****C++**

```
HRESULT get_TubeType(IRJointTubeType* );
HRESULT put_TubeType(IRJointTubeType* );
```

**C#**

```
public IRJointTubeType TubeType { get; set; }
```

**Visual Basic**

```
Public TubeType As IRJointTubeType
```

**Description**

Type (see page 2148) of tube connection Available since version 1.7.

**IV.1.2.10 WeldDiag****C++**

```
HRESULT get_WeldDiag(IRJointWeld** );
```

**C#**

```
public IRJointWeld WeldDiag { get; }
```

**Visual Basic**

```
Public ReadOnly WeldDiag As IRJointWeld
```

**Description**

Available since version 1.7.

**IV.1.2.11 WeldStiff****C++**

```
HRESULT get_WeldStiff(IRJointWeld** );
```

**C#**

```
public IRJointWeld WeldStiff { get; }
```

**Visual Basic**

```
Public ReadOnly WeldStiff As IRJointWeld
```

**Description**

Available since version 1.7.

**IV.2 IRJointTubeType****C++**

```
enum IRJointTubeType;
```

**C#**

```
public enum IRJointTubeType;
```

**Visual Basic**

```
Public Enum IRJointTubeType
```

**Members**

Members	Description
I_JTT_T = 0	Available since version 1.7.
I_JTT_Y = 1	Available since version 1.7.
I_JTT_X = 2	Available since version 1.7.
I_JTT_N = 3	Available since version 1.7.
I_JTT_K = 4	Available since version 1.7.
I_JTT_KT = 5	Available since version 1.7.
I_JTT_NONE = -1	Available since version 1.7.

**IV.3 IRJointTubeProfileType****C++**

```
enum IRJointTubeProfileType;
```

**C#**

```
public enum IRJointTubeProfileType;
```

**Visual Basic**

```
Public Enum IRJointTubeProfileType
```

**Members**

Members	Description
I_JPTT_RECT = 0	Available since version 1.7.
I_JPTT_TUBE = 1	Available since version 1.7.
I_JPTT_RECT_RECT = 2	Available since version 1.7.
I_JPTT_RECT_TUBE = 3	Available since version 1.7.
I_JPTT_TUBE_TUBE = 4	Available since version 1.7.
I_JPTT_UNKNOWN = -1	Available since version 1.7.

**IV.4 IRJointTubeFlangeProfile****Class Hierarchy****C++**

```
interface IRJointTubeFlangeProfile : IDispatch;
```

**C#**

```
public interface IRJointTubeFlangeProfile;
```

**Visual Basic**

```
Public Interface IRJointTubeFlangeProfile
```

**IV.4.1 IRJointTubeFlangeProfile Members**

The following tables list the members exposed by IRJointTubeFlangeProfile.

## Public Fields

	Name	Description
◆	BarNumber ( <a href="#">see page 2037</a> )	Bar number.
◆	Excentr ( <a href="#">see page 2037</a> )	Available since version 1.7.
◆	Exist ( <a href="#">see page 2038</a> )	
◆	Material ( <a href="#">see page 2038</a> )	Available since version 1.7.
◆	Profile ( <a href="#">see page 2038</a> )	Available since version ARSA 2010.
◆	Section ( <a href="#">see page 2039</a> )	Available since version 1.7.

## IV.4.2 IRJointTubeFlangeProfile Fields

The fields of the IRJointTubeFlangeProfile class are listed here.

### Public Fields

	Name	Description
◆	BarNumber ( <a href="#">see page 2037</a> )	Bar number.
◆	Excentr ( <a href="#">see page 2037</a> )	Available since version 1.7.
◆	Exist ( <a href="#">see page 2038</a> )	
◆	Material ( <a href="#">see page 2038</a> )	Available since version 1.7.
◆	Profile ( <a href="#">see page 2038</a> )	Available since version ARSA 2010.
◆	Section ( <a href="#">see page 2039</a> )	Available since version 1.7.

### IV.4.2.1 BarNumber

#### C++

```
HRESULT get_BarNumber(long*);  
HRESULT put_BarNumber(long);
```

#### C#

```
public long BarNumber { get; set; }
```

#### Visual Basic

```
Public BarNumber As long
```

#### Description

Bar number.

#### Version

Available since version 3.5.

### IV.4.2.2 Excentr

#### C++

```
HRESULT get_Excentr(double*);  
HRESULT put_Excentr(double);
```

#### C#

```
public double Excentr { get; set; }
```

#### Visual Basic

```
Public Excentr As double
```

#### Description

Available since version 1.7.

#### IV.4.2.3 Exist

**C++**

```
HRESULT get_Exist(VARIANT_BOOL* );
```

**C#**

```
public bool Exist { get; }
```

**Visual Basic**

```
Public ReadOnly Exist As Boolean
```

**Version**

Available since version 4.5.

#### IV.4.2.4 Material

**C++**

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Available since version 1.7.

#### IV.4.2.5 Profile

**C++**

```
HRESULT get_Profile(IRJointProfile** );
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

#### IV.4.2.6 Section

**C++**

```
HRESULT get_Section(BSTR* );
HRESULT put_Section(BSTR);
```

**C#**

```
public String Section { get; set; }
```

**Visual Basic**

```
Public Section As String
```

## Description

Available since version 1.7.

## IV.5 IRJointTubeDiagProfile

### Class Hierarchy

#### C++

```
interface IRJointTubeDiagProfile : IDispatch;
```

#### C#

```
public interface IRJointTubeDiagProfile;
```

### Visual Basic

```
Public Interface IRJointTubeDiagProfile
```

## IV.5.1 IRJointTubeDiagProfile Members

The following tables list the members exposed by IRJointTubeDiagProfile.

### Public Fields

	Name	Description
◆	Angle (↗ see page 2040)	Available since version 1.7.
◆	BarNumber (↗ see page 2040)	Bar number.
◆	Exist (↗ see page 2040)	
◆	Gap (↗ see page 2041)	Available since version 1.7.
◆	Length (↗ see page 2041)	Available since version 1.7.
◆	Material (↗ see page 2041)	Available since version 1.7.
◆	Overlap (↗ see page 2041)	Available since version 1.7.
◆	Profile (↗ see page 2042)	Available from version ARSA 2010.
◆	Section (↗ see page 2042)	Available since version 1.7.

## IV.5.2 IRJointTubeDiagProfile Fields

The fields of the IRJointTubeDiagProfile class are listed here.

### Public Fields

	Name	Description
◆	Angle (↗ see page 2040)	Available since version 1.7.
◆	BarNumber (↗ see page 2040)	Bar number.
◆	Exist (↗ see page 2040)	
◆	Gap (↗ see page 2041)	Available since version 1.7.
◆	Length (↗ see page 2041)	Available since version 1.7.
◆	Material (↗ see page 2041)	Available since version 1.7.
◆	Overlap (↗ see page 2041)	Available since version 1.7.
◆	Profile (↗ see page 2042)	Available from version ARSA 2010.
◆	Section (↗ see page 2042)	Available since version 1.7.

### IV.5.2.1 Angle

#### C++

```
HRESULT get_Angle(double* );
HRESULT put_Angle(double);
```

**C#**

```
public double Angle { get; set; }
```

**Visual Basic**

```
Public Angle As Double
```

**Description**

Available since version 1.7.

**IV.5.2.2 BarNumber****C++**

```
HRESULT get_BarNumber(long*);  
HRESULT put_BarNumber(long);
```

**C#**

```
public long BarNumber { get; set; }
```

**Visual Basic**

```
Public BarNumber As Long
```

**Description**

Bar number.

**Version**

Available since version 3.5.

**IV.5.2.3 Exist****C++**

```
HRESULT get_Exist(VARIANT_BOOL*);
```

**C#**

```
public bool Exist { get; }
```

**Visual Basic**

```
Public ReadOnly Exist As Boolean
```

**Version**

Available since version 4.5.

**IV.5.2.4 Gap****C++**

```
HRESULT get_Gap(double*);  
HRESULT put_Gap(double);
```

**C#**

```
public double Gap { get; set; }
```

**Visual Basic**

```
Public Gap As Double
```

**Description**

Available since version 1.7.

#### IV.5.2.5 Length

##### C++

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

##### C#

```
public double Length { get; set; }
```

##### Visual Basic

```
Public Length As Double
```

##### Description

Available since version 1.7.

#### IV.5.2.6 Material

##### C++

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

##### C#

```
public String Material { get; set; }
```

##### Visual Basic

```
Public Material As String
```

##### Description

Available since version 1.7.

#### IV.5.2.7 Overlap

##### C++

```
HRESULT get_Overlap(double*);  
HRESULT put_Overlap(double);
```

##### C#

```
public double Overlap { get; set; }
```

##### Visual Basic

```
Public Overlap As Double
```

##### Description

Available since version 1.7.

#### IV.5.2.8 Profile

##### C++

```
HRESULT get_Profile(IRJointProfile**);
```

##### C#

```
public IRJointProfile Profile { get; }
```

##### Visual Basic

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available from version ARSA 2010.

**Version**

Available since version 10.

**IV.5.2.9 Section****C++**

```
HRESULT get_Section(BSTR* );
HRESULT put_Section(BSTR);
```

**C#**

```
public String Section { get; set; }
```

**Visual Basic**

```
Public Section As String
```

**Description**

Available since version 1.7.

**IV.6 IRJointTubePostProfile****Class Hierarchy****C++**

```
interface IRJointTubePostProfile : IDispatch;
```

**C#**

```
public interface IRJointTubePostProfile;
```

**Visual Basic**

```
Public Interface IRJointTubePostProfile
```

**IV.6.1 IRJointTubePostProfile Members**

The following tables list the members exposed by IRJointTubePostProfile.

**Public Fields**

	Name	Description
❖	BarNumber (↗ see page 2043)	
❖	Exist (↗ see page 2043)	
❖	Length (↗ see page 2043)	Available since version 1.7.
❖	Material (↗ see page 2044)	Available since version 1.7.
❖	Profile (↗ see page 2044)	Available since version ARSA 2010.
❖	Section (↗ see page 2044)	Available since version 1.7.

**IV.6.2 IRJointTubePostProfile Fields**

The fields of the IRJointTubePostProfile class are listed here.

**Public Fields**

	Name	Description
❖	BarNumber (↗ see page 2043)	
❖	Exist (↗ see page 2043)	

◆	Length (see page 2043)	Available since version 1.7.
◆	Material (see page 2044)	Available since version 1.7.
◆	Profile (see page 2044)	Available since ARSA 2010.
◆	Section (see page 2044)	Available since version 1.7.

#### IV.6.2.1 BarNumber

##### C++

```
HRESULT get_BarNumber(long* );
HRESULT put_BarNumber(long);
```

##### C#

```
public long BarNumber { get; set; }
```

##### Visual Basic

```
Public BarNumber As long
```

##### Version

Available since version 4.5.

#### IV.6.2.2 Exist

##### C++

```
HRESULT get_Exist(VARIANT_BOOL* );
```

##### C#

```
public bool Exist { get; }
```

##### Visual Basic

```
Public ReadOnly Exist As Boolean
```

##### Version

Available since version 4.5.

#### IV.6.2.3 Length

##### C++

```
HRESULT get_Length(double* );
HRESULT put_Length(double);
```

##### C#

```
public double Length { get; set; }
```

##### Visual Basic

```
Public Length As double
```

##### Description

Available since version 1.7.

#### IV.6.2.4 Material

##### C++

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

##### C#

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Available since version 1.7.

**IV.6.2.5 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**IV.6.2.6 Section****C++**

```
HRESULT get_Section(BSTR*);  
HRESULT put_Section(BSTR);
```

**C#**

```
public String Section { get; set; }
```

**Visual Basic**

```
Public Section As String
```

**Description**

Available since version 1.7.

**IV.7 IRJointTubeLoad****Class Hierarchy****C++**

```
interface IRJointTubeLoad : IRJointLoad;
```

**C#**

```
public interface IRJointTubeLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointTubeLoad
```

**IV.7.1 IRJointTubeLoad Members**

The following tables list the members exposed by IRJointTubeLoad.

## Public Fields

	Name	Description
❖	Cases ( <a href="#">see page 2166</a> )	Selection of load cases, defined for the structure. .
❖	DiagLeftLowerM ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftLowerN ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftUpperM ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftUpperN ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagRightUpperM ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	DiagRightUpperN ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	FlangeM ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	FlangeN ( <a href="#">see page 2048</a> )	Available since version 1.7.
❖	FlangeQ ( <a href="#">see page 2048</a> )	Available since version 1.7.
❖	PostUpperN ( <a href="#">see page 2048</a> )	Available since version 1.7.
❖	Type ( <a href="#">see page 2166</a> )	Load type (defined manually or taken from Robot).

## IV.7.2 IRJointTubeLoad Fields

The fields of the IRJointTubeLoad class are listed here.

### Public Fields

	Name	Description
❖	DiagLeftLowerM ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftLowerN ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftUpperM ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagLeftUpperN ( <a href="#">see page 2046</a> )	Available since version 1.7.
❖	DiagRightUpperM ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	DiagRightUpperN ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	FlangeM ( <a href="#">see page 2047</a> )	Available since version 1.7.
❖	FlangeN ( <a href="#">see page 2048</a> )	Available since version 1.7.
❖	FlangeQ ( <a href="#">see page 2048</a> )	Available since version 1.7.
❖	PostUpperN ( <a href="#">see page 2048</a> )	Available since version 1.7.

### IV.7.2.1 DiagLeftLowerM

#### C++

```
HRESULT get_DiagLeftLowerM(double* );
HRESULT put_DiagLeftLowerM(double);
```

#### C#

```
public double DiagLeftLowerM { get; set; }
```

#### Visual Basic

```
Public DiagLeftLowerM As Double
```

#### Description

Available since version 1.7.

#### IV.7.2.2 DiagLeftLowerN

**C++**

```
HRESULT get_DiagLeftLowerN(double*);  
HRESULT put_DiagLeftLowerN(double);
```

**C#**

```
public double DiagLeftLowerN { get; set; }
```

**Visual Basic**

```
Public DiagLeftLowerN As double
```

**Description**

Available since version 1.7.

#### IV.7.2.3 DiagLeftUpperM

**C++**

```
HRESULT get_DiagLeftUpperM(double*);  
HRESULT put_DiagLeftUpperM(double);
```

**C#**

```
public double DiagLeftUpperM { get; set; }
```

**Visual Basic**

```
Public DiagLeftUpperM As double
```

**Description**

Available since version 1.7.

#### IV.7.2.4 DiagLeftUpperN

**C++**

```
HRESULT get_DiagLeftUpperN(double*);  
HRESULT put_DiagLeftUpperN(double);
```

**C#**

```
public double DiagLeftUpperN { get; set; }
```

**Visual Basic**

```
Public DiagLeftUpperN As double
```

**Description**

Available since version 1.7.

#### IV.7.2.5 DiagRightUpperM

**C++**

```
HRESULT get_DiagRightUpperM(double*);  
HRESULT put_DiagRightUpperM(double);
```

**C#**

```
public double DiagRightUpperM { get; set; }
```

**Visual Basic**

```
Public DiagRightUpperM As double
```

**Description**

Available since version 1.7.

**IV.7.2.6 DiagRightUpperN****C++**

```
HRESULT get_DiagRightUpperN(double*);  
HRESULT put_DiagRightUpperN(double);
```

**C#**

```
public double DiagRightUpperN { get; set; }
```

**Visual Basic**

```
Public DiagRightUpperN As Double
```

**Description**

Available since version 1.7.

**IV.7.2.7 FlangeM****C++**

```
HRESULT get_FlangeM(double*);  
HRESULT put_FlangeM(double);
```

**C#**

```
public double FlangeM { get; set; }
```

**Visual Basic**

```
Public FlangeM As Double
```

**Description**

Available since version 1.7.

**IV.7.2.8 FlangeN****C++**

```
HRESULT get_FlangeN(double*);  
HRESULT put_FlangeN(double);
```

**C#**

```
public double FlangeN { get; set; }
```

**Visual Basic**

```
Public FlangeN As Double
```

**Description**

Available since version 1.7.

**IV.7.2.9 FlangeQ****C++**

```
HRESULT get_FlangeQ(double*);  
HRESULT put_FlangeQ(double);
```

**C#**

```
public double FlangeQ { get; set; }
```

**Visual Basic**

```
Public FlangeQ As double
```

**Description**

Available since version 1.7.

**IV.7.2.10 PostUpperN****C++**

```
HRESULT get_PostUpperN(double* );
HRESULT put_PostUpperN(double);
```

**C#**

```
public double PostUpperN { get; set; }
```

**Visual Basic**

```
Public PostUpperN As double
```

**Description**

Available since version 1.7.

**V Gusset plate connections****Enumerations**

	<b>Name</b>	<b>Description</b>
	IRJointGussetFixType (see page 2057)	
	IRJointGussetSimpleProfilePosition (see page 2057)	
	IRJointGussetCornersType (see page 2067)	
	IRJointGussetCrossProfileCutting (see page 2075)	
	IRJointGussetFlangeProfileCutting (see page 2090)	
	IRJointGussetDiagonalePositionType (see page 2111)	
	IRJointGussetFlangePlateRegularType (see page 2111)	

**Interfaces**

	<b>Name</b>	<b>Description</b>
	IRJointGussetBoltsDiag (see page 2049)	
	IRJointGussetWeldsDiag (see page 2052)	
	IRJointGussetDiagonale (see page 2054)	
	IRJointGussetSimple (see page 2057)	
	IRJointGussetSimplePlate (see page 2061)	
	IRJointGussetSimpleAttachment (see page 2067)	

IRJointGussetSimpleAttachBolts (see page 2069)	
IRJointGussetSimpleAttachBoltsHorizontal (see page 2071)	
IRJointGussetSimpleAttachBoltsVertical (see page 2072)	
IRJointGussetSimpleAttachWelds (see page 2074)	
IRJointGussetCrossPlate (see page 2076)	
IRJointGussetCross (see page 2084)	
IRJointGussetFlangePlate (see page 2090)	
IRJointGussetFlange (see page 2100)	
IRJointGussetSimpleLoad (see page 2106)	
IRJointGussetCrossLoad (see page 2107)	
IRJointGussetFlangeLoad (see page 2108)	

## V.1 IRJointGussetBoltsDiag

### Class Hierarchy

#### C++

```
interface IRJointGussetBoltsDiag : IDispatch;
```

#### C#

```
public interface IRJointGussetBoltsDiag;
```

### Visual Basic

```
Public Interface IRJointGussetBoltsDiag
```

## V.1.1 IRJointGussetBoltsDiag Members

The following tables list the members exposed by IRJointGussetBoltsDiag.

### Public Fields

	Name	Description
◆	BoltAxisShift (see page 2050)	Available since version 1.7.
◆	ClassName (see page 2050)	Available since version 1.7.
◆	Diameter (see page 2051)	Available since version 1.7.
◆	DistanceH1 (see page 2051)	Available since version 1.7.
◆	Friction (see page 2051)	Available since version 1.7.
◆	Rows (see page 2052)	Available since version 1.7.
◆	Spacing (see page 2052)	Available since version 1.7.

## V.1.2 IRJointGussetBoltsDiag Fields

The fields of the IRJointGussetBoltsDiag class are listed here.

### Public Fields

	Name	Description
◆	BoltAxisShift (see page 2050)	Available since version 1.7.

◆	ClassName (see page 2050)	Available since version 1.7.
◆	Diameter (see page 2051)	Available since version 1.7.
◆	DistanceH1 (see page 2051)	Available since version 1.7.
◆	Friction (see page 2051)	Available since version 1.7.
◆	Rows (see page 2052)	Available since version 1.7.
◆	Spacing (see page 2052)	Available since version 1.7.

### V.1.2.1 BoltAxisShift

#### C++

```
HRESULT get_BoltAxisShift(double* );
HRESULT put_BoltAxisShift(double);
```

#### C#

```
public double BoltAxisShift { get; set; }
```

#### Visual Basic

```
Public BoltAxisShift As Double
```

#### Description

Available since version 1.7.

### V.1.2.2 ClassName

#### C++

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

#### C#

```
public String ClassName { get; set; }
```

#### Visual Basic

```
Public ClassName As String
```

#### Description

Available since version 1.7.

### V.1.2.3 Diameter

#### C++

```
HRESULT get_Diameter(double* );
HRESULT put_Diameter(double);
```

#### C#

```
public double Diameter { get; set; }
```

#### Visual Basic

```
Public Diameter As Double
```

#### Description

Available since version 1.7.

### V.1.2.4 DistanceH1

#### C++

```
HRESULT get_DistanceH1(double* );
HRESULT put_DistanceH1(double);
```

**C#**

```
public double DistanceH1 { get; set; }
```

**Visual Basic**

```
Public DistanceH1 As Double
```

**Description**

Available since version 1.7.

### V.1.2.5 Friction

**C++**

```
HRESULT get_Friction(double*);  
HRESULT put_Friction(double);
```

**C#**

```
public double Friction { get; set; }
```

**Visual Basic**

```
Public Friction As Double
```

**Description**

Available since version 1.7.

### V.1.2.6 Rows

**C++**

```
HRESULT get_Rows(long*);  
HRESULT put_Rows(long);
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Long
```

**Description**

Available since version 1.7.

### V.1.2.7 Spacing

**C++**

```
HRESULT get_Spacing(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray Spacing { get; }
```

**Visual Basic**

```
Public ReadOnly Spacing As IRobotValuesArray
```

**Description**

Available since version 1.7.

## V.2 IRJointGussetWeldsDiag

**Class Hierarchy**

**C++**

```
interface IRJointGussetWeldsDiag : IDispatch;
```

**C#**

```
public interface IRJointGussetWeldsDiag;
```

**Visual Basic**

```
Public Interface IRJointGussetWeldsDiag
```

**V.2.1 IRJointGussetWeldsDiag Members**

The following tables list the members exposed by IRJointGussetWeldsDiag.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Length1 (see page 2053)	Available since version ARSA 2010.
◆	Length2 (see page 2053)	Available since version ARSA 2010.
◆	ThickCorner (see page 2053)	Available since version 1.7.
◆	ThickFlange (see page 2054)	Available since version 1.7.

**V.2.2 IRJointGussetWeldsDiag Fields**

The fields of the IRJointGussetWeldsDiag class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Length1 (see page 2053)	Available since version ARSA 2010.
◆	Length2 (see page 2053)	Available since version ARSA 2010.
◆	ThickCorner (see page 2053)	Available since version 1.7.
◆	ThickFlange (see page 2054)	Available since version 1.7.

**V.2.2.1 Length1****C++**

```
HRESULT get_Length1( double* );
HRESULT put_Length1( double );
```

**C#**

```
public double Length1 { get; set; }
```

**Visual Basic**

```
Public Length1 As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.2.2.2 Length2****C++**

```
HRESULT get_Length2( double* );
HRESULT put_Length2( double );
```

**C#**

```
public double Length2 { get; set; }
```

**Visual Basic**

```
Public Length2 As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.2.2.3 ThickCorner

**C++**

```
HRESULT get_ThickCorner(double*);  
HRESULT put_ThickCorner(double);
```

**C#**

```
public double ThickCorner { get; set; }
```

**Visual Basic**

```
Public ThickCorner As Double
```

**Description**

Available since version 1.7.

### V.2.2.4 ThickFlange

**C++**

```
HRESULT get_ThickFlange(double*);  
HRESULT put_ThickFlange(double);
```

**C#**

```
public double ThickFlange { get; set; }
```

**Visual Basic**

```
Public ThickFlange As Double
```

**Description**

Available since version 1.7.

## V.3 IRJointGussetDiagonale

**Class Hierarchy****C++**

```
interface IRJointGussetDiagonale : IDispatch;
```

**C#**

```
public interface IRJointGussetDiagonale;
```

**Visual Basic**

```
Public Interface IRJointGussetDiagonale
```

### V.3.1 IRJointGussetDiagonale Members

The following tables list the members exposed by IRJointGussetDiagonale.

#### Public Fields

	Name	Description
◆	BarNumber ( [ see page 2055) )	
◆	BoltsDiag ( [ see page 2055) )	Available since version 1.7.
◆	DistanceEC ( [ see page 2055) )	Available since version 1.7.
◆	Exist ( [ see page 2056) )	
◆	Position ( [ see page 2056) )	Available since version 1.7.
◆	Profile ( [ see page 2056) )	Available since version 1.7.
◆	WeldsDiag ( [ see page 2056) )	Available since version 1.7.

### V.3.2 IRJointGussetDiagonale Fields

The fields of the IRJointGussetDiagonale class are listed here.

#### Public Fields

	Name	Description
◆	BarNumber ( [ see page 2055) )	
◆	BoltsDiag ( [ see page 2055) )	Available since version 1.7.
◆	DistanceEC ( [ see page 2055) )	Available since version 1.7.
◆	Exist ( [ see page 2056) )	
◆	Position ( [ see page 2056) )	Available since version 1.7.
◆	Profile ( [ see page 2056) )	Available since version 1.7.
◆	WeldsDiag ( [ see page 2056) )	Available since version 1.7.

#### V.3.2.1 BarNumber

##### C++

```
HRESULT get_BarNumber(long* );
HRESULT put_BarNumber(long);
```

##### C#

```
public long BarNumber { get; set; }
```

##### Visual Basic

```
Public BarNumber As long
```

##### Version

Available since version 4.5.

#### V.3.2.2 BoltsDiag

##### C++

```
HRESULT get_BoltsDiag(IRJointGussetBoltsDiag** );
```

##### C#

```
public IRJointGussetBoltsDiag BoltsDiag { get; }
```

##### Visual Basic

```
Public ReadOnly BoltsDiag As IRJointGussetBoltsDiag
```

**Description**

Available since version 1.7.

**V.3.2.3 DistanceEC****C++**

```
HRESULT get_DistanceEC(double*);  
HRESULT put_DistanceEC(double);
```

**C#**

```
public double DistanceEC { get; set; }
```

**Visual Basic**

```
Public DistanceEC As Double
```

**Description**

Available since version 1.7.

**V.3.2.4 Exist****C++**

```
HRESULT get_Exist(VARIANT_BOOL*);  
HRESULT put_Exist(VARIANT_BOOL);
```

**C#**

```
public bool Exist { get; set; }
```

**Visual Basic**

```
Public Exist As Boolean
```

**Version**

Available since version 4.5.

**V.3.2.5 Position****C++**

```
HRESULT get_Position(IRJointGussetDiagonalePositionType*);  
HRESULT put_Position(IRJointGussetDiagonalePositionType);
```

**C#**

```
public IRJointGussetDiagonalePositionType Position { get; set; }
```

**Visual Basic**

```
Public Position As IRJointGussetDiagonalePositionType
```

**Description**

Available since version 1.7.

**V.3.2.6 Profile****C++**

```
HRESULT get_Profile(IRJointProfile**);
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Description**

Available since version 1.7.

**V.3.2.7 WeldsDiag****C++**

```
HRESULT get_WeldsDiag(IRJointGussetWeldsDiag**);
```

**C#**

```
public IRJointGussetWeldsDiag WeldsDiag { get; }
```

**Visual Basic**

```
Public ReadOnly WeldsDiag As IRJointGussetWeldsDiag
```

**Description**

Available since version 1.7.

**V.4 IRJointGussetFixType****C++**

```
enum IRJointGussetFixType;
```

**C#**

```
public enum IRJointGussetFixType;
```

**Visual Basic**

```
Public Enum IRJointGussetFixType
```

**Members**

Members	Description
I_JGFT_BOLTED = 0	Available since version 1.7.
I_JGFT_WELDED = 1	Available since version 1.7.

**V.5 IRJointGussetSimpleProfilePosition****C++**

```
enum IRJointGussetSimpleProfilePosition;
```

**C#**

```
public enum IRJointGussetSimpleProfilePosition;
```

**Visual Basic**

```
Public Enum IRJointGussetSimpleProfilePosition
```

**Members**

Members	Description
I_JGSPP_VERTICAL = 0	Available since version 1.7.
I_JGSPP_DIAGONAL = 1	Available since version 1.7.

## V.6 IRJointGussetSimple

### Class Hierarchy

#### C++

```
interface IRJointGussetSimple : IRJointConnection;
```

#### C#

```
public interface IRJointGussetSimple : IRJointConnection;
```

### Visual Basic

```
Public Interface IRJointGussetSimple
```

## V.6.1 IRJointGussetSimple Members

The following tables list the members exposed by IRJointGussetSimple.

### Public Fields

	Name	Description
◆	Attachment ( <a href="#">see page 2058</a> )	Available since version 1.7.
◆	BoltsClassVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	BoltsDimensionNamesVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	BoltsDimensionsVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	Diagonale ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	FixType ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	GussetPlate ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	ProfilePosition ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	Type ( <a href="#">see page 2148</a> )	A connection type.
◆	WFRelPos ( <a href="#">see page 2148</a> )	A type of relative position of a beam and a column.

### Public Methods

	Name	Description
◆	GetFromRobot ( <a href="#">see page 2149</a> )	The function gets connection parameters from Robot and fills an object with the data.
◆	SetToRobot ( <a href="#">see page 2149</a> )	The function sets and saves connection parameter in Robot.

## V.6.2 IRJointGussetSimple Fields

The fields of the IRJointGussetSimple class are listed here.

### Public Fields

	Name	Description
◆	Attachment ( <a href="#">see page 2058</a> )	Available since version 1.7.
◆	BoltsClassVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	BoltsDimensionNamesVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	BoltsDimensionsVector ( <a href="#">see page 2059</a> )	Available since version ARSA 2010.
◆	Diagonale ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	FixType ( <a href="#">see page 2060</a> )	Available since version 1.7.
◆	GussetPlate ( <a href="#">see page 2060</a> )	Available since version 1.7.



ProfilePosition (see page 2060)

Available since version 1.7.

### V.6.2.1 Attachment

#### C++

```
HRESULT get_Attachment(IRJointGussetSimpleAttachment**);
```

#### C#

```
public IRJointGussetSimpleAttachment Attachment { get; }
```

#### Visual Basic

```
Public ReadOnly Attachment As IRJointGussetSimpleAttachment
```

#### Description

Available since version 1.7.

### V.6.2.2 BoltsClassVector

#### C++

```
HRESULT get_BoltsClassVector(IRobotNamesArray**);
```

#### C#

```
public IRobotNamesArray BoltsClassVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsClassVector As IRobotNamesArray
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.6.2.3 BoltsDimensionNamesVector

#### C++

```
HRESULT get_BoltsDimensionNamesVector(IRobotNamesArray**);
```

#### C#

```
public IRobotNamesArray BoltsDimensionNamesVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsDimensionNamesVector As IRobotNamesArray
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.6.2.4 BoltsDimensionsVector

#### C++

```
HRESULT get_BoltsDimensionsVector(IRobotValuesArray**);
```

#### C#

```
public IRobotValuesArray BoltsDimensionsVector { get; }
```

**Visual Basic**

```
Public ReadOnly BoltsDimensionsVector As IRobotValuesArray
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.6.2.5 Diagonale****C++**

```
HRESULT get_Diagonale(IRJointGussetDiagonale**);
```

**C#**

```
public IRJointGussetDiagonale Diagonale { get; }
```

**Visual Basic**

```
Public ReadOnly Diagonale As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

**V.6.2.6 FixType****C++**

```
HRESULT get_FixType(IRJointGussetFixType*);  
HRESULT put_FixType(IRJointGussetFixType*);
```

**C#**

```
public IRJointGussetFixType FixType { get; set; }
```

**Visual Basic**

```
Public FixType As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.6.2.7 GussetPlate****C++**

```
HRESULT get_GussetPlate(IRJointGussetSimplePlate**);
```

**C#**

```
public IRJointGussetSimplePlate GussetPlate { get; }
```

**Visual Basic**

```
Public ReadOnly GussetPlate As IRJointGussetSimplePlate
```

**Description**

Available since version 1.7.

**V.6.2.8 ProfilePosition****C++**

```
HRESULT get_ProfilePosition(IRJointGussetSimpleProfilePosition*);  
HRESULT put_ProfilePosition(IRJointGussetSimpleProfilePosition*);
```

**C#**

```
public IRJointGussetSimpleProfilePosition ProfilePosition { get; set; }
```

**Visual Basic**

```
Public ProfilePosition As IRJointGussetSimpleProfilePosition
```

**Description**

Available since version 1.7.

## V.7 IRJointGussetSimplePlate

**Class Hierarchy****C++**

```
interface IRJointGussetSimplePlate : IDispatch;
```

**C#**

```
public interface IRJointGussetSimplePlate;
```

**Visual Basic**

```
Public Interface IRJointGussetSimplePlate
```

### V.7.1 IRJointGussetSimplePlate Members

The following tables list the members exposed by IRJointGussetSimplePlate.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CornersType ( <a href="#">see page 2062</a> )	Available since version 1.7.
❖	DistanceD ( <a href="#">see page 2062</a> )	Available since version 1.7.
❖	DistanceEH ( <a href="#">see page 2062</a> )	Available since version 1.7.
❖	DistanceEV ( <a href="#">see page 2063</a> )	Available since version 1.7.
❖	H1 ( <a href="#">see page 2063</a> )	Available since version ARSA 2010.
❖	H2 ( <a href="#">see page 2063</a> )	Available since version ARSA 2010.
❖	H3 ( <a href="#">see page 2064</a> )	Available since version ARSA 2010.
❖	H4 ( <a href="#">see page 2064</a> )	Available since version ARSA 2010.
❖	Height ( <a href="#">see page 2064</a> )	Available since version 1.7.
❖	Material ( <a href="#">see page 2064</a> )	Available since version 1.7.
❖	Thick ( <a href="#">see page 2065</a> )	Available since version 1.7.
❖	V1 ( <a href="#">see page 2065</a> )	Available since version ARSA 2010.
❖	V2 ( <a href="#">see page 2065</a> )	Available since version ARSA 2010.
❖	V3 ( <a href="#">see page 2066</a> )	Available since version ARSA 2010.
❖	V4 ( <a href="#">see page 2066</a> )	Available since version ARSA 2010.
❖	Width ( <a href="#">see page 2066</a> )	Available since version 1.7.

### V.7.2 IRJointGussetSimplePlate Fields

The fields of the IRJointGussetSimplePlate class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	CornersType ( <a href="#">see page 2062</a> )	Available since version 1.7.
❖	DistanceD ( <a href="#">see page 2062</a> )	Available since version 1.7.

DistanceEH (see page 2062)	Available since version 1.7.
DistanceEV (see page 2063)	Available since version 1.7.
H1 (see page 2063)	Available since version ARSA 2010.
H2 (see page 2063)	Available since version ARSA 2010.
H3 (see page 2064)	Available since version ARSA 2010.
H4 (see page 2064)	Available since version ARSA 2010.
Height (see page 2064)	Available since version 1.7.
Material (see page 2064)	Available since version 1.7.
Thick (see page 2065)	Available since version 1.7.
V1 (see page 2065)	Available since version ARSA 2010.
V2 (see page 2065)	Available since version ARSA 2010.
V3 (see page 2066)	Available since version ARSA 2010.
V4 (see page 2066)	Available since version ARSA 2010.
Width (see page 2066)	Available since version 1.7.

### V.7.2.1 CornersType

#### C++

```
HRESULT get_CornersType(IRJointGussetCornersType* );
HRESULT put_CornersType(IRJointGussetCornersType);
```

#### C#

```
public IRJointGussetCornersType CornersType { get; set; }
```

#### Visual Basic

```
Public CornersType As IRJointGussetCornersType
```

#### Description

Available since version 1.7.

### V.7.2.2 DistanceD

#### C++

```
HRESULT get_DistanceD(double* );
HRESULT put_DistanceD(double);
```

#### C#

```
public double DistanceD { get; set; }
```

#### Visual Basic

```
Public DistanceD As Double
```

#### Description

Available since version 1.7.

### V.7.2.3 DistanceEH

#### C++

```
HRESULT get_DistanceEH(double* );
HRESULT put_DistanceEH(double);
```

#### C#

```
public double DistanceEH { get; set; }
```

**Visual Basic**

```
Public DistanceEH As double
```

**Description**

Available since version 1.7.

**V.7.2.4 DistanceEV****C++**

```
HRESULT get_DistanceEV(double*);  
HRESULT put_DistanceEV(double);
```

**C#**

```
public double DistanceEV { get; set; }
```

**Visual Basic**

```
Public DistanceEV As double
```

**Description**

Available since version 1.7.

**V.7.2.5 H1****C++**

```
HRESULT get_H1(double*);  
HRESULT put_H1(double);
```

**C#**

```
public double H1 { get; set; }
```

**Visual Basic**

```
Public H1 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.7.2.6 H2****C++**

```
HRESULT get_H2(double*);  
HRESULT put_H2(double);
```

**C#**

```
public double H2 { get; set; }
```

**Visual Basic**

```
Public H2 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.7.2.7 H3

#### C++

```
HRESULT get_H3(double* );
HRESULT put_H3(double);
```

#### C#

```
public double H3 { get; set; }
```

#### Visual Basic

```
Public H3 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.7.2.8 H4

#### C++

```
HRESULT get_H4(double* );
HRESULT put_H4(double);
```

#### C#

```
public double H4 { get; set; }
```

#### Visual Basic

```
Public H4 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.7.2.9 Height

#### C++

```
HRESULT get_Height(double* );
HRESULT put_Height(double);
```

#### C#

```
public double Height { get; set; }
```

#### Visual Basic

```
Public Height As double
```

#### Description

Available since version 1.7.

### V.7.2.10 Material

#### C++

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Available since version 1.7.

**V.7.2.11 Thick****C++**

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As Double
```

**Description**

Available since version 1.7.

**V.7.2.12 V1****C++**

```
HRESULT get_V1(double*);  
HRESULT put_V1(double);
```

**C#**

```
public double v1 { get; set; }
```

**Visual Basic**

```
Public v1 As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.7.2.13 V2****C++**

```
HRESULT get_V2(double*);  
HRESULT put_V2(double);
```

**C#**

```
public double v2 { get; set; }
```

**Visual Basic**

```
Public v2 As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.7.2.14 V3****C++**

```
HRESULT get_V3(double*);  
HRESULT put_V3(double);
```

**C#**

```
public double V3 { get; set; }
```

**Visual Basic**

```
Public V3 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.7.2.15 V4****C++**

```
HRESULT get_V4(double*);  
HRESULT put_V4(double);
```

**C#**

```
public double V4 { get; set; }
```

**Visual Basic**

```
Public V4 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.7.2.16 Width****C++**

```
HRESULT get_Width(double*);  
HRESULT put_Width(double);
```

**C#**

```
public double Width { get; set; }
```

**Visual Basic**

```
Public Width As double
```

**Description**

Available since version 1.7.

## V.8 IRJointGussetCornersType

### C++

```
enum IRJointGussetCornersType;
```

### C#

```
public enum IRJointGussetCornersType;
```

### Visual Basic

```
Public Enum IRJointGussetCornersType
```

### Members

Members	Description
I_JGCT_ACUTE = 0	Available since version 1.7.
I_JGCT_CUT_IN = 1	Available since version 1.7.
I_JGCT_CUT_OUT = 2	Available since version 1.7.

## V.9 IRJointGussetSimpleAttachment

### Class Hierarchy

### C++

```
interface IRJointGussetSimpleAttachment : IDispatch;
```

### C#

```
public interface IRJointGussetSimpleAttachment;
```

### Visual Basic

```
Public Interface IRJointGussetSimpleAttachment
```

### V.9.1 IRJointGussetSimpleAttachment Members

The following tables list the members exposed by IRJointGussetSimpleAttachment.

### Public Fields

	Name	Description
❖	BoltsAttach (see page 2068)	Available since version 1.7.
❖	FixType (see page 2068)	Available since version 1.7.
❖	WeldsAttach (see page 2068)	Available since version 1.7.
❖	WeldType (see page 2068)	Available since version ARSA 2010.

### V.9.2 IRJointGussetSimpleAttachment Fields

The fields of the IRJointGussetSimpleAttachment class are listed here.

### Public Fields

	Name	Description
❖	BoltsAttach (see page 2068)	Available since version 1.7.
❖	FixType (see page 2068)	Available since version 1.7.
❖	WeldsAttach (see page 2068)	Available since version 1.7.
❖	WeldType (see page 2068)	Available since version ARSA 2010.

### V.9.2.1 BoltsAttach

#### C++

```
HRESULT get_BoltsAttach(IRJointGussetSimpleAttachBolts**);
```

#### C#

```
public IRJointGussetSimpleAttachBolts BoltsAttach { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsAttach As IRJointGussetSimpleAttachBolts
```

#### Description

Available since version 1.7.

### V.9.2.2 FixType

#### C++

```
HRESULT get_FixType(IRJointGussetFixType*);  
HRESULT put_FixType(IRJointGussetFixType);
```

#### C#

```
public IRJointGussetFixType FixType { get; set; }
```

#### Visual Basic

```
Public FixType As IRJointGussetFixType
```

#### Description

Available since version 1.7.

### V.9.2.3 WeldsAttach

#### C++

```
HRESULT get_WeldsAttach(IRJointGussetSimpleAttachWelds**);
```

#### C#

```
public IRJointGussetSimpleAttachWelds WeldsAttach { get; }
```

#### Visual Basic

```
Public ReadOnly WeldsAttach As IRJointGussetSimpleAttachWelds
```

#### Description

Available since version 1.7.

### V.9.2.4 WeldType

#### C++

```
HRESULT get_WeldType(RJointWeldType*);  
HRESULT put_WeldType(RJointWeldType);
```

#### C#

```
public RJointWeldType WeldType { get; set; }
```

#### Visual Basic

```
Public WeldType As RJointWeldType
```

#### Description

Available since version ARSA 2010.

## Version

Available since version 10.

## V.10 IRJointGussetSimpleAttachBolts

### Class Hierarchy

#### C++

```
interface IRJointGussetSimpleAttachBolts : IDispatch;
```

#### C#

```
public interface IRJointGussetSimpleAttachBolts;
```

### Visual Basic

```
Public Interface IRJointGussetSimpleAttachBolts
```

## V.10.1 IRJointGussetSimpleAttachBolts Members

The following tables list the members exposed by IRJointGussetSimpleAttachBolts.

### Public Fields

	Name	Description
◆	ClassName ( [ see page 2069) )	Available since version 1.7.
◆	Diameter ( [ see page 2070) )	Available since version 1.7.
◆	Friction ( [ see page 2070) )	Available since version 1.7.
◆	Horizontal ( [ see page 2070) )	Available since version 1.7.
◆	Vertical ( [ see page 2070) )	Available since version 1.7.

## V.10.2 IRJointGussetSimpleAttachBolts Fields

The fields of the IRJointGussetSimpleAttachBolts class are listed here.

### Public Fields

	Name	Description
◆	ClassName ( [ see page 2069) )	Available since version 1.7.
◆	Diameter ( [ see page 2070) )	Available since version 1.7.
◆	Friction ( [ see page 2070) )	Available since version 1.7.
◆	Horizontal ( [ see page 2070) )	Available since version 1.7.
◆	Vertical ( [ see page 2070) )	Available since version 1.7.

### V.10.2.1 ClassName

#### C++

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

#### C#

```
public String ClassName { get; set; }
```

### Visual Basic

```
Public ClassName As String
```

### Description

Available since version 1.7.

### V.10.2.2 Diameter

#### C++

```
HRESULT get_Diameter(double*);  
HRESULT put_Diameter(double);
```

#### C#

```
public double Diameter { get; set; }
```

#### Visual Basic

```
Public Diameter As double
```

#### Description

Available since version 1.7.

### V.10.2.3 Friction

#### C++

```
HRESULT get_Friction(double*);  
HRESULT put_Friction(double);
```

#### C#

```
public double Friction { get; set; }
```

#### Visual Basic

```
Public Friction As double
```

#### Description

Available since version 1.7.

### V.10.2.4 Horizontal

#### C++

```
HRESULT get_Horizontal(IRJointGussetSimpleAttachBoltsHorizontal**);
```

#### C#

```
public IRJointGussetSimpleAttachBoltsHorizontal Horizontal { get; }
```

#### Visual Basic

```
Public ReadOnly Horizontal As IRJointGussetSimpleAttachBoltsHorizontal
```

#### Description

Available since version 1.7.

### V.10.2.5 Vertical

#### C++

```
HRESULT get_Vertical(IRJointGussetSimpleAttachBoltsVertical**);
```

#### C#

```
public IRJointGussetSimpleAttachBoltsVertical Vertical { get; }
```

#### Visual Basic

```
Public ReadOnly Vertical As IRJointGussetSimpleAttachBoltsVertical
```

#### Description

Available since version 1.7.

## V.11 IRJointGussetSimpleAttachBoltsHorizontal

### Class Hierarchy

### C++

```
interface IRJointGussetSimpleAttachBoltsHorizontal : IDispatch;
```

### C#

```
public interface IRJointGussetSimpleAttachBoltsHorizontal;
```

### Visual Basic

```
Public Interface IRJointGussetSimpleAttachBoltsHorizontal
```

### V.11.1 IRJointGussetSimpleAttachBoltsHorizontal Members

The following tables list the members exposed by IRJointGussetSimpleAttachBoltsHorizontal.

#### Public Fields

	Name	Description
◆	DistanceEB (see page 2071)	Available since version 1.7.
◆	DistanceH1 (see page 2072)	Available since version 1.7.
◆	Rows (see page 2072)	Available since version 1.7.
◆	Spacing (see page 2072)	Available since version 1.7.

### V.11.2 IRJointGussetSimpleAttachBoltsHorizontal Fields

The fields of the IRJointGussetSimpleAttachBoltsHorizontal class are listed here.

#### Public Fields

	Name	Description
◆	DistanceEB (see page 2071)	Available since version 1.7.
◆	DistanceH1 (see page 2072)	Available since version 1.7.
◆	Rows (see page 2072)	Available since version 1.7.
◆	Spacing (see page 2072)	Available since version 1.7.

#### V.11.2.1 DistanceEB

### C++

```
HRESULT get_DistanceEB(double* );
HRESULT put_DistanceEB(double);
```

### C#

```
public double DistanceEB { get; set; }
```

### Visual Basic

```
Public DistanceEB As Double
```

### Description

Available since version 1.7.

#### V.11.2.2 DistanceH1

### C++

```
HRESULT get_DistanceH1(double* );
HRESULT put_DistanceH1(double);
```

**C#**

```
public double DistanceH1 { get; set; }
```

**Visual Basic**

```
Public DistanceH1 As Double
```

**Description**

Available since version 1.7.

### V.11.2.3 Rows

**C++**

```
HRESULT get_Rows(long*);  
HRESULT put_Rows(long);
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Long
```

**Description**

Available since version 1.7.

### V.11.2.4 Spacing

**C++**

```
HRESULT get_Spacing(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray Spacing { get; }
```

**Visual Basic**

```
Public ReadOnly Spacing As IRobotValuesArray
```

**Description**

Available since version 1.7.

## V.12 IRJointGussetSimpleAttachBoltsVertical

**Class Hierarchy****C++**

```
interface IRJointGussetSimpleAttachBoltsVertical : IDispatch;
```

**C#**

```
public interface IRJointGussetSimpleAttachBoltsVertical;
```

**Visual Basic**

```
Public Interface IRJointGussetSimpleAttachBoltsVertical
```

### V.12.1 IRJointGussetSimpleAttachBoltsVertical Members

The following tables list the members exposed by IRJointGussetSimpleAttachBoltsVertical.

## Public Fields

	Name	Description
◆	DistanceEB (see page 2073)	Available since version 1.7.
◆	DistanceH1 (see page 2073)	Available since version 1.7.
◆	Rows (see page 2074)	Available since version 1.7.
◆	Spacing (see page 2074)	Available since version 1.7.

## V.12.2 IRJointGussetSimpleAttachBoltsVertical Fields

The fields of the IRJointGussetSimpleAttachBoltsVertical class are listed here.

## Public Fields

	Name	Description
◆	DistanceEB (see page 2073)	Available since version 1.7.
◆	DistanceH1 (see page 2073)	Available since version 1.7.
◆	Rows (see page 2074)	Available since version 1.7.
◆	Spacing (see page 2074)	Available since version 1.7.

### V.12.2.1 DistanceEB

#### C++

```
HRESULT get_DistanceEB(double* );
HRESULT put_DistanceEB(double);
```

#### C#

```
public double DistanceEB { get; set; }
```

#### Visual Basic

```
Public DistanceEB As double
```

#### Description

Available since version 1.7.

### V.12.2.2 DistanceH1

#### C++

```
HRESULT get_DistanceH1(double* );
HRESULT put_DistanceH1(double);
```

#### C#

```
public double DistanceH1 { get; set; }
```

#### Visual Basic

```
Public DistanceH1 As double
```

#### Description

Available since version 1.7.

### V.12.2.3 Rows

#### C++

```
HRESULT get_Rows(long* );
HRESULT put_Rows(long);
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As Long
```

**Description**

Available since version 1.7.

**V.12.2.4 Spacing****C++**

```
HRESULT get_Spacing(IRobotValuesArray**);
```

**C#**

```
public IRobotValuesArray Spacing { get; }
```

**Visual Basic**

```
Public ReadOnly Spacing As IRobotValuesArray
```

**Description**

Available since version 1.7.

**V.13 IRJointGussetSimpleAttachWelds****Class Hierarchy****C++**

```
interface IRJointGussetSimpleAttachWelds : IDispatch;
```

**C#**

```
public interface IRJointGussetSimpleAttachWelds;
```

**Visual Basic**

```
Public Interface IRJointGussetSimpleAttachWelds
```

**V.13.1 IRJointGussetSimpleAttachWelds Members**

The following tables list the members exposed by IRJointGussetSimpleAttachWelds.

**Public Fields**

	Name	Description
◆	ThickEdgeA (see page 2075)	Available since version 1.7.
◆	ThickEdgeB (see page 2075)	Available since version 1.7.

**V.13.2 IRJointGussetSimpleAttachWelds Fields**

The fields of the IRJointGussetSimpleAttachWelds class are listed here.

**Public Fields**

	Name	Description
◆	ThickEdgeA (see page 2075)	Available since version 1.7.
◆	ThickEdgeB (see page 2075)	Available since version 1.7.

### V.13.2.1 ThickEdgeA

#### C++

```
HRESULT get_ThickEdgeA(double* );
HRESULT put_ThickEdgeA(double);
```

#### C#

```
public double ThickEdgeA { get; set; }
```

#### Visual Basic

```
Public ThickEdgeA As Double
```

#### Description

Available since version 1.7.

### V.13.2.2 ThickEdgeB

#### C++

```
HRESULT get_ThickEdgeB(double* );
HRESULT put_ThickEdgeB(double);
```

#### C#

```
public double ThickEdgeB { get; set; }
```

#### Visual Basic

```
Public ThickEdgeB As Double
```

#### Description

Available since version 1.7.

## V.14 IRJointGussetCrossProfileCutting

#### C++

```
enum IRJointGussetCrossProfileCutting;
```

#### C#

```
public enum IRJointGussetCrossProfileCutting;
```

#### Visual Basic

```
Public Enum IRJointGussetCrossProfileCutting
```

#### Members

Members	Description
I_JGPC_CONTINUE_LEFTUPPER_RIGHTLOWER = 0	Available since version 1.7.
I_JGPC_CONTINUE_LEFTLOWER_RIGHTUPPER = 1	Available since version 1.7.
I_JGPC_ALL_CUTTED = 2	Available since version 1.7.

## V.15 IRJointGussetCrossPlate

#### Class Hierarchy

#### C++

```
interface IRJointGussetCrossPlate : IDispatch;
```

**C#**

```
public interface IRJointGussetCrossPlate;
```

**Visual Basic**

```
Public Interface IRJointGussetCrossPlate
```

**V.15.1 IRJointGussetCrossPlate Members**

The following tables list the members exposed by IRJointGussetCrossPlate.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	B1 ( <a href="#">see page 2077</a> )	Available since version 1.7.
◆	B2 ( <a href="#">see page 2077</a> )	Available since version 1.7.
◆	B3 ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	B4 ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	CornersType ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	CutH1 ( <a href="#">see page 2078</a> )	Available since version ARSA 2010.
◆	CutH2 ( <a href="#">see page 2079</a> )	Available since version ARSA 2010.
◆	CutH3 ( <a href="#">see page 2079</a> )	Available since version ARSA 2010.
◆	CutH4 ( <a href="#">see page 2079</a> )	Available since version ARSA 2010.
◆	CutV1 ( <a href="#">see page 2080</a> )	Available since version ARSA 2010.
◆	CutV2 ( <a href="#">see page 2080</a> )	Available since version ARSA 2010.
◆	CutV3 ( <a href="#">see page 2080</a> )	Available since version ARSA 2010.
◆	CutV4 ( <a href="#">see page 2081</a> )	Available since version ARSA 2010.
◆	H1 ( <a href="#">see page 2081</a> )	Available since version 1.7.
◆	H2 ( <a href="#">see page 2081</a> )	Available since version 1.7.
◆	H3 ( <a href="#">see page 2082</a> )	Available since version 1.7.
◆	H4 ( <a href="#">see page 2082</a> )	Available since version 1.7.
◆	Height ( <a href="#">see page 2082</a> )	Available since version ARSA 2010.
◆	HOffset ( <a href="#">see page 2082</a> )	Available since version ARSA 2010.
◆	HorizontalOffset ( <a href="#">see page 2083</a> )	Available since version ARSA 2010.
◆	Length ( <a href="#">see page 2083</a> )	Available since version ARSA 2010.
◆	Material ( <a href="#">see page 2083</a> )	Available since version 1.7.
◆	Thick ( <a href="#">see page 2084</a> )	Available since version 1.7.
◆	VOffset ( <a href="#">see page 2084</a> )	Available since version ARSA 2010.

**V.15.2 IRJointGussetCrossPlate Fields**

The fields of the IRJointGussetCrossPlate class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	B1 ( <a href="#">see page 2077</a> )	Available since version 1.7.
◆	B2 ( <a href="#">see page 2077</a> )	Available since version 1.7.
◆	B3 ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	B4 ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	CornersType ( <a href="#">see page 2078</a> )	Available since version 1.7.
◆	CutH1 ( <a href="#">see page 2078</a> )	Available since version ARSA 2010.
◆	CutH2 ( <a href="#">see page 2079</a> )	Available since version ARSA 2010.
◆	CutH3 ( <a href="#">see page 2079</a> )	Available since version ARSA 2010.

◆	CutH4 ( [ see page 2079 )	Available since version ARSA 2010.
◆	CutV1 ( [ see page 2080 )	Available since version ARSA 2010.
◆	CutV2 ( [ see page 2080 )	Available since version ARSA 2010.
◆	CutV3 ( [ see page 2080 )	Available since version ARSA 2010.
◆	CutV4 ( [ see page 2081 )	Available since version ARSA 2010.
◆	H1 ( [ see page 2081 )	Available since version 1.7.
◆	H2 ( [ see page 2081 )	Available since version 1.7.
◆	H3 ( [ see page 2082 )	Available since version 1.7.
◆	H4 ( [ see page 2082 )	Available since version 1.7.
◆	Height ( [ see page 2082 )	Available since version ARSA 2010.
◆	HOffset ( [ see page 2082 )	Available since version ARSA 2010.
◆	HorizontalOffset ( [ see page 2083 )	Available since version ARSA 2010.
◆	Length ( [ see page 2083 )	Available since version ARSA 2010.
◆	Material ( [ see page 2083 )	Available since version 1.7.
◆	Thick ( [ see page 2084 )	Available since version 1.7.
◆	VOffset ( [ see page 2084 )	Available since version ARSA 2010.

### V.15.2.1 B1

**C++**

```
HRESULT get_B1(double* );
HRESULT put_B1(double);
```

**C#**

```
public double B1 { get; set; }
```

**Visual Basic**

```
Public B1 As Double
```

**Description**

Available since version 1.7.

### V.15.2.2 B2

**C++**

```
HRESULT get_B2(double* );
HRESULT put_B2(double);
```

**C#**

```
public double B2 { get; set; }
```

**Visual Basic**

```
Public B2 As Double
```

**Description**

Available since version 1.7.

### V.15.2.3 B3

**C++**

```
HRESULT get_B3(double* );
HRESULT put_B3(double);
```

**C#**

```
public double B3 { get; set; }
```

**Visual Basic**

```
Public B3 As double
```

**Description**

Available since version 1.7.

**V.15.2.4 B4****C++**

```
HRESULT get_B4(double* );
HRESULT put_B4(double);
```

**C#**

```
public double B4 { get; set; }
```

**Visual Basic**

```
Public B4 As double
```

**Description**

Available since version 1.7.

**V.15.2.5 CornersType****C++**

```
HRESULT get_CornersType(IRJointGussetCornersType* );
HRESULT put_CornersType(IRJointGussetCornersType);
```

**C#**

```
public IRJointGussetCornersType CornersType { get; set; }
```

**Visual Basic**

```
Public CornersType As IRJointGussetCornersType
```

**Description**

Available since version 1.7.

**V.15.2.6 CutH1****C++**

```
HRESULT get_CutH1(double* );
HRESULT put_CutH1(double);
```

**C#**

```
public double CutH1 { get; set; }
```

**Visual Basic**

```
Public CutH1 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.15.2.7 CutH2

#### C++

```
HRESULT get_CutH2(double*);  
HRESULT put_CutH2(double);
```

#### C#

```
public double CutH2 { get; set; }
```

#### Visual Basic

```
Public CutH2 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.8 CutH3

#### C++

```
HRESULT get_CutH3(double*);  
HRESULT put_CutH3(double);
```

#### C#

```
public double CutH3 { get; set; }
```

#### Visual Basic

```
Public CutH3 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.9 CutH4

#### C++

```
HRESULT get_CutH4(double*);  
HRESULT put_CutH4(double);
```

#### C#

```
public double CutH4 { get; set; }
```

#### Visual Basic

```
Public CutH4 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.10 CutV1

#### C++

```
HRESULT get_CutV1(double*);  
HRESULT put_CutV1(double);
```

#### C#

```
public double CutV1 { get; set; }
```

#### Visual Basic

```
Public CutV1 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.11 CutV2

#### C++

```
HRESULT get_CutV2(double*);  
HRESULT put_CutV2(double);
```

#### C#

```
public double CutV2 { get; set; }
```

#### Visual Basic

```
Public CutV2 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.12 CutV3

#### C++

```
HRESULT get_CutV3(double*);  
HRESULT put_CutV3(double);
```

#### C#

```
public double CutV3 { get; set; }
```

#### Visual Basic

```
Public CutV3 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.13 CutV4

#### C++

```
HRESULT get_CutV4(double*);  
HRESULT put_CutV4(double);
```

#### C#

```
public double CutV4 { get; set; }
```

#### Visual Basic

```
Public CutV4 As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.14 H1

#### C++

```
HRESULT get_H1(double*);  
HRESULT put_H1(double);
```

#### C#

```
public double H1 { get; set; }
```

#### Visual Basic

```
Public H1 As double
```

#### Description

Available since version 1.7.

### V.15.2.15 H2

#### C++

```
HRESULT get_H2(double*);  
HRESULT put_H2(double);
```

#### C#

```
public double H2 { get; set; }
```

#### Visual Basic

```
Public H2 As double
```

#### Description

Available since version 1.7.

### V.15.2.16 H3

#### C++

```
HRESULT get_H3(double*);  
HRESULT put_H3(double);
```

#### C#

```
public double H3 { get; set; }
```

**Visual Basic**

```
Public H3 As double
```

**Description**

Available since version 1.7.

**V.15.2.17 H4****C++**

```
HRESULT get_H4(double*);  
HRESULT put_H4(double);
```

**C#**

```
public double H4 { get; set; }
```

**Visual Basic**

```
Public H4 As double
```

**Description**

Available since version 1.7.

**V.15.2.18 Height****C++**

```
HRESULT get_Height(double*);  
HRESULT put_Height(double);
```

**C#**

```
public double Height { get; set; }
```

**Visual Basic**

```
Public Height As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.15.2.19 HOffset****C++**

```
HRESULT get_HOffset(double*);  
HRESULT put_HOffset(double);
```

**C#**

```
public double HOffset { get; set; }
```

**Visual Basic**

```
Public HOffset As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.15.2.20 HorizontalOffset

#### C++

```
HRESULT get_HorizontalOffset(double*);  
HRESULT put_HorizontalOffset(double);
```

#### C#

```
public double HorizontalOffset { get; set; }
```

#### Visual Basic

```
Public HorizontalOffset As Double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.21 Length

#### C++

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

#### C#

```
public double Length { get; set; }
```

#### Visual Basic

```
Public Length As Double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.15.2.22 Material

#### C++

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

#### C#

```
public String Material { get; set; }
```

#### Visual Basic

```
Public Material As String
```

#### Description

Available since version 1.7.

### V.15.2.23 Thick

#### C++

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As Double
```

**Description**

Available since version 1.7.

**V.15.2.24 VOffset****C++**

```
HRESULT get_VOffset(double* );
HRESULT put_VOffset(double);
```

**C#**

```
public double VOffset { get; set; }
```

**Visual Basic**

```
Public VOffset As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.16 IRJointGussetCross****Class Hierarchy****C++**

```
interface IRJointGussetCross : IRJointConnection;
```

**C#**

```
public interface IRJointGussetCross : IRJointConnection;
```

**Visual Basic**

```
Public Interface IRJointGussetCross
```

**V.16.1 IRJointGussetCross Members**

The following tables list the members exposed by IRJointGussetCross.

**Public Fields**

	Name	Description
◆	BoltsClassVector ( [ see page 2086] )	Available since ARSA 2010.
◆	BoltsDimensionNamesVector ( [ see page 2086] )	Available since ARSA 2010.
◆	BoltsDimensionsVector ( [ see page 2087] )	Available since ARSA 2010.
◆	DiagLeftLower ( [ see page 2087] )	Available since version 1.7.
◆	DiagLeftUpper ( [ see page 2087] )	Available since version 1.7.
◆	DiagRightLower ( [ see page 2087] )	Available since version 1.7.
◆	DiagRightUpper ( [ see page 2088] )	Available since version 1.7.

◆	FixType_LeftL_RightU (see page 2088)	Available since version 1.7.
◆	FixType_LeftLower (see page 2088)	Available since ARSA 2010.
◆	FixType_LeftU_RightL (see page 2088)	Available since version 1.7.
◆	FixType_LeftUpper (see page 2089)	Available since ARSA 2010.
◆	FixType_RightLower (see page 2089)	Available since version ARSA 2010.
◆	FixType_RightUpper (see page 2089)	Available since ARSA 2010.
◆	GussetPlate (see page 2090)	Available since version 1.7.
◆	ProfileCutting (see page 2090)	Available since version 1.7.
◆	Type (see page 2148)	A connection type.
◆	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

#### Public Methods

	Name	Description
◆	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
◆	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

### V.16.2 IRJointGussetCross Fields

The fields of the IRJointGussetCross class are listed here.

#### Public Fields

	Name	Description
◆	BoltsClassVector (see page 2086)	Available since ARSA 2010.
◆	BoltsDimensionNamesVector (see page 2086)	Available since ARSA 2010.
◆	BoltsDimensionsVector (see page 2087)	Available since ARSA 2010.
◆	DiagLeftLower (see page 2087)	Available since version 1.7.
◆	DiagLeftUpper (see page 2087)	Available since version 1.7.
◆	DiagRightLower (see page 2087)	Available since version 1.7.
◆	DiagRightUpper (see page 2088)	Available since version 1.7.
◆	FixType_LeftL_RightU (see page 2088)	Available since version 1.7.
◆	FixType_LeftLower (see page 2088)	Available since ARSA 2010.
◆	FixType_LeftU_RightL (see page 2088)	Available since version 1.7.
◆	FixType_LeftUpper (see page 2089)	Available since ARSA 2010.
◆	FixType_RightLower (see page 2089)	Available since version ARSA 2010.
◆	FixType_RightUpper (see page 2089)	Available since ARSA 2010.
◆	GussetPlate (see page 2090)	Available since version 1.7.
◆	ProfileCutting (see page 2090)	Available since version 1.7.

### V.16.2.1 BoltsClassVector

#### C++

```
HRESULT get_BoltsClassVector(IRobotNamesArray**);
```

#### C#

```
public IRobotNamesArray BoltsClassVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsClassVector As IRobotNamesArray
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.16.2.2 BoltsDimensionNamesVector

#### C++

```
HRESULT get_BoltsDimensionNamesVector(IRobotNamesArray**);
```

#### C#

```
public IRobotNamesArray BoltsDimensionNamesVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsDimensionNamesVector As IRobotNamesArray
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.16.2.3 BoltsDimensionsVector

#### C++

```
HRESULT get_BoltsDimensionsVector(IRobotValuesArray**);
```

#### C#

```
public IRobotValuesArray BoltsDimensionsVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsDimensionsVector As IRobotValuesArray
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.16.2.4 DiagLeftLower

#### C++

```
HRESULT get_DiagLeftLower(IRJointGussetDiagonale**);
```

**C#**

```
public IRJointGussetDiagonale DiagLeftLower { get; }
```

**Visual Basic**

```
Public ReadOnly DiagLeftLower As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

### V.16.2.5 DiagLeftUpper

**C++**

```
HRESULT get_DiagLeftUpper(IRJointGussetDiagonale**);
```

**C#**

```
public IRJointGussetDiagonale DiagLeftUpper { get; }
```

**Visual Basic**

```
Public ReadOnly DiagLeftUpper As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

### V.16.2.6 DiagRightLower

**C++**

```
HRESULT get_DiagRightLower(IRJointGussetDiagonale**);
```

**C#**

```
public IRJointGussetDiagonale DiagRightLower { get; }
```

**Visual Basic**

```
Public ReadOnly DiagRightLower As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

### V.16.2.7 DiagRightUpper

**C++**

```
HRESULT get_DiagRightUpper(IRJointGussetDiagonale**);
```

**C#**

```
public IRJointGussetDiagonale DiagRightUpper { get; }
```

**Visual Basic**

```
Public ReadOnly DiagRightUpper As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

### V.16.2.8 FixType\_LeftL\_RightU

**C++**

```
HRESULT get_FixType_LeftL_RightU(IRJointGussetFixType*);  
HRESULT put_FixType_LeftL_RightU(IRJointGussetFixType*);
```

**C#**

```
public IRJointGussetFixType FixType_LeftL_RightU { get; set; }
```

**Visual Basic**

```
Public FixType_LeftL_RightU As IRJointGussetFixType
```

**Description**

Available since version 1.7.

### V.16.2.9 FixType\_LeftLower

**C++**

```
HRESULT get_FixType_LeftLower(IRJointGussetFixType*);  
HRESULT put_FixType_LeftLower(IRJointGussetFixType*);
```

**C#**

```
public IRJointGussetFixType FixType_LeftLower { get; set; }
```

**Visual Basic**

```
Public FixType_LeftLower As IRJointGussetFixType
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

### V.16.2.10 FixType\_LeftU\_RightL

**C++**

```
HRESULT get_FixType_LeftU_RightL(IRJointGussetFixType*);  
HRESULT put_FixType_LeftU_RightL(IRJointGussetFixType*);
```

**C#**

```
public IRJointGussetFixType FixType_LeftU_RightL { get; set; }
```

**Visual Basic**

```
Public FixType_LeftU_RightL As IRJointGussetFixType
```

**Description**

Available since version 1.7.

### V.16.2.11 FixType\_LeftUpper

**C++**

```
HRESULT get_FixType_LeftUpper(IRJointGussetFixType*);  
HRESULT put_FixType_LeftUpper(IRJointGussetFixType*);
```

**C#**

```
public IRJointGussetFixType FixType_LeftUpper { get; set; }
```

**Visual Basic**

```
Public FixType_LeftUpper As IRJointGussetFixType
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

**V.16.2.12 FixType\_RightLower****C++**

```
HRESULT get_FixType_RightLower(IRJointGussetFixType* );
HRESULT put_FixType_RightLower(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_RightLower { get; set; }
```

**Visual Basic**

```
Public FixType_RightLower As IRJointGussetFixType
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.16.2.13 FixType\_RightUpper****C++**

```
HRESULT get_FixType_RightUpper(IRJointGussetFixType* );
HRESULT put_FixType_RightUpper(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_RightUpper { get; set; }
```

**Visual Basic**

```
Public FixType_RightUpper As IRJointGussetFixType
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

**V.16.2.14 GussetPlate****C++**

```
HRESULT get_GussetPlate(IRJointGussetCrossPlate** );
```

**C#**

```
public IRJointGussetCrossPlate GussetPlate { get; }
```

**Visual Basic**

```
Public ReadOnly GussetPlate As IRJointGussetCrossPlate
```

**Description**

Available since version 1.7.

**V.16.2.15 ProfileCutting****C++**

```
HRESULT get_ProfileCutting(IRJointGussetCrossProfileCutting* );
```

```
HRESULT put_ProfileCutting(IRJointGussetCrossProfileCutting);
```

**C#**

```
public IRJointGussetCrossProfileCutting ProfileCutting { get; set; }
```

**Visual Basic**

```
Public ProfileCutting As IRJointGussetCrossProfileCutting
```

**Description**

Available since version 1.7.

## V.17 IRJointGussetFlangeProfileCutting

**C++**

```
enum IRJointGussetFlangeProfileCutting;
```

**C#**

```
public enum IRJointGussetFlangeProfileCutting;
```

**Visual Basic**

```
Public Enum IRJointGussetFlangeProfileCutting
```

**Members**

Members	Description
I_JGFC_CONTINUOUS = 0	Available since version 1.7.
I_JGFC_CUTTED = 1	Available since version 1.7.

## V.18 IRJointGussetFlangePlate

**Class Hierarchy****C++**

```
interface IRJointGussetFlangePlate : IDispatch;
```

**C#**

```
public interface IRJointGussetFlangePlate;
```

**Visual Basic**

```
Public Interface IRJointGussetFlangePlate
```

### V.18.1 IRJointGussetFlangePlate Members

The following tables list the members exposed by IRJointGussetFlangePlate.

**Public Fields**

	Name	Description
◆	B1 (see page 2092)	Available since version 1.7.
◆	B2 (see page 2092)	Available since version 1.7.
◆	B3 (see page 2093)	Available since version 1.7.
◆	B4 (see page 2093)	Available since version 1.7.
◆	CornersType (see page 2093)	Available since version 1.7.
◆	CutH1 (see page 2093)	Available since ARSA 2010.
◆	CutH2 (see page 2094)	Available since ARSA 2010.
◆	CutH3 (see page 2094)	Available since ARSA 2010.

◆	CutH4 ( <a href="#">see page 2094</a> )	Available since ARSA 2010.
◆	CutV1 ( <a href="#">see page 2095</a> )	Available since version ARSA 2010.
◆	CutV2 ( <a href="#">see page 2095</a> )	Available since ARSA 2010.
◆	CutV3 ( <a href="#">see page 2095</a> )	Available since ARSA 2010.
◆	CutV4 ( <a href="#">see page 2096</a> )	Available since ARSA 2010.
◆	EH ( <a href="#">see page 2096</a> )	Available since version 1.7.
◆	EV ( <a href="#">see page 2096</a> )	Available since version 1.7.
◆	Gheight ( <a href="#">see page 2097</a> )	Available since version ARSA 2010.
◆	H1 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H2 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H3 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H4 ( <a href="#">see page 2098</a> )	Available since version 1.7.
◆	HOffset ( <a href="#">see page 2098</a> )	Available since version ARSA 2010.
◆	Length ( <a href="#">see page 2098</a> )	Available since version ARSA 2010.
◆	Material ( <a href="#">see page 2099</a> )	Available since version 1.7.
◆	NewPlateDefinition ( <a href="#">see page 2099</a> )	Available since version ARSA 2010.
◆	Thick ( <a href="#">see page 2099</a> )	Available since version 1.7.
◆	Type ( <a href="#">see page 2099</a> )	Available since version 1.7.
◆	VOffset ( <a href="#">see page 2100</a> )	Available since version ARSA 2010.

## V.18.2 IRJointGussetFlangePlate Fields

The fields of the IRJointGussetFlangePlate class are listed here.

### Public Fields

	Name	Description
◆	B1 ( <a href="#">see page 2092</a> )	Available since version 1.7.
◆	B2 ( <a href="#">see page 2092</a> )	Available since version 1.7.
◆	B3 ( <a href="#">see page 2093</a> )	Available since version 1.7.
◆	B4 ( <a href="#">see page 2093</a> )	Available since version 1.7.
◆	CornersType ( <a href="#">see page 2093</a> )	Available since version 1.7.
◆	CutH1 ( <a href="#">see page 2093</a> )	Available since ARSA 2010.
◆	CutH2 ( <a href="#">see page 2094</a> )	Available since ARSA 2010.
◆	CutH3 ( <a href="#">see page 2094</a> )	Available since ARSA 2010.
◆	CutH4 ( <a href="#">see page 2094</a> )	Available since ARSA 2010.
◆	CutV1 ( <a href="#">see page 2095</a> )	Available since version ARSA 2010.
◆	CutV2 ( <a href="#">see page 2095</a> )	Available since ARSA 2010.
◆	CutV3 ( <a href="#">see page 2095</a> )	Available since ARSA 2010.
◆	CutV4 ( <a href="#">see page 2096</a> )	Available since ARSA 2010.
◆	EH ( <a href="#">see page 2096</a> )	Available since version 1.7.
◆	EV ( <a href="#">see page 2096</a> )	Available since version 1.7.
◆	Gheight ( <a href="#">see page 2097</a> )	Available since version ARSA 2010.
◆	H1 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H2 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H3 ( <a href="#">see page 2097</a> )	Available since version 1.7.
◆	H4 ( <a href="#">see page 2098</a> )	Available since version 1.7.
◆	HOffset ( <a href="#">see page 2098</a> )	Available since version ARSA 2010.
◆	Length ( <a href="#">see page 2098</a> )	Available since version ARSA 2010.
◆	Material ( <a href="#">see page 2099</a> )	Available since version 1.7.

◆	NewPlateDefinition ( [ see page 2099) )	Available since version ARSA 2010.
◆	Thick ( [ see page 2099)	Available since version 1.7.
◆	Type ( [ see page 2099)	Available since version 1.7.
◆	VOffset ( [ see page 2100)	Available since version ARSA 2010.

### V.18.2.1 B1

**C++**

```
HRESULT get_B1(double* );
HRESULT put_B1(double);
```

**C#**

```
public double B1 { get; set; }
```

**Visual Basic**

```
Public B1 As double
```

**Description**

Available since version 1.7.

### V.18.2.2 B2

**C++**

```
HRESULT get_B2(double* );
HRESULT put_B2(double);
```

**C#**

```
public double B2 { get; set; }
```

**Visual Basic**

```
Public B2 As double
```

**Description**

Available since version 1.7.

### V.18.2.3 B3

**C++**

```
HRESULT get_B3(double* );
HRESULT put_B3(double);
```

**C#**

```
public double B3 { get; set; }
```

**Visual Basic**

```
Public B3 As double
```

**Description**

Available since version 1.7.

### V.18.2.4 B4

**C++**

```
HRESULT get_B4(double* );
HRESULT put_B4(double);
```

**C#**

```
public double B4 { get; set; }
```

**Visual Basic**

```
Public B4 As Double
```

**Description**

Available since version 1.7.

### V.18.2.5 CornersType

**C++**

```
HRESULT get_CornersType(IRJointGussetCornersType* );
HRESULT put_CornersType(IRJointGussetCornersType* );
```

**C#**

```
public IRJointGussetCornersType CornersType { get; set; }
```

**Visual Basic**

```
Public CornersType As IRJointGussetCornersType
```

**Description**

Available since version 1.7.

### V.18.2.6 CutH1

**C++**

```
HRESULT get_CutH1(double* );
HRESULT put_CutH1(double);
```

**C#**

```
public double CutH1 { get; set; }
```

**Visual Basic**

```
Public CutH1 As Double
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

### V.18.2.7 CutH2

**C++**

```
HRESULT get_CutH2(double* );
HRESULT put_CutH2(double);
```

**C#**

```
public double CutH2 { get; set; }
```

**Visual Basic**

```
Public CutH2 As Double
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

**V.18.2.8 CutH3****C++**

```
HRESULT get_CutH3(double*);  
HRESULT put_CutH3(double);
```

**C#**

```
public double CutH3 { get; set; }
```

**Visual Basic**

```
Public CutH3 As double
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

**V.18.2.9 CutH4****C++**

```
HRESULT get_CutH4(double*);  
HRESULT put_CutH4(double);
```

**C#**

```
public double CutH4 { get; set; }
```

**Visual Basic**

```
Public CutH4 As double
```

**Description**

Available since ARSA 2010.

**Version**

Available since version 10.

**V.18.2.10 CutV1****C++**

```
HRESULT get_CutV1(double*);  
HRESULT put_CutV1(double);
```

**C#**

```
public double CutV1 { get; set; }
```

**Visual Basic**

```
Public CutV1 As double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.18.2.11 CutV2

#### C++

```
HRESULT get_CutV2(double*);  
HRESULT put_CutV2(double);
```

#### C#

```
public double CutV2 { get; set; }
```

#### Visual Basic

```
Public CutV2 As double
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.18.2.12 CutV3

#### C++

```
HRESULT get_CutV3(double*);  
HRESULT put_CutV3(double);
```

#### C#

```
public double CutV3 { get; set; }
```

#### Visual Basic

```
Public CutV3 As double
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.18.2.13 CutV4

#### C++

```
HRESULT get_CutV4(double*);  
HRESULT put_CutV4(double);
```

#### C#

```
public double CutV4 { get; set; }
```

#### Visual Basic

```
Public CutV4 As double
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.18.2.14 EH

#### C++

```
HRESULT get_EH(double*);  
HRESULT put_EH(double);
```

#### C#

```
public double EH { get; set; }
```

#### Visual Basic

```
Public EH As double
```

#### Description

Available since version 1.7.

### V.18.2.15 EV

#### C++

```
HRESULT get_EV(double*);  
HRESULT put_EV(double);
```

#### C#

```
public double EV { get; set; }
```

#### Visual Basic

```
Public EV As double
```

#### Description

Available since version 1.7.

### V.18.2.16 Gheight

#### C++

```
HRESULT get_Gheight(double*);  
HRESULT put_Gheight(double);
```

#### C#

```
public double Gheight { get; set; }
```

#### Visual Basic

```
Public Gheight As double
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.18.2.17 H1

#### C++

```
HRESULT get_H1(double*);  
HRESULT put_H1(double);
```

#### C#

```
public double H1 { get; set; }
```

**Visual Basic**

```
Public H1 As double
```

**Description**

Available since version 1.7.

**V.18.2.18 H2****C++**

```
HRESULT get_H2(double*);  
HRESULT put_H2(double);
```

**C#**

```
public double H2 { get; set; }
```

**Visual Basic**

```
Public H2 As double
```

**Description**

Available since version 1.7.

**V.18.2.19 H3****C++**

```
HRESULT get_H3(double*);  
HRESULT put_H3(double);
```

**C#**

```
public double H3 { get; set; }
```

**Visual Basic**

```
Public H3 As double
```

**Description**

Available since version 1.7.

**V.18.2.20 H4****C++**

```
HRESULT get_H4(double*);  
HRESULT put_H4(double);
```

**C#**

```
public double H4 { get; set; }
```

**Visual Basic**

```
Public H4 As double
```

**Description**

Available since version 1.7.

**V.18.2.21 HOffset****C++**

```
HRESULT get_HOffset(double*);  
HRESULT put_HOffset(double);
```

**C#**

```
public double HOffset { get; set; }
```

**Visual Basic**

```
Public HOffset As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.18.2.22 Length

**C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

### V.18.2.23 Material

**C++**

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Available since version 1.7.

### V.18.2.24 NewPlateDefinition

**C++**

```
HRESULT get_NewPlateDefinition(VARIANT_BOOL*);  
HRESULT put_NewPlateDefinition(VARIANT_BOOL);
```

**C#**

```
public bool NewPlateDefinition { get; set; }
```

**Visual Basic**

```
Public NewPlateDefinition As Boolean
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.18.2.25 Thick****C++**

```
HRESULT get_Thickness(double*);  
HRESULT put_Thickness(double);
```

**C#**

```
public double Thickness { get; set; }
```

**Visual Basic**

```
Public Thickness As Double
```

**Description**

Available since version 1.7.

**V.18.2.26 Type****C++**

```
HRESULT get_Type(IRJointGussetFlangePlateRegularType*);  
HRESULT put_Type(IRJointGussetFlangePlateRegularType);
```

**C#**

```
public IRJointGussetFlangePlateRegularType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointGussetFlangePlateRegularType
```

**Description**

Available since version 1.7.

**V.18.2.27 VOffset****C++**

```
HRESULT get_VOffset(double*);  
HRESULT put_VOffset(double);
```

**C#**

```
public double VOffset { get; set; }
```

**Visual Basic**

```
Public VOffset As Double
```

**Description**

Available since version ARSA 2010.

**Version**

Available since version 10.

**V.19 IRJointGussetFlange****Class Hierarchy**

**C++**

```
interface IRJointGussetFlange : IRJointConnection;
```

**C#**

```
public interface IRJointGussetFlange : IRJointConnection;
```

**Visual Basic**

```
Public Interface IRJointGussetFlange
```

**V.19.1 IRJointGussetFlange Members**

The following tables list the members exposed by IRJointGussetFlange.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BoltsClassVector ( <a href="#">see page 2102</a> )	Available since version ARSA 2010.
❖	BoltsDimensionNamesVector ( <a href="#">see page 2102</a> )	Available since ARSA 2010.
❖	BoltsDimensionsVector ( <a href="#">see page 2102</a> )	Available since version ARSA 2010.
❖	DiagonalLeft ( <a href="#">see page 2103</a> )	Available since version 1.7.
❖	DiagonalRight ( <a href="#">see page 2103</a> )	Available since version 1.7.
❖	DiagonalUp ( <a href="#">see page 2103</a> )	Available since version 1.7.
❖	FixType_DiagonalLeft ( <a href="#">see page 2103</a> )	Available since version 1.7.
❖	FixType_DiagonalRight ( <a href="#">see page 2104</a> )	Available since version 1.7.
❖	FixType_DiagonalUp ( <a href="#">see page 2104</a> )	Available since version 1.7.
❖	FixType_FlangeLeft ( <a href="#">see page 2104</a> )	Available since version 1.7.
❖	FixType_FlangeRight ( <a href="#">see page 2104</a> )	Available since version 1.7.
❖	FlangeLeft ( <a href="#">see page 2105</a> )	Available since version 1.7.
❖	FlangeRight ( <a href="#">see page 2105</a> )	Available since version 1.7.
❖	GussetPlate ( <a href="#">see page 2105</a> )	Available since version 1.7.
❖	ProfileCutting ( <a href="#">see page 2105</a> )	Available since version 1.7.
❖	Type ( <a href="#">see page 2148</a> )	A connection type.
❖	WFRelPos ( <a href="#">see page 2148</a> )	A type of relative position of a beam and a column.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	GetFromRobot ( <a href="#">see page 2149</a> )	The function gets connection parameters from Robot and fills an object with the data.
❖	SetToRobot ( <a href="#">see page 2149</a> )	The function sets and saves connection parameter in Robot.

**V.19.2 IRJointGussetFlange Fields**

The fields of the IRJointGussetFlange class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BoltsClassVector ( <a href="#">see page 2102</a> )	Available since version ARSA 2010.

◆	BoltsDimensionNamesVector ( [ see page 2102) )	Available since ARSA 2010.
◆	BoltsDimensionsVector ( [ see page 2102)	Available since version ARSA 2010.
◆	DiagonalLeft ( [ see page 2103)	Available since version 1.7.
◆	DiagonalRight ( [ see page 2103)	Available since version 1.7.
◆	DiagonalUp ( [ see page 2103)	Available since version 1.7.
◆	FixType_DiagonalLeft ( [ see page 2103)	Available since version 1.7.
◆	FixType_DiagonalRight ( [ see page 2104)	Available since version 1.7.
◆	FixType_DiagonalUp ( [ see page 2104)	Available since version 1.7.
◆	FixType_FlangeLeft ( [ see page 2104)	Available since version 1.7.
◆	FixType_FlangeRight ( [ see page 2104)	Available since version 1.7.
◆	FlangeLeft ( [ see page 2105)	Available since version 1.7.
◆	FlangeRight ( [ see page 2105)	Available since version 1.7.
◆	GussetPlate ( [ see page 2105)	Available since version 1.7.
◆	ProfileCutting ( [ see page 2105)	Available since version 1.7.

### V.19.2.1 BoltsClassVector

#### C++

```
HRESULT get_BoltsClassVector( IRobotNamesArray** );
```

#### C#

```
public IRobotNamesArray BoltsClassVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsClassVector As IRobotNamesArray
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.19.2.2 BoltsDimensionNamesVector

#### C++

```
HRESULT get_BoltsDimensionNamesVector( IRobotNamesArray** );
```

#### C#

```
public IRobotNamesArray BoltsDimensionNamesVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsDimensionNamesVector As IRobotNamesArray
```

#### Description

Available since ARSA 2010.

#### Version

Available since version 10.

### V.19.2.3 BoltsDimensionsVector

#### C++

```
HRESULT get_BoltsDimensionsVector(IRobotValuesArray**);
```

#### C#

```
public IRobotValuesArray BoltsDimensionsVector { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsDimensionsVector As IRobotValuesArray
```

#### Description

Available since version ARSA 2010.

#### Version

Available since version 10.

### V.19.2.4 DiagonalLeft

#### C++

```
HRESULT get_DiagonalLeft(IRJointGussetDiagonale**);
```

#### C#

```
public IRJointGussetDiagonale DiagonalLeft { get; }
```

#### Visual Basic

```
Public ReadOnly DiagonalLeft As IRJointGussetDiagonale
```

#### Description

Available since version 1.7.

### V.19.2.5 DiagonalRight

#### C++

```
HRESULT get_DiagonalRight(IRJointGussetDiagonale**);
```

#### C#

```
public IRJointGussetDiagonale DiagonalRight { get; }
```

#### Visual Basic

```
Public ReadOnly DiagonalRight As IRJointGussetDiagonale
```

#### Description

Available since version 1.7.

### V.19.2.6 DiagonalUp

#### C++

```
HRESULT get_DiagonalUp(IRJointGussetDiagonale**);
```

#### C#

```
public IRJointGussetDiagonale DiagonalUp { get; }
```

#### Visual Basic

```
Public ReadOnly DiagonalUp As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

**V.19.2.7 FixType\_DiagonalLeft****C++**

```
HRESULT get_FixType_DiagonalLeft(IRJointGussetFixType* );
HRESULT put_FixType_DiagonalLeft(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_DiagonalLeft { get; set; }
```

**Visual Basic**

```
Public FixType_DiagonalLeft As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.19.2.8 FixType\_DiagonalRight****C++**

```
HRESULT get_FixType_DiagonalRight(IRJointGussetFixType* );
HRESULT put_FixType_DiagonalRight(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_DiagonalRight { get; set; }
```

**Visual Basic**

```
Public FixType_DiagonalRight As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.19.2.9 FixType\_DiagonalUp****C++**

```
HRESULT get_FixType_DiagonalUp(IRJointGussetFixType* );
HRESULT put_FixType_DiagonalUp(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_DiagonalUp { get; set; }
```

**Visual Basic**

```
Public FixType_DiagonalUp As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.19.2.10 FixType\_FlangeLeft****C++**

```
HRESULT get_FixType_FlangeLeft(IRJointGussetFixType* );
HRESULT put_FixType_FlangeLeft(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_FlangeLeft { get; set; }
```

**Visual Basic**

```
Public FixType_FlangeLeft As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.19.2.11 FixType\_FlangeRight****C++**

```
HRESULT get_FixType_FlangeRight(IRJointGussetFixType* );
HRESULT put_FixType_FlangeRight(IRJointGussetFixType);
```

**C#**

```
public IRJointGussetFixType FixType_FlangeRight { get; set; }
```

**Visual Basic**

```
Public FixType_FlangeRight As IRJointGussetFixType
```

**Description**

Available since version 1.7.

**V.19.2.12 FlangeLeft****C++**

```
HRESULT get_FlangeLeft(IRJointGussetDiagonale** );
```

**C#**

```
public IRJointGussetDiagonale FlangeLeft { get; }
```

**Visual Basic**

```
Public ReadOnly FlangeLeft As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

**V.19.2.13 FlangeRight****C++**

```
HRESULT get_FlangeRight(IRJointGussetDiagonale** );
```

**C#**

```
public IRJointGussetDiagonale FlangeRight { get; }
```

**Visual Basic**

```
Public ReadOnly FlangeRight As IRJointGussetDiagonale
```

**Description**

Available since version 1.7.

**V.19.2.14 GussetPlate****C++**

```
HRESULT get_GussetPlate(IRJointGussetFlangePlate** );
```

**C#**

```
public IRJointGussetFlangePlate GussetPlate { get; }
```

**Visual Basic**

```
Public ReadOnly GussetPlate As IRJointGussetFlangePlate
```

**Description**

Available since version 1.7.

**V.19.2.15 ProfileCutting****C++**

```
HRESULT get_ProfileCutting(IRJointGussetFlangeProfileCutting* );
HRESULT put_ProfileCutting(IRJointGussetFlangeProfileCutting);
```

**C#**

```
public IRJointGussetFlangeProfileCutting ProfileCutting { get; set; }
```

**Visual Basic**

```
Public ProfileCutting As IRJointGussetFlangeProfileCutting
```

**Description**

Available since version 1.7.

**V.20 IRJointGussetSimpleLoad****Class Hierarchy****C++**

```
interface IRJointGussetSimpleLoad : IRJointLoad;
```

**C#**

```
public interface IRJointGussetSimpleLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointGussetSimpleLoad
```

**V.20.1 IRJointGussetSimpleLoad Members**

The following tables list the members exposed by IRJointGussetSimpleLoad.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Cases ( [ see page 2166)	Selection of load cases, defined for the structure. .
❖	Fx ( [ see page 2106)	Available since version 1.7.
❖	Type ( [ see page 2166)	Load type (defined manually or taken from Robot).

**V.20.2 IRJointGussetSimpleLoad Fields**

The fields of the IRJointGussetSimpleLoad class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Fx ( [ see page 2106)	Available since version 1.7.

**V.20.2.1 Fx****C++**

```
HRESULT get_Fx(double* );
```

```
HRESULT put_Fx(double);
```

**C#**

```
public double Fx { get; set; }
```

**Visual Basic**

```
Public Fx As Double
```

**Description**

Available since version 1.7.

## V.21 IRJointGussetCrossLoad

**Class Hierarchy****C++**

```
interface IRJointGussetCrossLoad : IRJointLoad;
```

**C#**

```
public interface IRJointGussetCrossLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointGussetCrossLoad
```

### V.21.1 IRJointGussetCrossLoad Members

The following tables list the members exposed by IRJointGussetCrossLoad.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Cases (see page 2166)	Selection of load cases, defined for the structure..
❖	Fx_LeftLower (see page 2107)	Available since version 1.7.
❖	Fx_LeftUpper (see page 2108)	Available since version 1.7.
❖	Fx_RightLower (see page 2108)	Available since version 1.7.
❖	Fx_RightUpper (see page 2108)	Available since version 1.7.
❖	Type (see page 2166)	Load type (defined manually or taken from Robot).

### V.21.2 IRJointGussetCrossLoad Fields

The fields of the IRJointGussetCrossLoad class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Fx_LeftLower (see page 2107)	Available since version 1.7.
❖	Fx_LeftUpper (see page 2108)	Available since version 1.7.
❖	Fx_RightLower (see page 2108)	Available since version 1.7.
❖	Fx_RightUpper (see page 2108)	Available since version 1.7.

#### V.21.2.1 Fx\_LeftLower

**C++**

```
HRESULT get_Fx_LeftLower(double*);  
HRESULT put_Fx_LeftLower(double);
```

**C#**

```
public double Fx_LeftLower { get; set; }
```

**Visual Basic**

```
Public Fx_LeftLower As Double
```

**Description**

Available since version 1.7.

### V.21.2.2 Fx\_LeftUpper

**C++**

```
HRESULT get_Fx_LeftUpper(double*);  
HRESULT put_Fx_LeftUpper(double);
```

**C#**

```
public double Fx_LeftUpper { get; set; }
```

**Visual Basic**

```
Public Fx_LeftUpper As Double
```

**Description**

Available since version 1.7.

### V.21.2.3 Fx\_RightLower

**C++**

```
HRESULT get_Fx_RightLower(double*);  
HRESULT put_Fx_RightLower(double);
```

**C#**

```
public double Fx_RightLower { get; set; }
```

**Visual Basic**

```
Public Fx_RightLower As Double
```

**Description**

Available since version 1.7.

### V.21.2.4 Fx\_RightUpper

**C++**

```
HRESULT get_Fx_RightUpper(double*);  
HRESULT put_Fx_RightUpper(double);
```

**C#**

```
public double Fx_RightUpper { get; set; }
```

**Visual Basic**

```
Public Fx_RightUpper As Double
```

**Description**

Available since version 1.7.

## V.22 IRJointGussetFlangeLoad

**Class Hierarchy**

**C++**

```
interface IRJointGussetFlangeLoad : IRJointLoad;
```

**C#**

```
public interface IRJointGussetFlangeLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointGussetFlangeLoad
```

**V.22.1 IRJointGussetFlangeLoad Members**

The following tables list the members exposed by IRJointGussetFlangeLoad.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Cases (see page 2166)	Selection of load cases, defined for the structure. .
❖	Fx_FlangeLeft (see page 2109)	Available since version 1.7.
❖	Fx_FlangeRight (see page 2110)	Available since version 1.7.
❖	Fx_LeftUpper (see page 2110)	Available since version 1.7.
❖	Fx_RightUpper (see page 2110)	Available since version 1.7.
❖	Fx_Upper (see page 2110)	Available since version 1.7.
❖	Type (see page 2166)	Load type (defined manually or taken from Robot).

**V.22.2 IRJointGussetFlangeLoad Fields**

The fields of the IRJointGussetFlangeLoad class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Fx_FlangeLeft (see page 2109)	Available since version 1.7.
❖	Fx_FlangeRight (see page 2110)	Available since version 1.7.
❖	Fx_LeftUpper (see page 2110)	Available since version 1.7.
❖	Fx_RightUpper (see page 2110)	Available since version 1.7.
❖	Fx_Upper (see page 2110)	Available since version 1.7.

**V.22.2.1 Fx\_FlangeLeft****C++**

```
HRESULT get_Fx_FlangeLeft(double* );
HRESULT put_Fx_FlangeLeft(double);
```

**C#**

```
public double Fx_FlangeLeft { get; set; }
```

**Visual Basic**

```
Public Fx_FlangeLeft As Double
```

**Description**

Available since version 1.7.

**V.22.2.2 Fx\_FlangeRight****C++**

```
HRESULT get_Fx_FlangeRight(double* );
HRESULT put_Fx_FlangeRight(double);
```

**C#**

```
public double Fx_FlangeRight { get; set; }
```

**Visual Basic**

```
Public Fx_FlangeRight As Double
```

**Description**

Available since version 1.7.

**V.22.2.3 Fx\_LeftUpper****C++**

```
HRESULT get_Fx_LeftUpper(double*);  
HRESULT put_Fx_LeftUpper(double);
```

**C#**

```
public double Fx_LeftUpper { get; set; }
```

**Visual Basic**

```
Public Fx_LeftUpper As Double
```

**Description**

Available since version 1.7.

**V.22.2.4 Fx\_RightUpper****C++**

```
HRESULT get_Fx_RightUpper(double*);  
HRESULT put_Fx_RightUpper(double);
```

**C#**

```
public double Fx_RightUpper { get; set; }
```

**Visual Basic**

```
Public Fx_RightUpper As Double
```

**Description**

Available since version 1.7.

**V.22.2.5 Fx\_Upper****C++**

```
HRESULT get_Fx_Upper(double*);  
HRESULT put_Fx_Upper(double);
```

**C#**

```
public double Fx_Upper { get; set; }
```

**Visual Basic**

```
Public Fx_Upper As Double
```

**Description**

Available since version 1.7.

## V.23 IRJointGussetDiagonalePositionType

### C++

```
enum IRJointGussetDiagonalePositionType;
```

### C#

```
public enum IRJointGussetDiagonalePositionType;
```

### Visual Basic

```
Public Enum IRJointGussetDiagonalePositionType
```

### Members

Members	Description
I_JGDPT_LONG_FLANGE = 0	Available since version 1.7.
I_JGDPT_SHORT_FLANGE = 1	Available since version 1.7.
I_JGDPT_LONG_FLANGE_INV = 2	Available since version 1.7.
I_JGDPT_SHORT_FLANGE_INV = 3	Available since version 1.7.

## V.24 IRJointGussetFlangePlateRegularType

### C++

```
enum IRJointGussetFlangePlateRegularType;
```

### C#

```
public enum IRJointGussetFlangePlateRegularType;
```

### Visual Basic

```
Public Enum IRJointGussetFlangePlateRegularType
```

### Members

Members	Description
I_JGFPRT_REGULAR = 0	Available since version 1.7.
I_JGFPRT_IRREGULAR = 1	Available since version 1.7.

## VI Beam-principal beam connection

Available since version 7.5.

### Enumerations

	Name	Description
	IRJointConnectorsType (see page 2140)	
	IRJointElementType (see page 2140)	

### Interfaces

	Name	Description
	IRJointBeamGirderBolts (see page 2112)	Bolt description for stiffener and plate.
	IRJointBeamGirderStiffener (see page 2116)	Stiffener description.

	IRJointBeamGirderPlate ( <a href="#">see page 2120</a> )	Plate description.
	IRJointBeamGirderSeat ( <a href="#">see page 2124</a> )	Description of seat profile.
	IRJointBeamGirder ( <a href="#">see page 2127</a> )	Beam-principal beam connection.
	IRJointBeamGirderLoad ( <a href="#">see page 2134</a> )	
	IRJointBeamGirderSeatBolts ( <a href="#">see page 2137</a> )	
	IRJointBeamGirderAngle ( <a href="#">see page 2141</a> )	
	IRJointBeamGirderBeam ( <a href="#">see page 2143</a> )	

## VI.1 IRJointBeamGirderBolts

### Class Hierarchy

#### C++

```
interface IRJointBeamGirderBolts : IDispatch;
```

#### C#

```
public interface IRJointBeamGirderBolts;
```

#### Visual Basic

```
Public Interface IRJointBeamGirderBolts
```

### Description

Bolt description for stiffener and plate.

### Version

Available since version 7.5.

## VI.1.1 IRJointBeamGirderBolts Members

The following tables list the members exposed by IRJointBeamGirderBolts.

### Public Fields

	Name	Description
	Area ( <a href="#">see page 2113</a> )	
	ClassName ( <a href="#">see page 2113</a> )	Bolt class.
	Cols ( <a href="#">see page 2114</a> )	
	Diameter ( <a href="#">see page 2114</a> )	Bolt diameter.
	DiameterName ( <a href="#">see page 2114</a> )	
	DistFromUpperBeamEdge ( <a href="#">see page 2115</a> )	Distance of 1st bolt row to upper beam edge.
	DistFromVertBeamEdge ( <a href="#">see page 2115</a> )	Distance of 1st bolt column to vertical beam edge.
	Friction ( <a href="#">see page 2115</a> )	
	Rows ( <a href="#">see page 2115</a> )	Number of bolt rows.
	SpacingH ( <a href="#">see page 2116</a> )	Horizontal spacing.
	SpacingV ( <a href="#">see page 2116</a> )	Vertical spacing.

## VI.1.2 IRJointBeamGirderBolts Fields

The fields of the IRJointBeamGirderBolts class are listed here.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 2113</a> )	
◆	ClassName ( <a href="#">see page 2113</a> )	Bolt class.
◆	Cols ( <a href="#">see page 2114</a> )	
◆	Diameter ( <a href="#">see page 2114</a> )	Bolt diameter.
◆	DiameterName ( <a href="#">see page 2114</a> )	
◆	DistFromUpperBeamEdge ( <a href="#">see page 2115</a> )	Distance of 1st bolt row to upper beam edge.
◆	DistFromVertBeamEdge ( <a href="#">see page 2115</a> )	Distance of 1st bolt column to vertical beam edge.
◆	Friction ( <a href="#">see page 2115</a> )	
◆	Rows ( <a href="#">see page 2115</a> )	Number of bolt rows.
◆	SpacingH ( <a href="#">see page 2116</a> )	Horizontal spacing.
◆	SpacingV ( <a href="#">see page 2116</a> )	Vertical spacing.

### VI.1.2.1 Area

#### C++

```
HRESULT get_Area( double* );
```

#### C#

```
public double Area { get; }
```

#### Visual Basic

```
Public ReadOnly Area As Double
```

#### Version

Available since version 7.5.

### VI.1.2.2 ClassName

#### C++

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

#### C#

```
public String ClassName { get; set; }
```

#### Visual Basic

```
Public ClassName As String
```

#### Description

Bolt class.

#### Version

Available since version 7.5.

### VI.1.2.3 Cols

#### C++

```
HRESULT get_Cols( long* );
HRESULT put_Cols( long );
```

#### C#

```
public long Cols { get; set; }
```

#### Visual Basic

```
Public Cols As long
```

#### Version

Available since version 7.5.

### VI.1.2.4 Diameter

#### C++

```
HRESULT get_Diameter( long* );
```

#### C#

```
public long Diameter { get; }
```

#### Visual Basic

```
Public ReadOnly Diameter As long
```

#### Description

Bolt diameter.

#### Version

Available since version 7.5.

### VI.1.2.5 DiameterName

#### C++

```
HRESULT get_DiameterName(BSTR* );
HRESULT put_DiameterName(BSTR);
```

#### C#

```
public String DiameterName { get; set; }
```

#### Visual Basic

```
Public DiameterName As String
```

#### Version

Available since version 7.5.

### VI.1.2.6 DistFromUpperBeamEdge

#### C++

```
HRESULT get_DistFromUpperBeamEdge( double* );
HRESULT put_DistFromUpperBeamEdge( double );
```

#### C#

```
public double DistFromUpperBeamEdge { get; set; }
```

**Visual Basic**

```
Public DistFromUpperBeamEdge As double
```

**Description**

Distance of 1st bolt row to upper beam edge.

**Version**

Available since version 7.5.

**VI.1.2.7 DistFromVertBeamEdge****C++**

```
HRESULT get_DistFromVertBeamEdge(double*);  
HRESULT put_DistFromVertBeamEdge(double);
```

**C#**

```
public double DistFromVertBeamEdge { get; set; }
```

**Visual Basic**

```
Public DistFromVertBeamEdge As double
```

**Description**

Distance of 1st bolt column to vertical beam edge.

**Version**

Available since version 7.5.

**VI.1.2.8 Friction****C++**

```
HRESULT get_Friction( double* );  
HRESULT put_Friction( double );
```

**C#**

```
public double Friction { get; set; }
```

**Visual Basic**

```
Public Friction As double
```

**Version**

Available since version 7.5.

**VI.1.2.9 Rows****C++**

```
HRESULT get_Rows( long* );  
HRESULT put_Rows( long );
```

**C#**

```
public long Rows { get; set; }
```

**Visual Basic**

```
Public Rows As long
```

**Description**

Number of bolt rows.

**Version**

Available since version 7.5.

**VI.1.2.10 SpacingH****C++**

```
HRESULT get_SpacingH( double* );
HRESULT put_SpacingH( double );
```

**C#**

```
public double SpacingH { get; set; }
```

**Visual Basic**

```
Public SpacingH As Double
```

**Description**

Horizontal spacing.

**Version**

Available since version 7.5.

**VI.1.2.11 SpacingV****C++**

```
HRESULT get_SpacingV( double* );
HRESULT put_SpacingV( double );
```

**C#**

```
public double SpacingV { get; set; }
```

**Visual Basic**

```
Public SpacingV As Double
```

**Description**

Vertical spacing.

**Version**

Available since version 7.5.

**VI.2 IRJointBeamGirderStiffener****Class Hierarchy****C++**

```
interface IRJointBeamGirderStiffener : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderStiffener;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderStiffener
```

**Description**

Stiffener description.

## Version

Available since version 7.5.

### VI.2.1 IRJointBeamGirderStiffener Members

The following tables list the members exposed by IRJointBeamGirderStiffener.

#### Public Fields

	Name	Description
◆	BoltsBeam ( <a href="#">see page 2118</a> )	Bolts for stiffener-beam connection.
◆	ConnectorsType ( <a href="#">see page 2118</a> )	Connector type for stiffener-beam connection.
◆	CutHeightDown ( <a href="#">see page 2118</a> )	
◆	CutHeightUp ( <a href="#">see page 2118</a> )	
◆	CutLength ( <a href="#">see page 2119</a> )	Length ( <a href="#">see page 2119</a> ) of stiffener cut-off.
◆	Length ( <a href="#">see page 2119</a> )	Stiffener length.
◆	Material ( <a href="#">see page 2119</a> )	Stiffener material.
◆	Thick ( <a href="#">see page 2120</a> )	Stiffener thickness.
◆	WeldsBeam ( <a href="#">see page 2120</a> )	Welds for stiffener-beam connection.

### VI.2.2 IRJointBeamGirderStiffener Fields

The fields of the IRJointBeamGirderStiffener class are listed here.

#### Public Fields

	Name	Description
◆	BoltsBeam ( <a href="#">see page 2118</a> )	Bolts for stiffener-beam connection.
◆	ConnectorsType ( <a href="#">see page 2118</a> )	Connector type for stiffener-beam connection.
◆	CutHeightDown ( <a href="#">see page 2118</a> )	
◆	CutHeightUp ( <a href="#">see page 2118</a> )	
◆	CutLength ( <a href="#">see page 2119</a> )	Length ( <a href="#">see page 2119</a> ) of stiffener cut-off.
◆	Length ( <a href="#">see page 2119</a> )	Stiffener length.
◆	Material ( <a href="#">see page 2119</a> )	Stiffener material.
◆	Thick ( <a href="#">see page 2120</a> )	Stiffener thickness.
◆	WeldsBeam ( <a href="#">see page 2120</a> )	Welds for stiffener-beam connection.

#### VI.2.2.1 BoltsBeam

##### C++

```
HRESULT get_BoltsBeam( IRJointBeamGirderBolts** );
```

##### C#

```
public IRJointBeamGirderBolts BoltsBeam { get; }
```

##### Visual Basic

```
Public ReadOnly BoltsBeam As IRJointBeamGirderBolts
```

#### Description

Bolts for stiffener-beam connection.

#### Version

Available since version 7.5.

### VI.2.2.2 ConnectorsType

#### C++

```
HRESULT get_ConnectorsType(IRJointConnectorsType* );
HRESULT put_ConnectorsType(IRJointConnectorsType);
```

#### C#

```
public IRJointConnectorsType ConnectorsType { get; set; }
```

#### Visual Basic

```
Public ConnectorsType As IRJointConnectorsType
```

#### Description

Connector type for stiffener-beam connection.

#### Version

Available since version 7.5.

### VI.2.2.3 CutHeightDown

#### C++

```
HRESULT get_CutHeightDown(double* );
HRESULT put_CutHeightDown(double);
```

#### C#

```
public double CutHeightDown { get; set; }
```

#### Visual Basic

```
Public CutHeightDown As double
```

#### Version

Available since version 7.5.

### VI.2.2.4 CutHeightUp

#### C++

```
HRESULT get_CutHeightUp(double* );
HRESULT put_CutHeightUp(double);
```

#### C#

```
public double CutHeightUp { get; set; }
```

#### Visual Basic

```
Public CutHeightUp As double
```

#### Version

Available since version 7.5.

### VI.2.2.5 CutLength

#### C++

```
HRESULT get_CutLength(double* );
HRESULT put_CutLength(double);
```

#### C#

```
public double CutLength { get; set; }
```

**Visual Basic**

```
Public CutLength As double
```

**Description**

Length (█ see page 2119) of stiffener cut-off.

**Version**

Available since version 7.5.

**VI.2.2.6 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As double
```

**Description**

Stiffener length.

**Version**

Available since version 7.5.

**VI.2.2.7 Material****C++**

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Stiffener material.

**Version**

Available since version 7.5.

**VI.2.2.8 Thick****C++**

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As double
```

**Description**

Stiffener thickness.

**Version**

Available since version 7.5.

**VI.2.2.9 WeldsBeam****C++**

```
HRESULT get_WeldsBeam(IRJointWeld**);
```

**C#**

```
public IRJointWeld WeldsBeam { get; }
```

**Visual Basic**

```
Public ReadOnly WeldsBeam As IRJointWeld
```

**Description**

Welds for stiffener-beam connection.

**Version**

Available since version 7.5.

**VI.3 IRJointBeamGirderPlate****Class Hierarchy****C++**

```
interface IRJointBeamGirderPlate : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderPlate;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderPlate
```

**Description**

Plate description.

**Version**

Available since version 7.5.

**VI.3.1 IRJointBeamGirderPlate Members**

The following tables list the members exposed by IRJointBeamGirderPlate.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	BoltsBeam ( <a href="#">see page 2121</a> )	Bolts connecting plate with beam.
❖	ConnectorsType ( <a href="#">see page 2122</a> )	Connector type for the plate-beam connection.
❖	DistFromUpperBeamEdge ( <a href="#">see page 2122</a> )	Distance of upper plate edge to upper beam edge.
❖	Length ( <a href="#">see page 2122</a> )	Plate length.
❖	Material ( <a href="#">see page 2123</a> )	Material name for plate.
❖	Thick ( <a href="#">see page 2123</a> )	Plate thickness.

❖	WeldsBeam (see page 2123)	Welds connecting plate with beam.
❖	WeldsGirder (see page 2124)	Welds connecting principal beam flange with plate.
❖	Width (see page 2124)	Plate width.

## VI.3.2 IRJointBeamGirderPlate Fields

The fields of the IRJointBeamGirderPlate class are listed here.

### Public Fields

	Name	Description
❖	BoltsBeam (see page 2121)	Bolts connecting plate with beam.
❖	ConnectorsType (see page 2122)	Connector type for the plate-beam connection.
❖	DistFromUpperBeamEdge (see page 2122)	Distance of upper plate edge to upper beam edge.
❖	Length (see page 2122)	Plate length.
❖	Material (see page 2123)	Material name for plate.
❖	Thick (see page 2123)	Plate thickness.
❖	WeldsBeam (see page 2123)	Welds connecting plate with beam.
❖	WeldsGirder (see page 2124)	Welds connecting principal beam flange with plate.
❖	Width (see page 2124)	Plate width.

### VI.3.2.1 BoltsBeam

#### C++

```
HRESULT get_BoltsBeam(IRJointBeamGirderBolts**);
```

#### C#

```
public IRJointBeamGirderBolts BoltsBeam { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsBeam As IRJointBeamGirderBolts
```

#### Description

Bolts connecting plate with beam.

#### Version

Available since version 7.5.

### VI.3.2.2 ConnectorsType

#### C++

```
HRESULT get_ConnectorsType(IRJointConnectorsType*);  
HRESULT put_ConnectorsType(IRJointConnectorsType);
```

#### C#

```
public IRJointConnectorsType ConnectorsType { get; set; }
```

#### Visual Basic

```
Public ConnectorsType As IRJointConnectorsType
```

#### Description

Connector type for the plate-beam connection.

#### Version

Available since version 7.5.

### VI.3.2.3 DistFromUpperBeamEdge

#### C++

```
HRESULT get_DistFromUpperBeamEdge( double* );
HRESULT put_DistFromUpperBeamEdge( double );
```

#### C#

```
public double DistFromUpperBeamEdge { get; set; }
```

#### Visual Basic

```
Public DistFromUpperBeamEdge As Double
```

#### Description

Distance of upper plate edge to upper beam edge.

#### Version

Available since version 7.5.

### VI.3.2.4 Length

#### C++

```
HRESULT get_Length( double* );
HRESULT put_Length( double );
```

#### C#

```
public double Length { get; set; }
```

#### Visual Basic

```
Public Length As Double
```

#### Description

Plate length.

#### Version

Available since version 7.5.

### VI.3.2.5 Material

#### C++

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

#### C#

```
public String Material { get; set; }
```

#### Visual Basic

```
Public Material As String
```

#### Description

Material name for plate.

#### Version

Available since version 7.5.

### VI.3.2.6 Thick

#### C++

```
HRESULT get_Thick( double* );
HRESULT put_Thick( double );
```

#### C#

```
public double Thick { get; set; }
```

#### Visual Basic

```
Public Thick As Double
```

#### Description

Plate thickness.

#### Version

Available since version 7.5.

### VI.3.2.7 WeldsBeam

#### C++

```
HRESULT get_WeldsBeam( IRJointWeld** );
```

#### C#

```
public IRJointWeld WeldsBeam { get; }
```

#### Visual Basic

```
Public ReadOnly WeldsBeam As IRJointWeld
```

#### Description

Welds connecting plate with beam.

#### Version

Available since version 7.5.

### VI.3.2.8 WeldsGirder

#### C++

```
HRESULT get_WeldsGirder( IRJointWeld** );
```

#### C#

```
public IRJointWeld WeldsGirder { get; }
```

#### Visual Basic

```
Public ReadOnly WeldsGirder As IRJointWeld
```

#### Description

Welds connecting principal beam flange with plate.

#### Version

Available since version 7.5.

### VI.3.2.9 Width

#### C++

```
HRESULT get_Width( double* );
HRESULT put_Width( double );
```

**C#**

```
public double Width { get; set; }
```

**Visual Basic**

```
Public Width As Double
```

**Description**

Plate width.

**Version**

Available since version 7.5.

## VI.4 IRJointBeamGirderSeat

**Class Hierarchy****C++**

```
interface IRJointBeamGirderSeat : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderSeat;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderSeat
```

**Description**

Description of seat profile.

**Version**

Available since version 7.5.

### VI.4.1 IRJointBeamGirderSeat Members

The following tables list the members exposed by IRJointBeamGirderSeat.

**Public Fields**

	Name	Description
❖	BoltsBeam ( <a href="#">see page 2125</a> )	Bolts connecting seat profile and beam flange.
❖	BoltsGirder ( <a href="#">see page 2126</a> )	Bolts connecting seat profile and principal beam web.
❖	ConnectorsToWebType ( <a href="#">see page 2126</a> )	Connector type for seat profile-principal beam web connection.
❖	Length ( <a href="#">see page 2126</a> )	Length of seat profile.
❖	Material ( <a href="#">see page 2126</a> )	Name of seat profile material.
❖	Section ( <a href="#">see page 2127</a> )	Section name.
❖	WeldsGirder ( <a href="#">see page 2127</a> )	Welds connecting seat profile and principal beam web.

### VI.4.2 IRJointBeamGirderSeat Fields

The fields of the IRJointBeamGirderSeat class are listed here.

**Public Fields**

	Name	Description
❖	BoltsBeam ( <a href="#">see page 2125</a> )	Bolts connecting seat profile and beam flange.
❖	BoltsGirder ( <a href="#">see page 2126</a> )	Bolts connecting seat profile and principal beam web.

◆	ConnectorsToWebType (see page 2126)	Connector type for seat profile-principal beam web connection.
◆	Length (see page 2126)	Length of seat profile.
◆	Material (see page 2126)	Name of seat profile material.
◆	Section (see page 2127)	Section name.
◆	WeldsGirder (see page 2127)	Welds connecting seat profile and principal beam web.

#### VI.4.2.1 BoltsBeam

##### C++

```
HRESULT get_BoltsBeam(IRJointBeamGirderSeatBolts**);
```

##### C#

```
public IRJointBeamGirderSeatBolts BoltsBeam { get; }
```

##### Visual Basic

```
Public ReadOnly BoltsBeam As IRJointBeamGirderSeatBolts
```

##### Description

Bolts connecting seat profile and beam flange.

##### Version

Available since version 7.5.

#### VI.4.2.2 BoltsGirder

##### C++

```
HRESULT get_BoltsGirder(IRJointBeamGirderSeatBolts**);
```

##### C#

```
public IRJointBeamGirderSeatBolts BoltsGirder { get; }
```

##### Visual Basic

```
Public ReadOnly BoltsGirder As IRJointBeamGirderSeatBolts
```

##### Description

Bolts connecting seat profile and principal beam web.

##### Version

Available since version 7.5.

#### VI.4.2.3 ConnectorsToWebType

##### C++

```
HRESULT get_ConnectorsToWebType(IRJointConnectorsType*);  
HRESULT put_ConnectorsToWebType(IRJointConnectorsType);
```

##### C#

```
public IRJointConnectorsType ConnectorsToWebType { get; set; }
```

##### Visual Basic

```
Public ConnectorsToWebType As IRJointConnectorsType
```

##### Description

Connector type for seat profile-principal beam web connection.

**Version**

Available since version 7.5.

**VI.4.2.4 Length****C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Description**

Length of seat profile.

**Version**

Available since version 7.5.

**VI.4.2.5 Material****C++**

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Name of seat profile material.

**Version**

Available since version 7.5.

**VI.4.2.6 Section****C++**

```
HRESULT get_Section(BSTR*);  
HRESULT put_Section(BSTR);
```

**C#**

```
public String Section { get; set; }
```

**Visual Basic**

```
Public Section As String
```

**Description**

Section name.

**Version**

Available since version 7.5.

### VI.4.2.7 WeldsGirder

#### C++

```
HRESULT get_WeldsGirder(IRJointWeld**);
```

#### C#

```
public IRJointWeld WeldsGirder { get; }
```

#### Visual Basic

```
Public ReadOnly WeldsGirder As IRJointWeld
```

#### Description

Welds connecting seat profile and principal beam web.

#### Version

Available since version 7.5.

## VI.5 IRJointBeamGirder

#### Class Hierarchy

#### C++

```
interface IRJointBeamGirder : IRJointConnection;
```

#### C#

```
public interface IRJointBeamGirder : IRJointConnection;
```

#### Visual Basic

```
Public Interface IRJointBeamGirder
```

#### Description

Beam-principal beam connection.

#### Version

Available since version 7.5.

### VI.5.1 IRJointBeamGirder Members

The following tables list the members exposed by IRJointBeamGirder.

#### Public Fields

	Name	Description
◆	AngleLeft ( [ see page 2129) ]	
◆	AngleRight ( [ see page 2130) ]	
◆	BeamLeft ( [ see page 2130) ]	Right beam section.
◆	BeamRight ( [ see page 2130) ]	Section of principal beam.
◆	DistGirderFlangeToBeamFlangeLeft ( [ see page 2130) ]	
◆	DistGirderFlangeToBeamFlangeRight ( [ see page 2131) ]	
◆	DistGirderToBeamLeft ( [ see page 2131) ]	
◆	DistGirderToBeamRight ( [ see page 2131) ]	
◆	ElementType ( [ see page 2131) ]	

❖	Girder ( <a href="#">see page 2132</a> )	Section of principal beam.
❖	PlateLeft ( <a href="#">see page 2132</a> )	
❖	PlateRight ( <a href="#">see page 2132</a> )	
❖	SeatLeftDown ( <a href="#">see page 2133</a> )	
❖	SeatLeftUp ( <a href="#">see page 2133</a> )	
❖	SeatRightDown ( <a href="#">see page 2133</a> )	
❖	SeatRightUp ( <a href="#">see page 2133</a> )	
❖	StiffenerLeft ( <a href="#">see page 2134</a> )	
❖	StiffenerRight ( <a href="#">see page 2134</a> )	
❖	Type ( <a href="#">see page 2148</a> )	A connection type.
❖	WFRelPos ( <a href="#">see page 2148</a> )	A type of relative position of a beam and a column.

## Public Methods

	Name	Description
❖	GetFromRobot ( <a href="#">see page 2149</a> )	The function gets connection parameters from Robot and fills an object with the data.
❖	SetToRobot ( <a href="#">see page 2149</a> )	The function sets and saves connection parameter in Robot.

## VI.5.2 IRJointBeamGirder Fields

The fields of the IRJointBeamGirder class are listed here.

### Public Fields

	Name	Description
❖	AngleLeft ( <a href="#">see page 2129</a> )	
❖	AngleRight ( <a href="#">see page 2130</a> )	
❖	BeamLeft ( <a href="#">see page 2130</a> )	Right beam section.
❖	BeamRight ( <a href="#">see page 2130</a> )	Section of principal beam.
❖	DistGirderFlangeToBeamFlangeLeft ( <a href="#">see page 2130</a> )	
❖	DistGirderFlangeToBeamFlangeRight ( <a href="#">see page 2131</a> )	
❖	DistGirderToBeamLeft ( <a href="#">see page 2131</a> )	
❖	DistGirderToBeamRight ( <a href="#">see page 2131</a> )	
❖	ElementType ( <a href="#">see page 2131</a> )	
❖	Girder ( <a href="#">see page 2132</a> )	Section of principal beam.
❖	PlateLeft ( <a href="#">see page 2132</a> )	
❖	PlateRight ( <a href="#">see page 2132</a> )	
❖	SeatLeftDown ( <a href="#">see page 2133</a> )	
❖	SeatLeftUp ( <a href="#">see page 2133</a> )	
❖	SeatRightDown ( <a href="#">see page 2133</a> )	
❖	SeatRightUp ( <a href="#">see page 2133</a> )	
❖	StiffenerLeft ( <a href="#">see page 2134</a> )	
❖	StiffenerRight ( <a href="#">see page 2134</a> )	

### VI.5.2.1 AngleLeft

#### C++

```
HRESULT get_AngleLeft(IRJointBeamGirderAngle**);
```

**C#**

```
public IRJointBeamGirderAngle AngleLeft { get; }
```

**Visual Basic**

```
Public ReadOnly AngleLeft As IRJointBeamGirderAngle
```

**Version**

Available since version 7.5.

### VI.5.2.2 AngleRight

**C++**

```
HRESULT get_AngleRight(IRJointBeamGirderAngle**);
```

**C#**

```
public IRJointBeamGirderAngle AngleRight { get; }
```

**Visual Basic**

```
Public ReadOnly AngleRight As IRJointBeamGirderAngle
```

**Version**

Available since version 7.5.

### VI.5.2.3 BeamLeft

**C++**

```
HRESULT get_BeamLeft(IRJointBeamGirderBeam**);
```

**C#**

```
public IRJointBeamGirderBeam BeamLeft { get; }
```

**Visual Basic**

```
Public ReadOnly BeamLeft As IRJointBeamGirderBeam
```

**Description**

Right beam section.

**Version**

Available since version 7.5.

### VI.5.2.4 BeamRight

**C++**

```
HRESULT get_BeamRight(IRJointBeamGirderBeam**);
```

**C#**

```
public IRJointBeamGirderBeam BeamRight { get; }
```

**Visual Basic**

```
Public ReadOnly BeamRight As IRJointBeamGirderBeam
```

**Description**

Section of principal beam.

**Version**

Available since version 7.5.

### VI.5.2.5 DistGirderFlangeToBeamFlangeLeft

#### C++

```
HRESULT get_DistGirderFlangeToBeamFlangeLeft(double*);  
HRESULT put_DistGirderFlangeToBeamFlangeLeft(double);
```

#### C#

```
public double DistGirderFlangeToBeamFlangeLeft { get; set; }
```

#### Visual Basic

```
Public DistGirderFlangeToBeamFlangeLeft As Double
```

#### Version

Available since version 7.5.

### VI.5.2.6 DistGirderFlangeToBeamFlangeRight

#### C++

```
HRESULT get_DistGirderFlangeToBeamFlangeRight(double*);  
HRESULT put_DistGirderFlangeToBeamFlangeRight(double);
```

#### C#

```
public double DistGirderFlangeToBeamFlangeRight { get; set; }
```

#### Visual Basic

```
Public DistGirderFlangeToBeamFlangeRight As Double
```

#### Version

Available since version 7.5.

### VI.5.2.7 DistGirderToBeamLeft

#### C++

```
HRESULT get_DistGirderToBeamLeft(double*);  
HRESULT put_DistGirderToBeamLeft(double);
```

#### C#

```
public double DistGirderToBeamLeft { get; set; }
```

#### Visual Basic

```
Public DistGirderToBeamLeft As Double
```

#### Version

Available since version 7.5.

### VI.5.2.8 DistGirderToBeamRight

#### C++

```
HRESULT get_DistGirderToBeamRight(double*);  
HRESULT put_DistGirderToBeamRight(double);
```

#### C#

```
public double DistGirderToBeamRight { get; set; }
```

#### Visual Basic

```
Public DistGirderToBeamRight As Double
```

**Version**

Available since version 7.5.

**VI.5.2.9 ElementType****C++**

```
HRESULT get_ElementType(IRJointElementType* );
HRESULT put_ElementType(IRJointElementType);
```

**C#**

```
public IRJointElementType ElementType { get; set; }
```

**Visual Basic**

```
Public ElementType As IRJointElementType
```

**Version**

Available since version 7.5.

**VI.5.2.10 Girder****C++**

```
HRESULT get_Girder(IRJointProfile** );
```

**C#**

```
public IRJointProfile Girder { get; }
```

**Visual Basic**

```
Public ReadOnly Girder As IRJointProfile
```

**Description**

Section of principal beam.

**Version**

Available since version 7.5.

**VI.5.2.11 PlateLeft****C++**

```
HRESULT get_PlateLeft(IRJointBeamGirderPlate** );
```

**C#**

```
public IRJointBeamGirderPlate PlateLeft { get; }
```

**Visual Basic**

```
Public ReadOnly PlateLeft As IRJointBeamGirderPlate
```

**Version**

Available since version 7.5.

**VI.5.2.12 PlateRight****C++**

```
HRESULT get_PlateRight(IRJointBeamGirderPlate** );
```

**C#**

```
public IRJointBeamGirderPlate PlateRight { get; }
```

**Visual Basic**

```
Public ReadOnly PlateRight As IRJointBeamGirderPlate
```

**Version**

Available since version 7.5.

**VI.5.2.13 SeatLeftDown****C++**

```
HRESULT get_SeatLeftDown(IRJointBeamGirderSeat**);
```

**C#**

```
public IRJointBeamGirderSeat SeatLeftDown { get; }
```

**Visual Basic**

```
Public ReadOnly SeatLeftDown As IRJointBeamGirderSeat
```

**Version**

Available since version 7.5.

**VI.5.2.14 SeatLeftUp****C++**

```
HRESULT get_SeatLeftUp(IRJointBeamGirderSeat**);
```

**C#**

```
public IRJointBeamGirderSeat SeatLeftUp { get; }
```

**Visual Basic**

```
Public ReadOnly SeatLeftUp As IRJointBeamGirderSeat
```

**Version**

Available since version 7.5.

**VI.5.2.15 SeatRightDown****C++**

```
HRESULT get_SeatRightDown(IRJointBeamGirderSeat**);
```

**C#**

```
public IRJointBeamGirderSeat SeatRightDown { get; }
```

**Visual Basic**

```
Public ReadOnly SeatRightDown As IRJointBeamGirderSeat
```

**Version**

Available since version 7.5.

**VI.5.2.16 SeatRightUp****C++**

```
HRESULT get_SeatRightUp(IRJointBeamGirderSeat**);
```

**C#**

```
public IRJointBeamGirderSeat SeatRightUp { get; }
```

**Visual Basic**

```
Public ReadOnly SeatRightUp As IRJointBeamGirderSeat
```

**Version**

Available since version 7.5.

**VI.5.2.17 StiffenerLeft****C++**

```
HRESULT get_StiffenerLeft(IRJointBeamGirderStiffener**);
```

**C#**

```
public IRJointBeamGirderStiffener StiffenerLeft { get; }
```

**Visual Basic**

```
Public ReadOnly StiffenerLeft As IRJointBeamGirderStiffener
```

**Version**

Available since version 7.5.

**VI.5.2.18 StiffenerRight****C++**

```
HRESULT get_StiffenerRight(IRJointBeamGirderStiffener**);
```

**C#**

```
public IRJointBeamGirderStiffener StiffenerRight { get; }
```

**Visual Basic**

```
Public ReadOnly StiffenerRight As IRJointBeamGirderStiffener
```

**Version**

Available since version 7.5.

**VI.6 IRJointBeamGirderLoad****Class Hierarchy****C++**

```
interface IRJointBeamGirderLoad : IRJointLoad;
```

**C#**

```
public interface IRJointBeamGirderLoad : IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderLoad
```

**Version**

Available since version 7.5.

**VI.6.1 IRJointBeamGirderLoad Members**

The following tables list the members exposed by IRJointBeamGirderLoad.

## Public Fields

	Name	Description
❖	Cases ( <a href="#">see page 2166</a> )	Selection of load cases, defined for the structure. .
❖	LFx ( <a href="#">see page 2135</a> )	Axial force (left side).
❖	LFz ( <a href="#">see page 2135</a> )	Shear force (left side).
❖	LMy ( <a href="#">see page 2136</a> )	Bending moment (left side).
❖	RFx ( <a href="#">see page 2136</a> )	Axial force (right side).
❖	RFz ( <a href="#">see page 2136</a> )	Shear force (right side).
❖	RMy ( <a href="#">see page 2137</a> )	Bending moment (right side).
❖	Type ( <a href="#">see page 2166</a> )	Load type (defined manually or taken from Robot).

## VI.6.2 IRJointBeamGirderLoad Fields

The fields of the IRJointBeamGirderLoad class are listed here.

## Public Fields

	Name	Description
❖	LFx ( <a href="#">see page 2135</a> )	Axial force (left side).
❖	LFz ( <a href="#">see page 2135</a> )	Shear force (left side).
❖	LMy ( <a href="#">see page 2136</a> )	Bending moment (left side).
❖	RFx ( <a href="#">see page 2136</a> )	Axial force (right side).
❖	RFz ( <a href="#">see page 2136</a> )	Shear force (right side).
❖	RMy ( <a href="#">see page 2137</a> )	Bending moment (right side).

### VI.6.2.1 LFx

#### C++

```
HRESULT get_LFx( double* );
HRESULT put_LFx( double );
```

#### C#

```
public double LFx { get; set; }
```

#### Visual Basic

```
Public LFx As Double
```

#### Description

Axial force (left side).

#### Version

Available since version 7.5.

### VI.6.2.2 LFz

#### C++

```
HRESULT get_LFz( double* );
HRESULT put_LFz( double );
```

#### C#

```
public double LFz { get; set; }
```

#### Visual Basic

```
Public LFz As Double
```

**Description**

Shear force (left side).

**Version**

Available since version 7.5.

**VI.6.2.3 LMy****C++**

```
HRESULT get_LMy( double* );
HRESULT put_LMy( double );
```

**C#**

```
public double LMy { get; set; }
```

**Visual Basic**

```
Public LMy As Double
```

**Description**

Bending moment (left side).

**Version**

Available since version 7.5.

**VI.6.2.4 RFx****C++**

```
HRESULT get_RFx( double* );
HRESULT put_RFx( double );
```

**C#**

```
public double RFx { get; set; }
```

**Visual Basic**

```
Public RFx As Double
```

**Description**

Axial force (right side).

**Version**

Available since version 7.5.

**VI.6.2.5 RFz****C++**

```
HRESULT get_RFz( double* );
HRESULT put_RFz( double );
```

**C#**

```
public double RFz { get; set; }
```

**Visual Basic**

```
Public RFz As Double
```

**Description**

Shear force (right side).

**Version**

Available since version 7.5.

**VI.6.2.6 RMy****C++**

```
HRESULT get_RMy( double* );
HRESULT put_RMy( double );
```

**C#**

```
public double RMy { get; set; }
```

**Visual Basic**

```
Public RMy As Double
```

**Description**

Bending moment (right side).

**Version**

Available since version 7.5.

**VI.7 IRJointBeamGirderSeatBolts****Class Hierarchy****C++**

```
interface IRJointBeamGirderSeatBolts : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderSeatBolts;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderSeatBolts
```

**Version**

Available since version 7.5.

**VI.7.1 IRJointBeamGirderSeatBolts Members**

The following tables list the members exposed by IRJointBeamGirderSeatBolts.

**Public Fields**

	Name	Description
◆	Area ( <a href="#">see page 2138</a> )	
◆	ClassName ( <a href="#">see page 2138</a> )	
◆	Cols ( <a href="#">see page 2138</a> )	
◆	Diameter ( <a href="#">see page 2139</a> )	
◆	DiameterName ( <a href="#">see page 2139</a> )	
◆	DistFromPerpendicularArmEdge ( <a href="#">see page 2139</a> )	Distance of bolt row to edge of perpendicular leg of seat profile.
◆	DistFromVertEdge ( <a href="#">see page 2139</a> )	Distance of extreme bolt to vertical edge of seat profile.
◆	Friction ( <a href="#">see page 2140</a> )	

## VI.7.2 IRJointBeamGirderSeatBolts Fields

The fields of the IRJointBeamGirderSeatBolts class are listed here.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 2138</a> )	
◆	ClassName ( <a href="#">see page 2138</a> )	
◆	Cols ( <a href="#">see page 2138</a> )	
◆	Diameter ( <a href="#">see page 2139</a> )	
◆	DiameterName ( <a href="#">see page 2139</a> )	
◆	DistFromPerpendicularArmEdge ( <a href="#">see page 2139</a> )	Distance of bolt row to edge of perpendicular leg of seat profile.
◆	DistFromVertEdge ( <a href="#">see page 2139</a> )	Distance of extreme bolt to vertical edge of seat profile.
◆	Friction ( <a href="#">see page 2140</a> )	

### VI.7.2.1 Area

#### C++

```
HRESULT get_Area( double* );
```

#### C#

```
public double Area { get; }
```

#### Visual Basic

```
Public ReadOnly Area As Double
```

#### Version

Available since version 7.5.

### VI.7.2.2 ClassName

#### C++

```
HRESULT get_ClassName(BSTR* );
HRESULT put_ClassName(BSTR);
```

#### C#

```
public String ClassName { get; set; }
```

#### Visual Basic

```
Public ClassName As String
```

#### Version

Available since version 7.5.

### VI.7.2.3 Cols

#### C++

```
HRESULT get_Cols( long* );
HRESULT put_Cols( long );
```

#### C#

```
public long Cols { get; set; }
```

**Visual Basic**

```
Public Cols As long
```

**Version**

Available since version 7.5.

**VI.7.2.4 Diameter****C++**

```
HRESULT get_Diameter( long* );
```

**C#**

```
public long Diameter { get; }
```

**Visual Basic**

```
Public ReadOnly Diameter As long
```

**Version**

Available since version 7.5.

**VI.7.2.5 DiameterName****C++**

```
HRESULT get_DiameterName(BSTR* );
HRESULT put_DiameterName(BSTR);
```

**C#**

```
public String DiameterName { get; set; }
```

**Visual Basic**

```
Public DiameterName As String
```

**Version**

Available since version 7.5.

**VI.7.2.6 DistFromPerpendicularArmEdge****C++**

```
HRESULT get_DistFromPerpendicularArmEdge( double* );
HRESULT put_DistFromPerpendicularArmEdge( double );
```

**C#**

```
public double DistFromPerpendicularArmEdge { get; set; }
```

**Visual Basic**

```
Public DistFromPerpendicularArmEdge As double
```

**Description**

Distance of bolt row to edge of perpendicular leg of seat profile.

**Version**

Available since version 7.5.

**VI.7.2.7 DistFromVertEdge****C++**

```
HRESULT get_DistFromVertEdge( double* );
```

```

HRESULT put_DistFromVertEdge( double );

```

**C#**

```

public double DistFromVertEdge { get; set; }

```

**Visual Basic**

```

Public DistFromVertEdge As Double

```

**Description**

Distance of extreme bolt to vertical edge of seat profile.

**Version**

Available since version 7.5.

**VI.7.2.8 Friction**

**C++**

```

HRESULT get_Friction( double* );
HRESULT put_Friction( double );

```

**C#**

```

public double Friction { get; set; }

```

**Visual Basic**

```

Public Friction As Double

```

**Version**

Available since version 7.5.

**VI.8 IRJointConnectorsType**

**C++**

```

enum IRJointConnectorsType;

```

**C#**

```

public enum IRJointConnectorsType;

```

**Visual Basic**

```

Public Enum IRJointConnectorsType

```

**Members**

Members	Description
I_JCT_BOLTS = 0	Available since version 7.5.
I_JCT_WELDS = 1	Available since version 7.5.

**Version**

Available since version 7.5.

**VI.9 IRJointElementType**

**C++**

```

enum IRJointElementType;

```

---

(c) 2020 Autodesk Inc. All rights reserved

**C#**

```
public enum IRJointElementType;
```

**Visual Basic**

```
Public Enum IRJointElementType
```

**Members**

Members	Description
I_JET_RANGLES_WEB = 0	Available since version 7.5.
I_JET_RANGLES_FLANGE = 1	Available since version 7.5.
I_JET_R_STIFF_SHORT = 2	Available since version 7.5.
I_JET_R_STIFF_LONG = 3	Available since version 7.5.
I_JET_R_SEATS = 4	Available since version 7.5.
I_JET_R_SEATSANGLES = 5	Available since version 7.5.
I_JET_R_PLATES = 6	Available since version 7.5.
I_JET_LANGLES_RANGLES_WEB = 7	Available since version 7.5.
I_JET_LANGLES_RANGLES_FLANGE = 8	Available since version 7.5.
I_JET_LSTIFF_SHORT_RSTIFF_SHORT = 9	Available since version 7.5.
I_JET_LSTIFF_SHORT_RSTIFF_LONG = 10	Available since version 7.5.
I_JET_LSTIFF_LONG_RSTIFF_SHORT = 11	Available since version 7.5.
I_JET_LSTIFF_LONG_RSTIFF_LONG = 12	Available since version 7.5.
I_JET_LSEATS_RSEATS = 13	Available since version 7.5.
I_JET_LSEATS_RSEATSANGLES = 14	Available since version 7.5.
I_JET_LSEATSANGLES_RSEATS = 15	Available since version 7.5.
I_JET_LSEATSANGLES_RSEATSANGLES = 16	Available since version 7.5.
I_JET_LPLATES_RPLATES = 17	Available since version 7.5.
I_JET_LPLATES_RANGLES = 18	Available since version 7.5.
I_JET_LANGLES_RPLATES = 19	Available since version 7.5.

**Version**

Available since version 7.5.

**VI.10 IRJointBeamGirderAngle****Class Hierarchy****C++**

```
interface IRJointBeamGirderAngle : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderAngle;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderAngle
```

**Version**

Available since version 7.5.

**VI.10.1 IRJointBeamGirderAngle Members**

The following tables list the members exposed by IRJointBeamGirderAngle.

## Public Fields

	Name	Description
◆	BoltsBeam (see page 2142)	
◆	BoltsGirder (see page 2142)	
◆	Element (see page 2142)	

## VI.10.2 IRJointBeamGirderAngle Fields

The fields of the IRJointBeamGirderAngle class are listed here.

## Public Fields

	Name	Description
◆	BoltsBeam (see page 2142)	
◆	BoltsGirder (see page 2142)	
◆	Element (see page 2142)	

### VI.10.2.1 BoltsBeam

#### C++

```
HRESULT get_BoltsBeam(IRJointAngleBolts**);
```

#### C#

```
public IRJointAngleBolts BoltsBeam { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsBeam As IRJointAngleBolts
```

#### Version

Available since version 7.5.

### VI.10.2.2 BoltsGirder

#### C++

```
HRESULT get_BoltsGirder(IRJointAngleBolts**);
```

#### C#

```
public IRJointAngleBolts BoltsGirder { get; }
```

#### Visual Basic

```
Public ReadOnly BoltsGirder As IRJointAngleBolts
```

#### Version

Available since version 7.5.

### VI.10.2.3 Element

#### C++

```
HRESULT get_Element(IRJointAngle**);
```

#### C#

```
public IRJointAngle Element { get; }
```

#### Visual Basic

```
Public ReadOnly Element As IRJointAngle
```

**Version**

Available since version 7.5.

## VI.11 IRJointBeamGirderBeam

**Class Hierarchy****C++**

```
interface IRJointBeamGirderBeam : IDispatch;
```

**C#**

```
public interface IRJointBeamGirderBeam;
```

**Visual Basic**

```
Public Interface IRJointBeamGirderBeam
```

**Version**

Available since version 7.5.

### VI.11.1 IRJointBeamGirderBeam Members

The following tables list the members exposed by IRJointBeamGirderBeam.

**Public Fields**

	Name	Description
◆	CutEnd ( <a href="#">see page 2143</a> )	
◆	Profile ( <a href="#">see page 2144</a> )	

### VI.11.2 IRJointBeamGirderBeam Fields

The fields of the IRJointBeamGirderBeam class are listed here.

**Public Fields**

	Name	Description
◆	CutEnd ( <a href="#">see page 2143</a> )	
◆	Profile ( <a href="#">see page 2144</a> )	

#### VI.11.2.1 CutEnd

**C++**

```
HRESULT get_CutEnd( IRJointBeamCut** );
```

**C#**

```
public IRJointBeamCut CutEnd { get; }
```

**Visual Basic**

```
Public ReadOnly CutEnd As IRJointBeamCut
```

**Version**

Available since version 7.5.

#### VI.11.2.2 Profile

**C++**

```
HRESULT get_Profile( IRJointProfile** );
```

**C#**

```
public IRJointProfile Profile { get; }
```

**Visual Basic**

```
Public ReadOnly Profile As IRJointProfile
```

**Version**

Available since version 7.5.

## VII IRJointConnectionDefType

**C++**

```
enum IRJointConnectionDefType;
```

**C#**

```
public enum IRJointConnectionDefType;
```

**Visual Basic**

```
Public Enum IRJointConnectionDefType
```

**Members**

Members	Description
I_JCDT_IN_STRUCTURE = 0	A connection defined in a structure.
I_JCDT_STANDALONE = 1	A connection defined independently on a structure.

**Description**

A method of connection definition.

## VIII IRJointConnectionType

**C++**

```
enum IRJointConnectionType;
```

**C#**

```
public enum IRJointConnectionType;
```

**Visual Basic**

```
Public Enum IRJointConnectionType
```

**Members**

Members	Description
I_JCT_KNEE_BOLTED = 0	A bolted knee connection.
I_JCT_KNEE_WELDED = 1	A welded knee connection .
I_JCT_ANGLE = 2	Angle connection.
I_JCT_TRNS = 3	Transversal connection of beams (not available).
I_JCT_COL_PINNED = 5	A pinned column base.
I_JCT_COL_FIXED = 6	A fixed column base.
I_JCT_COL_CONCR = 7	A concrete column base.
I_JCT_TUBE = 8	Tube connection.
I_JCT_GUSSET_SIMPLE = 9	Available since version 1.7.

I_JCT_GUSSET_CROSS = 10	Available since version 1.7.
I_JCT_GUSSET_FLANGE = 11	Available since version 1.7.
I_JCT_BEAM_GIRDER = 12	Available since version 7.5.

**Description**

Types of connections available in a connections module of Robot.

## IX IRJointConnectionInfo

**Class Hierarchy****C++**

```
interface IRJointConnectionInfo : IDispatch;
```

**C#**

```
public interface IRJointConnectionInfo;
```

**Visual Basic**

```
Public Interface IRJointConnectionInfo
```

**Description**

Information about connection. .

### IX.1 IRJointConnectionInfo Members

The following tables list the members exposed by IRJointConnectionInfo.

**Public Fields**

	Name	Description
❖	DefType ( <a href="#">see page 2146</a> )	A method of connection definition.
❖	Elements ( <a href="#">see page 2146</a> )	A selection of elements creating a connection.
❖	Node ( <a href="#">see page 2146</a> )	Number ( <a href="#">see page 2146</a> ) of a connection node.
❖	Number ( <a href="#">see page 2146</a> )	Connection number in the Robot connection server.
❖	Type ( <a href="#">see page 2147</a> )	Connection type.
❖	Uniqueld ( <a href="#">see page 2147</a> )	Unique connection identifier Available since version 1.7.

### IX.2 IRJointConnectionInfo Fields

The fields of the IRJointConnectionInfo class are listed here.

**Public Fields**

	Name	Description
❖	DefType ( <a href="#">see page 2146</a> )	A method of connection definition.
❖	Elements ( <a href="#">see page 2146</a> )	A selection of elements creating a connection.
❖	Node ( <a href="#">see page 2146</a> )	Number ( <a href="#">see page 2146</a> ) of a connection node.
❖	Number ( <a href="#">see page 2146</a> )	Connection number in the Robot connection server.
❖	Type ( <a href="#">see page 2147</a> )	Connection type.
❖	Uniqueld ( <a href="#">see page 2147</a> )	Unique connection identifier Available since version 1.7.

## IX.2.1 DefType

### C++

```
HRESULT get_DefType(IRJointConnectionDefType* );
HRESULT put_DefType(IRJointConnectionDefType);
```

### C#

```
public IRJointConnectionDefType DefType { get; set; }
```

### Visual Basic

```
Public DefType As IRJointConnectionDefType
```

### Description

A method of connection definition.

## IX.2.2 Elements

### C++

```
HRESULT get_Elements(IRobotSelection** );
```

### C#

```
public IRobotSelection Elements { get; }
```

### Visual Basic

```
Public ReadOnly Elements As IRobotSelection
```

### Description

A selection of elements creating a connection.

## IX.2.3 Node

### C++

```
HRESULT get_Node( long* );
HRESULT put_Node( long );
```

### C#

```
public long Node { get; set; }
```

### Visual Basic

```
Public Node As long
```

### Description

Number (see page 2146) of a connection node.

## IX.2.4 Number

### C++

```
HRESULT get_Number( long* );
HRESULT put_Number( long );
```

### C#

```
public long Number { get; set; }
```

### Visual Basic

```
Public Number As long
```

**Description**

Connection number in the Robot connection server.

**IX.2.5 Type****C++**

```
HRESULT get_Type(IRJointConnectionType* );
HRESULT put_Type(IRJointConnectionType);
```

**C#**

```
public IRJointConnectionType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointConnectionType
```

**Description**

Connection type.

**IX.2.6 UniqueId****C++**

```
HRESULT get_UniqueId(long* );
```

**C#**

```
public long UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As long
```

**Description**

Unique connection identifier Available since version 1.7.

**X IRJointWebFlangeRelativePos****C++**

```
enum IRJointWebFlangeRelativePos;
```

**C#**

```
public enum IRJointWebFlangeRelativePos;
```

**Visual Basic**

```
Public Enum IRJointWebFlangeRelativePos
```

**Description**

Possible types of a beam - column mutual position. .

**XI IRJointConnection****Class Hierarchy****C++**

```
interface IRJointConnection : IDispatch;
```

**C#**

```
public interface IRJointConnection;
```

**Visual Basic**

```
Public Interface IRJointConnection
```

**Description**

A data type describing common features for all connections that can be defined in Robot.

## XI.1 IRJointConnection Members

The following tables list the members exposed by IRJointConnection.

**Public Fields**

	Name	Description
◆	Type (see page 2148)	A connection type.
◆	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

**Public Methods**

	Name	Description
◆	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
◆	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

## XI.2 IRJointConnection Fields

The fields of the IRJointConnection class are listed here.

**Public Fields**

	Name	Description
◆	Type (see page 2148)	A connection type.
◆	WFRelPos (see page 2148)	A type of relative position of a beam and a column.

### XI.2.1 Type

**C++**

```
HRESULT get_Type(IRJointConnectionType*);
```

**C#**

```
public IRJointConnectionType Type { get; }
```

**Visual Basic**

```
Public ReadOnly Type As IRJointConnectionType
```

**Description**

A connection type.

### XI.2.2 WFRelPos

**C++**

```
HRESULT get_WFRelPos(IRJointWebFlangeRelativePos*);  
HRESULT put_WFRelPos(IRJointWebFlangeRelativePos*);
```

**C#**

```
public IRJointWebFlangeRelativePos WFRelPos { get; set; }
```

**Visual Basic**

```
Public WFRelPos As IRJointWebFlangeRelativePos
```

**Description**

A type of relative position of a beam and a column.

## XI.3 IRJointConnection Methods

The methods of the IRJointConnection class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetFromRobot (see page 2149)	The function gets connection parameters from Robot and fills an object with the data.
💡	SetToRobot (see page 2149)	The function sets and saves connection parameter in Robot.

### XI.3.1 GetFromRobot

**C++**

```
HRESULT GetFromRobot(IRJointConnectionInfo* _info);
```

**C#**

```
public void GetFromRobot(IRJointConnectionInfo _info);
```

**Visual Basic**

```
Public Sub GetFromRobot(ByRef _info As IRJointConnectionInfo)
```

**Description**

The function gets connection parameters from Robot and fills an object with the data.

### XI.3.2 SetToRobot

**C++**

```
HRESULT SetToRobot(IRJointConnectionInfo* _info);
```

**C#**

```
public void SetToRobot(IRJointConnectionInfo _info);
```

**Visual Basic**

```
Public Sub SetToRobot(ByRef _info As IRJointConnectionInfo)
```

**Description**

The function sets and saves connection parameter in Robot.

## XII IRJointWeld

**Class Hierarchy****C++**

```
interface IRJointWeld : IDispatch;
```

**C#**

```
public interface IRJointWeld;
```

**Visual Basic**

```
Public Interface IRJointWeld
```

**Description**

Description of a weld.

## XII.1 IRJointWeld Members

The following tables list the members exposed by IRJointWeld.

**Public Fields**

	Name	Description
◆	Strength (see page 2150)	Weld resistance.
◆	Thick (see page 2150)	Weld thickness.

## XII.2 IRJointWeld Fields

The fields of the IRJointWeld class are listed here.

**Public Fields**

	Name	Description
◆	Strength (see page 2150)	Weld resistance.
◆	Thick (see page 2150)	Weld thickness.

### XII.2.1 Strength

**C++**

```
HRESULT get_Strength(double* );
HRESULT put_Strength(double);
```

**C#**

```
public double Strength { get; set; }
```

**Visual Basic**

```
Public Strength As double
```

**Description**

Weld resistance.

### XII.2.2 Thick

**C++**

```
HRESULT get_Thick(double* );
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As double
```

**Description**

Weld thickness.

## XIII IRJointPlate

**Class Hierarchy****C++**

```
interface IRJointPlate : IDispatch;
```

**C#**

```
public interface IRJointPlate;
```

**Visual Basic**

```
Public Interface IRJointPlate
```

**Description**

Plate description.

### XIII.1 IRJointPlate Members

The following tables list the members exposed by IRJointPlate.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Exist ( <a href="#">see page 2152</a> )	Information about a plate existance .
❖	Length ( <a href="#">see page 2152</a> )	Plate length .
❖	Material ( <a href="#">see page 2152</a> )	Name of the component material.
❖	Thick ( <a href="#">see page 2152</a> )	Plate thickness.
❖	Width ( <a href="#">see page 2153</a> )	Plate width.

**Public Methods**

	<b>Name</b>	<b>Description</b>
❖	AngleExt ( <a href="#">see page 2153</a> )	Angle of bracket inclination, depending on the parameter - in degrees or radians.

### XIII.2 IRJointPlate Fields

The fields of the IRJointPlate class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
❖	Exist ( <a href="#">see page 2152</a> )	Information about a plate existance .
❖	Length ( <a href="#">see page 2152</a> )	Plate length .
❖	Material ( <a href="#">see page 2152</a> )	Name of the component material.
❖	Thick ( <a href="#">see page 2152</a> )	Plate thickness.
❖	Width ( <a href="#">see page 2153</a> )	Plate width.

#### XIII.2.1 Exist

**C++**

```
HRESULT get_Exist(VARIANT_BOOL* );
```

**C#**

```
public bool Exist { get; }
```

**Visual Basic**

```
Public ReadOnly Exist As Boolean
```

**Description**

Information about a plate existance .

### XIII.2.2 Length

**C++**

```
HRESULT get_Length(double*);  
HRESULT put_Length(double);
```

**C#**

```
public double Length { get; set; }
```

**Visual Basic**

```
Public Length As Double
```

**Description**

Plate length .

### XIII.2.3 Material

**C++**

```
HRESULT get_Material(BSTR*);  
HRESULT put_Material(BSTR);
```

**C#**

```
public BSTR Material { get; set; }
```

**Visual Basic**

```
Public Material As BSTR
```

**Description**

Name of the component material.

**Version**

Available since version 9.5.

### XIII.2.4 Thick

**C++**

```
HRESULT get_Thick(double*);  
HRESULT put_Thick(double);
```

**C#**

```
public double Thick { get; set; }
```

**Visual Basic**

```
Public Thick As Double
```

**Description**

Plate thickness.

### XIII.2.5 Width

#### C++

```
HRESULT get_Width(double* );
HRESULT put_Width(double);
```

#### C#

```
public double Width { get; set; }
```

#### Visual Basic

```
Public Width As Double
```

#### Description

Plate width.

## XIII.3 IRJointPlate Methods

The methods of the IRJointPlate class are listed here.

#### Public Methods

	Name	Description
✳	AngleExt (see page 2153)	Angle of bracket inclination, depending on the parameter - in degrees or radians.

### XIII.3.1 AngleExt

#### C++

```
HRESULT AngleExt(double* ret);
```

#### C#

```
public double AngleExt();
```

#### Visual Basic

```
Public Function AngleExt() As Double
```

#### Description

Angle of bracket inclination, depending on the parameter - in degrees or radians.

#### Version

Available since version 9.5.

## XIV IRJointBolts

#### Class Hierarchy

#### C++

```
interface IRJointBolts : IDispatch;
```

#### C#

```
public interface IRJointBolts;
```

#### Visual Basic

```
Public Interface IRJointBolts
```

## Description

Definition of a group of bolts with common parameters.

## XIV.1 IRJointBolts Members

The following tables list the members exposed by IRJointBolts.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 2154</a> )	Bolt section area.
◆	ClassName ( <a href="#">see page 2155</a> )	Bolt class.
◆	Cols ( <a href="#">see page 2155</a> )	Number of columns.
◆	Diameter ( <a href="#">see page 2155</a> )	Bolt diameter.
◆	DiameterName ( <a href="#">see page 2155</a> )	Bolt diameter name.
◆	Friction ( <a href="#">see page 2156</a> )	Bolt friction.
◆	Height1 ( <a href="#">see page 2156</a> )	Distance of first bolt (e.g. from the top edge of a plate for knee connections) .
◆	Rows ( <a href="#">see page 2156</a> )	Number of rows.

## XIV.2 IRJointBolts Fields

The fields of the IRJointBolts class are listed here.

### Public Fields

	Name	Description
◆	Area ( <a href="#">see page 2154</a> )	Bolt section area.
◆	ClassName ( <a href="#">see page 2155</a> )	Bolt class.
◆	Cols ( <a href="#">see page 2155</a> )	Number of columns.
◆	Diameter ( <a href="#">see page 2155</a> )	Bolt diameter.
◆	DiameterName ( <a href="#">see page 2155</a> )	Bolt diameter name.
◆	Friction ( <a href="#">see page 2156</a> )	Bolt friction.
◆	Height1 ( <a href="#">see page 2156</a> )	Distance of first bolt (e.g. from the top edge of a plate for knee connections) .
◆	Rows ( <a href="#">see page 2156</a> )	Number of rows.

### XIV.2.1 Area

#### C++

```
HRESULT get_Area(double*);
```

#### C#

```
public double Area { get; }
```

#### Visual Basic

```
Public ReadOnly Area As Double
```

### Description

Bolt section area.

### XIV.2.2 ClassName

#### C++

```
HRESULT get_ClassName(BSTR*);
```

```
HRESULT put_ClassName(BSTR);
```

**C#**

```
public String ClassName { get; set; }
```

**Visual Basic**

```
Public ClassName As String
```

**Description**

Bolt class.

## XIV.2.3 Cols

**C++**

```
HRESULT get_Cols(int*);  
HRESULT put_Cols(int);
```

**C#**

```
public int Cols { get; set; }
```

**Visual Basic**

```
Public Cols As int
```

**Description**

Number of columns.

## XIV.2.4 Diameter

**C++**

```
HRESULT get_Diameter(double*);
```

**C#**

```
public double Diameter { get; }
```

**Visual Basic**

```
Public ReadOnly Diameter As double
```

**Description**

Bolt diameter.

## XIV.2.5 DiameterName

**C++**

```
HRESULT get_DiameterName(BSTR*);  
HRESULT put_DiameterName(BSTR);
```

**C#**

```
public String DiameterName { get; set; }
```

**Visual Basic**

```
Public DiameterName As String
```

**Description**

Bolt diameter name.

## XIV.2.6 Friction

### C++

```
HRESULT get_Friction(double* );
HRESULT put_Friction(double);
```

### C#

```
public double Friction { get; set; }
```

### Visual Basic

```
Public Friction As Double
```

### Description

Bolt friction.

## XIV.2.7 Height1

### C++

```
HRESULT get_Height1(double* );
HRESULT put_Height1(double);
```

### C#

```
public double Height1 { get; set; }
```

### Visual Basic

```
Public Height1 As Double
```

### Description

Distance of first bolt (e.g. from the top edge of a plate for knee connections).

## XIV.2.8 Rows

### C++

```
HRESULT get_Rows(int* );
HRESULT put_Rows(int);
```

### C#

```
public int Rows { get; set; }
```

### Visual Basic

```
Public Rows As Integer
```

### Description

Number of rows.

## XV IRJointBoltType

### C++

```
enum IRJointBoltType;
```

### C#

```
public enum IRJointBoltType;
```

**Visual Basic**

```
Public Enum IRJointBoltType
```

**Description**

Available bolt types.

## XVI IRJointWebType

**C++**

```
enum IRJointWebType;
```

**C#**

```
public enum IRJointWebType;
```

**Visual Basic**

```
Public Enum IRJointWebType
```

**Description**

Types of reinforcing plates.

## XVII IRJointConnectionServer

**Class Hierarchy****C++**

```
interface IRJointConnectionServer : IDispatch;
```

**C#**

```
public interface IRJointConnectionServer;
```

**Visual Basic**

```
Public Interface IRJointConnectionServer
```

**Description**

The connection server administers all connection defined in the project. Each connection is associated with a number. The numbering of connections may be discontinuous.

### XVII.1 IRJointConnectionServer Members

The following tables list the members exposed by IRJointConnectionServer.

**Public Fields**

	Name	Description
◆	Count ( <a href="#">see page 2158</a> )	Number of defined connections .

**Public Methods**

	Name	Description
◆	Calculate ( <a href="#">see page 2159</a> )	The function carries out verification of the connection with the given number for the defined loads and returns the verification results.
◆	CalculateNote ( <a href="#">see page 2159</a> )	The function carries out calculations for the indicated connection as the function Calculate ( <a href="#">see page 2159</a> ()). Additionally, it generates a calculation note with calculation results.

	Create ( <a href="#">see page 2160</a> )	The function creates and returns an object representing a new connection of the indicated type.
	CreateInfo ( <a href="#">see page 2160</a> )	The function creates and returns a new object representing the information about a connection.
	Delete ( <a href="#">see page 2160</a> )	The function removes the connection with the given number. .
	Exist ( <a href="#">see page 2160</a> )	The function checks if there is a connection with the indicated number. .
	Find ( <a href="#">see page 2161</a> )	The function finds a connection defined for the indicated selection of bars and returns its number. If there is no such connection, it returns the value 0.
	FindWithId ( <a href="#">see page 2161</a> )	Function returns number of a connection with the specified unique identifier. If the connection is not found, zero value is returned. Available since version 1.7.
	Get ( <a href="#">see page 2161</a> )	The function returns the connection with the given number. .
	GetAllNumbers ( <a href="#">see page 2162</a> )	The function returns a table containing numbers of all defined connections.
	GetInfo ( <a href="#">see page 2162</a> )	The function returns an object describing the connection with the given number.

## XVII.2 IRJointConnectionServer Fields

The fields of the IRJointConnectionServer class are listed here.

### Public Fields

	Name	Description
	Count ( <a href="#">see page 2158</a> )	Number of defined connections .

### XVII.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of defined connections .

## XVII.3 IRJointConnectionServer Methods

The methods of the IRJointConnectionServer class are listed here.

### Public Methods

	Name	Description
	Calculate ( <a href="#">see page 2159</a> )	The function carries out verification of the connection with the given number for the defined loads and returns the verification results.
	CalculateNote ( <a href="#">see page 2159</a> )	The function carries out calculations for the indicated connection as the function Calculate ( <a href="#">see page 2159</a> ()). Additionally, it generates a calculation note with calculation results.
	Create ( <a href="#">see page 2160</a> )	The function creates and returns an object representing a new connection of the indicated type.
	CreateInfo ( <a href="#">see page 2160</a> )	The function creates and returns a new object representing the information about a connection.

	Delete ( <a href="#">see page 2160</a> )	The function removes the connection with the given number. .
	Exist ( <a href="#">see page 2160</a> )	The function checks if there is a connection with the indicated number. .
	Find ( <a href="#">see page 2161</a> )	The function finds a connection defined for the indicated selection of bars and returns its number. If there is no such connection, it returns the value 0.
	FindWithId ( <a href="#">see page 2161</a> )	Function returns number of a connection with the specified unique identifier. If the connection is not found, zero value is returned. Available since version 1.7.
	Get ( <a href="#">see page 2161</a> )	The function returns the connection with the given number. .
	GetAllNumbers ( <a href="#">see page 2162</a> )	The function returns a table containing numbers of all defined connections.
	GetInfo ( <a href="#">see page 2162</a> )	The function returns an object describing the connection with the given number.

### XVII.3.1 Calculate

**C++**

```
HRESULT Calculate(long _num, IRJointLoad* _joint_loads, double* ret);
```

**C#**

```
public double Calculate(long _num, IRJointLoad _joint_loads);
```

**Visual Basic**

```
Public Function Calculate(_num As long, ByRef _joint_loads As IRJointLoad) As double
```

**Description**

The function carries out verification of the connection with the given number for the defined loads and returns the verification results.

### XVII.3.2 CalculateNote

**C++**

```
HRESULT CalculateNote(long _num, IRJointLoad* _joint_loads, BSTR _file_path, double* ret);
```

**C#**

```
public double CalculateNote(long _num, IRJointLoad _joint_loads, String _file_path);
```

**Visual Basic**

```
Public Function CalculateNote(_num As long, ByRef _joint_loads As IRJointLoad, _file_path As String) As double
```

**Description**

The function carries out calculations for the indicated connection as the function Calculate ([see page 2159](#)()). Additionally, it generates a calculation note with calculation results.

### XVII.3.3 Create

**C++**

```
HRESULT Create(IRJointConnectionType _joint_type, IRJointConnection** ret);
```

**C#**

```
public IRJointConnection Create(IRJointConnectionType _joint_type);
```

**Visual Basic**

```
Public Function Create(_joint_type As IRJointConnectionType) As IRJointConnection
```

**Description**

The function creates and returns an object representing a new connection of the indicated type.

### XVII.3.4 CreateInfo

**C++**

```
HRESULT CreateInfo(IRJointConnectionInfo** ret);
```

**C#**

```
public IRJointConnectionInfo CreateInfo();
```

**Visual Basic**

```
Public Function CreateInfo() As IRJointConnectionInfo
```

**Description**

The function creates and returns a new object representing the information about a connection.

### XVII.3.5 Delete

**C++**

```
HRESULT Delete(long _num);
```

**C#**

```
public void Delete(long _num);
```

**Visual Basic**

```
Public Sub Delete(_num As long)
```

**Description**

The function removes the connection with the given number. .

### XVII.3.6 Exist

**C++**

```
HRESULT Exist(long _num, VARIANT_BOOL* ret);
```

**C#**

```
public bool Exist(long _num);
```

**Visual Basic**

```
Public Function Exist(_num As long) As Boolean
```

**Description**

The function checks if there is a connection with the indicated number. .

### XVII.3.7 Find

**C++**

```
HRESULT Find(BSTR _sel_text, long* ret);
```

**C#**

```
public long Find(String _sel_text);
```

**Visual Basic**

```
Public Function Find(_sel_text As String) As long
```

**Description**

The function finds a connection defined for the indicated selection of bars and returns its number. If there is no such connection, it returns the value 0.

**XVII.3.8 FindWithId****C++**

```
HRESULT FindWithId(long _unique_id, long* ret);
```

**C#**

```
public long FindWithId(long _unique_id);
```

**Visual Basic**

```
Public Function FindWithId(_unique_id As long) As long
```

**Description**

Function returns number of a connection with the specified unique identifier. If the connection is not found, zero value is returned. Available since version 1.7.

**XVII.3.9 Get****C++**

```
HRESULT Get(long _num, IRJointConnection** ret);
```

**C#**

```
public IRJointConnection Get(long _num);
```

**Visual Basic**

```
Public Function Get(_num As long) As IRJointConnection
```

**Description**

The function returns the connection with the given number. .

**XVII.3.10 GetAllNumbers****C++**

```
HRESULT GetAllNumbers(IRobotNumbersArray** ret);
```

**C#**

```
public IRobotNumbersArray GetAllNumbers();
```

**Visual Basic**

```
Public Function GetAllNumbers() As IRobotNumbersArray
```

**Description**

The function returns a table containing numbers of all defined connections.

### XVII.3.11 GetInfo

**C++**

```
HRESULT GetInfo(long _num, IRJointConnectionInfo** ret);
```

**C#**

```
public IRJointConnectionInfo GetInfo(long _num);
```

**Visual Basic**

```
Public Function GetInfo(_num As long) As IRJointConnectionInfo
```

**Description**

The function returns an object describing the connection with the given number.

## XVIII IRJointProfile

**Class Hierarchy**

**C++**

```
interface IRJointProfile : IDispatch;
```

**C#**

```
public interface IRJointProfile;
```

**Visual Basic**

```
Public Interface IRJointProfile
```

**Description**

Structure describing a single connection component. .

### XVIII.1 IRJointProfile Members

The following tables list the members exposed by IRJointProfile.

**Public Fields**

	Name	Description
◆	Angle (↗ see page 2163)	Connection angle.
◆	BarNumber (↗ see page 2163)	Bar number.
◆	Material (↗ see page 2164)	Material name .
◆	Section (↗ see page 2164)	Section name .

**Public Methods**

	Name	Description
☞	AngleExt (↗ see page 2164)	Angle (↗ see page 2163) of section inclination, depending on the parameter - in degrees or radians.
☞	Edit (↗ see page 2165)	Function gets a section type through the user interface.
☞	EditComponent (↗ see page 2165)	Function takes component type through user interface.

### XVIII.2 IRJointProfile Fields

The fields of the IRJointProfile class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Angle (see page 2163)	Connection angle.
◆	BarNumber (see page 2163)	Bar number.
◆	Material (see page 2164)	Material name .
◆	Section (see page 2164)	Section name .

**XVIII.2.1 Angle****C++**

```
HRESULT get_Angle(double* );
HRESULT put_Angle(double);
```

**C#**

```
public double Angle { get; set; }
```

**Visual Basic**

```
Public Angle As Double
```

**Description**

Connection angle.

**XVIII.2.2 BarNumber****C++**

```
HRESULT get_BarNumber(long* );
HRESULT put_BarNumber(long);
```

**C#**

```
public long BarNumber { get; set; }
```

**Visual Basic**

```
Public BarNumber As Long
```

**Description**

Bar number.

**Version**

Available since version 3.5.

**XVIII.2.3 Material****C++**

```
HRESULT get_Material(BSTR* );
HRESULT put_Material(BSTR);
```

**C#**

```
public String Material { get; set; }
```

**Visual Basic**

```
Public Material As String
```

**Description**

Material name .

## XVIII.2.4 Section

### C++

```
HRESULT get_Section(BSTR* );
HRESULT put_Section(BSTR);
```

### C#

```
public String Section { get; set; }
```

### Visual Basic

```
Public Section As String
```

### Description

Section name .

## XVIII.3 IRJointProfile Methods

The methods of the IRJointProfile class are listed here.

### Public Methods

	Name	Description
💡	AngleExt (see page 2164)	Angle (see page 2163) of section inclination, depending on the parameter - in degrees or radians.
💡	Edit (see page 2165)	Function gets a section type through the user interface.
💡	EditComponent (see page 2165)	Function takes component type through user interface.

## XVIII.3.1 AngleExt

### C++

```
HRESULT AngleExt(IRJointAngleUnit _unit, double* ret);
```

### C#

```
public double AngleExt(IRJointAngleUnit _unit);
```

### Visual Basic

```
Public Function AngleExt(_unit As IRJointAngleUnit) As double
```

### Description

Angle (see page 2163) of section inclination, depending on the parameter - in degrees or radians.

### Version

Available since version 9.5.

## XVIII.3.2 Edit

### C++

```
HRESULT Edit(IRJointConnectionType _cType, void* ret);
```

### C#

```
public void Edit(IRJointConnectionType _cType);
```

### Visual Basic

```
Public Function Edit(_cType As IRJointConnectionType) As void
```

**Description**

Function gets a section type through the user interface.

**Version**

Available since version 9.5.

**XVIII.3.3 EditComponent****C++**

```
HRESULT EditComponent( IRJointComponentType _cType, void* ret );
```

**C#**

```
public void EditComponent( IRJointComponentType _cType );
```

**Visual Basic**

```
Public Function EditComponent(_cType As IRJointComponentType) As void
```

**Description**

Function takes component type through user interface.

**Version**

Available since version 10.

**XIX IRJointLoad****Class Hierarchy****C++**

```
interface IRJointLoad : IDispatch;
```

**C#**

```
public interface IRJointLoad;
```

**Visual Basic**

```
Public Interface IRJointLoad
```

**Description**

Interface describing a load applied to the connection. .

**XIX.1 IRJointLoad Members**

The following tables list the members exposed by IRJointLoad.

**Public Fields**

	Name	Description
❖	Cases ( <a href="#">see page 2166</a> )	Selection of load cases, defined for the structure. .
❖	Type ( <a href="#">see page 2166</a> )	Load type (defined manually or taken from Robot).

**XIX.2 IRJointLoad Fields**

The fields of the IRJointLoad class are listed here.

**Public Fields**

	Name	Description
❖	Cases (see page 2166)	Selection of load cases, defined for the structure..
❖	Type (see page 2166)	Load type (defined manually or taken from Robot).

**XIX.2.1 Cases****C++**

```
HRESULT get_Cases(IRobotSelection**);
```

**C#**

```
public IRobotSelection Cases { get; }
```

**Visual Basic**

```
Public ReadOnly Cases As IRobotSelection
```

**Description**

Selection of load cases, defined for the structure..

**XIX.2.2 Type****C++**

```
HRESULT get_Type(IRJointLoadType*);  
HRESULT put_Type(IRJointLoadType);
```

**C#**

```
public IRJointLoadType Type { get; set; }
```

**Visual Basic**

```
Public Type As IRJointLoadType
```

**Description**

Load type (defined manually or taken from Robot).

**XX IRJointLoadType****C++**

```
enum IRJointLoadType;
```

**C#**

```
public enum IRJointLoadType;
```

**Visual Basic**

```
Public Enum IRJointLoadType
```

**Members**

Members	Description
I_JLT_ROBOT_FORCES = 1	Load taken from structure calculations .
I_JLT_MANUAL_FORCES = 2	Loads defined manually for the connection.

## XXI IRJointAngleUnit

### C++

```
enum IRJointAngleUnit;
```

### C#

```
public enum IRJointAngleUnit;
```

### Visual Basic

```
Public Enum IRJointAngleUnit
```

### Members

Members	Description
I_JAU_RADIANS = 0	Available since version 9.5.
I_JAU_DEGREES = 1	Available since version 9.5.

### Version

Available since version 9.5.

## XXII IRJointExtType

### C++

```
enum IRJointExtType;
```

### C#

```
public enum IRJointExtType;
```

### Visual Basic

```
Public Enum IRJointExtType
```

### Members

Members	Description
I_EXT_NONE = 0	Available since version 9.5.
I_EXT_ANGLE2BEAM = 1	Available since version 9.5.

### Version

Available since version 9.5.

## XXIII IRJointComponentType

### C++

```
enum IRJointComponentType;
```

### C#

```
public enum IRJointComponentType;
```

### Visual Basic

```
Public Enum IRJointComponentType
```

## Members

Members	Description
I_JCMT_BEAM_MAIN = 0	Available since version 10.
I_JCMT_BEAM_ANGLES = 1	Available since version 10.
I_JCMT_BEAM_BRACKETS = 2	Available since version 10.

## Version

Available since version 10.

# External parameters

It is possible to define arbitrary parameters and assign them to individual structure elements and to the entire project.

## Version

Available since version 8.2.

## Enumerations

	Name	Description
	IRobotParamValueType (see page 2190)	Available types of external parameter values.

## Interfaces

	Name	Description
	IRobotParamServer (see page 2169)	Interface for access to parameters of structure elements of all types. Structure elements are indicated by means of unique identifiers.
	IRobotParamCollection (see page 2174)	Parameter collection.
	IRobotParamSchemaDef (see page 2177)	Definition of the parameter schema.
	IRobotParamSchemaMngr (see page 2181)	Interface for management of all parameter schemata defined in the structure.
	IRobotParamDef (see page 2185)	Complete definition of an external parameter.
	IRobotParamSchema (see page 2190)	Parameter schemata have been implemented on order to avoid a collision of names of parameters defined by different external applications. It is recommended that each external application should define its own separate parameter schema.

# | IRobotParamServer

## Class Hierarchy

### C++

```
interface IRobotParamServer : IDispatch;
```

### C#

```
public interface IRobotParamServer;
```

## Visual Basic

```
Public Interface IRobotParamServer
```

## Description

Interface for access to parameters of structure elements of all types. Structure elements are indicated by means of unique

identifiers.

## Version

Available since version 8.2.

## I.1 IRobotParamServer Members

The following tables list the members exposed by IRobotParamServer.

### Public Fields

	Name	Description
◆	Schemas (see page 2170)	External parameter schemata.

### Public Methods

	Name	Description
◆	GetAllParams (see page 2171)	Function enters to the collection all parameters assigned to the given structure element. The number of parameters is returned.
◆	GetAllParamsForSchema (see page 2171)	Function enters to the collection all parameters from the specified schema assigned to the specified structure element. The number of parameters is returned.
◆	GetObjectsWithParam (see page 2171)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.
◆	GetObjectsWithParamVal (see page 2172)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.
◆	GetParam (see page 2172)	Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value.
◆	ParamUniqueIdToName (see page 2173)	Function gives access to the name of the parameter and of the schema it belongs to. If the parameter with the specified identifier has not been found, function returns the False value. .
◆	RemoveParam (see page 2173)	Function deletes a determined parameter from the specified structure element.
◆	RemoveSchemaParams (see page 2173)	Function deletes all parameters belonging to the determined schema from the specified structure element.
◆	ResetParam (see page 2174)	Function assigns a default value of the determined parameter to a given structure element.
◆	SetParam (see page 2174)	Function assigns a determined value of the specified parameter to a given structure element.

## I.2 IRobotParamServer Fields

The fields of the IRobotParamServer class are listed here.

### Public Fields

	Name	Description
◆	Schemas (see page 2170)	External parameter schemata.

### I.2.1 Schemas

#### C++

```
HRESULT get_Schemas( IRobotParamSchemaMngr** );
```

#### C#

```
public IRobotParamSchemaMngr Schemas { get; }
```

## Visual Basic

```
Public ReadOnly Schemas As IRobotParamSchemaMngr
```

### Description

External parameter schemata.

### Version

Available since version 8.2.

## I.3 IRobotParamServer Methods

The methods of the IRobotParamServer class are listed here.

### Public Methods

	Name	Description
⊕	GetAllParams (see page 2171)	Function enters to the collection all parameters assigned to the given structure element. The number of parameters is returned.
⊕	GetAllParamsForSchema (see page 2171)	Function enters to the collection all parameters from the specified schema assigned to the specified structure element. The number of parameters is returned.
⊕	GetObjectsWithParam (see page 2171)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.
⊕	GetObjectsWithParamVal (see page 2172)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.
⊕	GetParam (see page 2172)	Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value.
⊕	ParamUniqueIdToName (see page 2173)	Function gives access to the name of the parameter and of the schema it belongs to. If the parameter with the specified identifier has not been found, function returns the False value. .
⊕	RemoveParam (see page 2173)	Function deletes a determined parameter from the specified structure element.
⊕	RemoveSchemaParams (see page 2173)	Function deletes all parameters belonging to the determined schema from the specified structure element.
⊕	ResetParam (see page 2174)	Function assigns a default value of the determined parameter to a given structure element.
⊕	SetParam (see page 2174)	Function assigns a determined value of the specified parameter to a given structure element.

### I.3.1 GetAllParams

#### C++

```
HRESULT GetAllParams(long _unique_obj_id, IRobotParamCollection* _params_col, long* ret);
```

#### C#

```
public long GetAllParams(long _unique_obj_id, IRobotParamCollection _params_col);
```

### Visual Basic

```
Public Function GetAllParams(_unique_obj_id As long, ByRef _params_col As IRobotParamCollection) As long
```

### Description

Function enters to the collection all parameters assigned to the given structure element. The number of parameters is returned.

**Version**

Available since version 8.2.

**I.3.2 GetAllParamsForSchema****C++**

```
HRESULT GetAllParamsForSchema(long _unique_obj_id, BSTR _schema_name,
IRobotParamCollection* _params_col, long* ret);
```

**C#**

```
public long GetAllParamsForSchema(long _unique_obj_id, String _schema_name,
IRobotParamCollection _params_col);
```

**Visual Basic**

```
Public Function GetAllParamsForSchema(_unique_obj_id As long, _schema_name As String, ByRef
_params_col As IRobotParamCollection) As long
```

**Description**

Function enters to the collection all parameters from the specified schema assigned to the specified structure element. The number of parameters is returned.

**Version**

Available since version 8.2.

**I.3.3 GetObjectsWithParam****C++**

```
HRESULT GetObjectsWithParam(BSTR _param_id, IRobotNumbersArray* _obj_ids, long* ret);
```

**C#**

```
public long GetObjectsWithParam(String _param_id, IRobotNumbersArray _obj_ids);
```

**Visual Basic**

```
Public Function GetObjectsWithParam(_param_id As String, ByRef _obj_ids As
IRobotNumbersArray) As long
```

**Description**

Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.

**Version**

Available since version 8.2.

**I.3.4 GetObjectsWithParamVal****C++**

```
HRESULT GetObjectsWithParamVal(BSTR _param_id, BSTR _param_val, IRobotNumbersArray*
_obj_ids, long* ret);
```

**C#**

```
public long GetObjectsWithParamVal(String _param_id, String _param_val, IRobotNumbersArray
_obj_ids);
```

**Visual Basic**

```
Public Function GetObjectsWithParamVal(_param_id As String, _param_val As String, ByRef
_obj_ids As IRobotNumbersArray) As long
```

**Description**

Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.

**Version**

Available since version 8.2.

**I.3.5 GetParam****C++**

```
HRESULT GetParam(long _unique_obj_id, BSTR _param_id, BSTR _ret_param_val, VARIANT_BOOL* ret);
```

**C#**

```
public bool GetParam(long _unique_obj_id, String _param_id, String _ret_param_val);
```

**Visual Basic**

```
Public Function GetParam(_unique_obj_id As long, _param_id As String, ByRef _ret_param_val
As String) As Boolean
```

**Description**

Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value.

**Version**

Available since version 8.2.

**I.3.6 ParamUniqueIdToName****C++**

```
HRESULT ParamUniqueIdToName(BSTR _param_id, BSTR _ret_schema_name, BSTR _ret_param_name,
VARIANT_BOOL* ret);
```

**C#**

```
public bool ParamUniqueIdToName(String _param_id, String _ret_schema_name, String
_ret_param_name);
```

**Visual Basic**

```
Public Function ParamUniqueIdToName(_param_id As String, ByRef _ret_schema_name As String,
ByRef _ret_param_name As String) As Boolean
```

**Description**

Function gives access to the name of the parameter and of the schema it belongs to. If the parameter with the specified identifier has not been found, function returns the False value. .

**Version**

Available since version 8.2.

**I.3.7 RemoveParam****C++**

```
HRESULT RemoveParam(long _unique_obj_id, BSTR _param_id);
```

**C#**

```
public void RemoveParam(long _unique_obj_id, String _param_id);
```

**Visual Basic**

```
Public Sub RemoveParam(_unique_obj_id As long, _param_id As String)
```

**Description**

Function deletes a determined parameter from the specified structure element.

**Version**

Available since version 8.2.

### I.3.8 RemoveSchemaParams

**C++**

```
HRESULT RemoveSchemaParams(long _unique_obj_id, BSTR _schema_name);
```

**C#**

```
public void RemoveSchemaParams(long _unique_obj_id, String _schema_name);
```

**Visual Basic**

```
Public Sub RemoveSchemaParams(_unique_obj_id As long, _schema_name As String)
```

**Description**

Function deletes all parameters belonging to the determined schema from the specified structure element.

**Version**

Available since version 8.2.

### I.3.9 ResetParam

**C++**

```
HRESULT ResetParam(long _unique_obj_id, BSTR _param_id);
```

**C#**

```
public void ResetParam(long _unique_obj_id, String _param_id);
```

**Visual Basic**

```
Public Sub ResetParam(_unique_obj_id As long, _param_id As String)
```

**Description**

Function assigns a default value of the determined parameter to a given structure element.

**Version**

Available since version 8.2.

### I.3.10 SetParam

**C++**

```
HRESULT SetParam(long _unique_obj_id, BSTR _param_id, BSTR _param_val);
```

**C#**

```
public void SetParam(long _unique_obj_id, String _param_id, String _param_val);
```

**Visual Basic**

```
Public Sub SetParam(_unique_obj_id As long, _param_id As String, _param_val As String)
```

**Description**

Function assigns a determined value of the specified parameter to a given structure element.

**Version**

Available since version 8.2.

## II IRobotParamCollection

**Class Hierarchy****C++**

```
interface IRobotParamCollection : IDispatch;
```

**C#**

```
public interface IRobotParamCollection;
```

**Visual Basic**

```
Public Interface IRobotParamCollection
```

**Description**

Parameter collection.

**Version**

Available since version 8.2.

### II.1 IRobotParamCollection Members

The following tables list the members exposed by IRobotParamCollection.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Count ( <a href="#">see page 2175</a> )	Number of parameters in the collection.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Find ( <a href="#">see page 2176</a> )	Function finds in the collection a parameter with the specified name and returns its index.
◆	GetFullName ( <a href="#">see page 2176</a> )	Function gives access to a name of the parameter and a name of the schema it belongs to.
◆	GetId ( <a href="#">see page 2176</a> )	Function gives access to the unique identifier of a parameter with the specified index.
◆	GetName ( <a href="#">see page 2177</a> )	Function gives access to the parameter name.
◆	GetValue ( <a href="#">see page 2177</a> )	Function gives access to a value of the parameter with the specified index.

### II.2 IRobotParamCollection Fields

The fields of the IRobotParamCollection class are listed here.

## Public Fields

	Name	Description
◆	Count (see page 2175)	Number of parameters in the collection.

### II.2.1 Count

#### C++

```
HRESULT get_Count(long* );
```

#### C#

```
public long Count { get; }
```

#### Visual Basic

```
Public ReadOnly Count As long
```

#### Description

Number of parameters in the collection.

#### Version

Available since version 8.2.

## II.3 IRobotParamCollection Methods

The methods of the IRobotParamCollection class are listed here.

#### Public Methods

	Name	Description
◆	Find (see page 2176)	Function finds in the collection a parameter with the specified name and returns its index.
◆	GetFullName (see page 2176)	Function gives access to a name of the parameter and a name of the schema it belongs to.
◆	GetId (see page 2176)	Function gives access to the unique identifier of a parameter with the specified index.
◆	GetName (see page 2177)	Function gives access to the parameter name.
◆	GetValue (see page 2177)	Function gives access to a value of the parameter with the specified index.

### II.3.1 Find

#### C++

```
HRESULT Find(BSTR _param_name, BSTR _schema_name, long* ret);
```

#### C#

```
public long Find(String _param_name, String _schema_name);
```

#### Visual Basic

```
Public Function Find(_param_name As String, _schema_name As String) As long
```

#### Description

Function finds in the collection a parameter with the specified name and returns its index.

#### Version

Available since version 8.2.

## II.3.2 GetFullName

**C++**

```
HRESULT GetFullName(long _idx, BSTR _param_name, BSTR _schema_name);
```

**C#**

```
public void GetFullName(long _idx, String _param_name, String _schema_name);
```

**Visual Basic**

```
Public Sub GetFullName(_idx As long, ByRef _param_name As String, ByRef _schema_name As String)
```

**Description**

Function gives access to a name of the parameter and a name of the schema it belongs to.

**Version**

Available since version 8.2.

## II.3.3 GetId

**C++**

```
HRESULT GetId(long _idx, BSTR* ret);
```

**C#**

```
public String GetId(long _idx);
```

**Visual Basic**

```
Public Function GetId(_idx As long) As String
```

**Description**

Function gives access to the unique identifier of a parameter with the specified index.

**Version**

Available since version 8.2.

## II.3.4 GetName

**C++**

```
HRESULT GetName(long _idx, BSTR* ret);
```

**C#**

```
public String GetName(long _idx);
```

**Visual Basic**

```
Public Function GetName(_idx As long) As String
```

**Description**

Function gives access to the parameter name.

**Version**

Available since version 8.2.

## II.3.5 GetValue

### C++

```
HRESULT GetValue(long _idx, BSTR* ret);
```

### C#

```
public String GetValue(long _idx);
```

### Visual Basic

```
Public Function GetValue(_idx As long) As String
```

### Description

Function gives access to a value of the parameter with the specified index.

### Version

Available since version 8.2.

## III IRobotParamSchemaDef

### Class Hierarchy

### C++

```
interface IRobotParamSchemaDef : IDispatch;
```

### C#

```
public interface IRobotParamSchemaDef;
```

### Visual Basic

```
Public Interface IRobotParamSchemaDef
```

### Description

Definition of the parameter schema.

### Version

Available since version 8.2.

## III.1 IRobotParamSchemaDef Members

The following tables list the members exposed by IRobotParamSchemaDef.

### Public Fields

	Name	Description
❖	Name (see page 2178)	Schema name.
❖	ParamCount (see page 2179)	Number of parameters defined in the schema.

### Public Methods

	Name	Description
❖	AddParam (see page 2179)	Function defines a new parameter. If the parameter with the specified name already exists in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element fails and function returns the False value.

	AddSimpleParam ( <a href="#">see page 2180</a> )	Function adds a simple parameter with the specified name to the schema. If the parameter with the specified name has been earlier defined in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element results in error and function returns the False value.
	GetParam ( <a href="#">see page 2180</a> )	Function returns the parameter definition with the specified index in the schema.
	GetParamName ( <a href="#">see page 2180</a> )	Function returns the parameter name with the specified index.
	ParamNameToUniqueId ( <a href="#">see page 2181</a> )	Function returns a unique identifier of the parameter with the specified name.
	RemoveParam ( <a href="#">see page 2181</a> )	Function deletes the parameter with the specified name from the schema. Simultaneously, the parameter will be removed from all structure elements.

## III.2 IRobotParamSchemaDef Fields

The fields of the IRobotParamSchemaDef class are listed here.

### Public Fields

	Name	Description
	Name ( <a href="#">see page 2178</a> )	Schema name.
	ParamCount ( <a href="#">see page 2179</a> )	Number of parameters defined in the schema.

### III.2.1 Name

#### C++

```
HRESULT get_Name(BSTR*);
```

#### C#

```
public String Name { get; }
```

#### Visual Basic

```
Public ReadOnly Name As String
```

#### Description

Schema name.

#### Version

Available since version 8.2.

### III.2.2 ParamCount

#### C++

```
HRESULT get_ParamCount(long*);
```

#### C#

```
public long ParamCount { get; }
```

#### Visual Basic

```
Public ReadOnly ParamCount As long
```

#### Description

Number of parameters defined in the schema.

#### Version

Available since version 8.2.

### III.3 IRobotParamSchemaDef Methods

The methods of the IRobotParamSchemaDef class are listed here.

#### Public Methods

	Name	Description
💡	AddParam (see page 2179)	Function defines a new parameter. If the parameter with the specified name already exists in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element fails and function returns the False value.
💡	AddSimpleParam (see page 2180)	Function adds a simple parameter with the specified name to the schema. If the parameter with the specified name has been earlier defined in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element results in error and function returns the False value.
💡	GetParam (see page 2180)	Function returns the parameter definition with the specified index in the schema.
💡	GetParamName (see page 2180)	Function returns the parameter name with the specified index.
💡	ParamNameToUniqueld (see page 2181)	Function returns a unique identifier of the parameter with the specified name.
💡	RemoveParam (see page 2181)	Function deletes the parameter with the specified name from the schema. Simultaneously, the parameter will be removed from all structure elements.

#### III.3.1 AddParam

##### C++

```
HRESULT AddParam(IRobotParamDef* _param_def, VARIANT_BOOL* ret);
```

##### C#

```
public bool AddParam(IRobotParamDef _param_def);
```

##### Visual Basic

```
Public Function AddParam(ByRef _param_def As IRobotParamDef) As Boolean
```

##### Description

Function defines a new parameter. If the parameter with the specified name already exists in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element fails and function returns the False value.

##### Version

Available since version 8.2.

#### III.3.2 AddSimpleParam

##### C++

```
HRESULT AddSimpleParam(BSTR _name, BSTR _default_val, VARIANT_BOOL* ret);
```

##### C#

```
public bool AddSimpleParam(String _name, String _default_val);
```

##### Visual Basic

```
Public Function AddSimpleParam(_name As String, _default_val As String) As Boolean
```

## Description

Function adds a simple parameter with the specified name to the schema. If the parameter with the specified name has been earlier defined in the schema, its definition will be changed. An attempt to change the definition of a parameter which has already been assigned to the structure element results in error and function returns the False value.

## Version

Available since version 8.2.

### III.3.3 GetParam

#### C++

```
HRESULT GetParam(long _idx, IRobotParamDef** ret);
```

#### C#

```
public IRobotParamDef GetParam(long _idx);
```

#### Visual Basic

```
Public Function GetParam(_idx As long) As IRobotParamDef
```

## Description

Function returns the parameter definition with the specified index in the schema.

## Version

Available since version 8.2.

### III.3.4 GetParamName

#### C++

```
HRESULT GetParamName(long _idx, BSTR* ret);
```

#### C#

```
public String GetParamName(long _idx);
```

#### Visual Basic

```
Public Function GetParamName(_idx As long) As String
```

## Description

Function returns the parameter name with the specified index.

## Version

Available since version 8.2.

### III.3.5 ParamNameToUniqueId

#### C++

```
HRESULT ParamNameToUniqueId(BSTR _name, BSTR* ret);
```

#### C#

```
public String ParamNameToUniqueId(String _name);
```

#### Visual Basic

```
Public Function ParamNameToUniqueId(_name As String) As String
```

**Description**

Function returns a unique identifier of the parameter with the specified name.

**Version**

Available since version 8.2.

**III.3.6 RemoveParam****C++**

```
HRESULT RemoveParam(BSTR _param_name);
```

**C#**

```
public void RemoveParam(String _param_name);
```

**Visual Basic**

```
Public Sub RemoveParam(_param_name As String)
```

**Description**

Function deletes the parameter with the specified name from the schema. Simultaneously, the parameter will be removed from all structure elements.

**Version**

Available since version 8.2.

**IV IRobotParamSchemaMngr****Class Hierarchy****C++**

```
interface IRobotParamSchemaMngr : IDispatch;
```

**C#**

```
public interface IRobotParamSchemaMngr;
```

**Visual Basic**

```
Public Interface IRobotParamSchemaMngr
```

**Description**

Interface for management of all parameter schemata defined in the structure.

**Version**

Available since version 8.2.

**IV.1 IRobotParamSchemaMngr Members**

The following tables list the members exposed by IRobotParamSchemaMngr.

**Public Fields**

	Name	Description
!	SchemaCount (see page 2182)	Number of defined schemata.

## Public Methods

	Name	Description
💡	Clear (🔗 see page 2183)	Function deletes all parameter schemata defined in the project. It involves simultaneously deletion of all external parameters defined in the project.
💡	Create (🔗 see page 2183)	Function creates a schema with the specified name. If the schema with this name already exists, the function returns its definition.
💡	Exist (🔗 see page 2184)	Function checks if the schema with the specified name exists.
💡	Get (🔗 see page 2184)	Function returns the parameter schema with the specified index.
💡	GetByName (🔗 see page 2184)	Function returns the schema with the specified name.
💡	Remove (🔗 see page 2185)	Function deletes the schema with the specified index.
💡	RemoveByName (🔗 see page 2185)	Function deletes the schema with the specified name.

## IV.2 IRobotParamSchemaMngr Fields

The fields of the IRobotParamSchemaMngr class are listed here.

### Public Fields

	Name	Description
💡	SchemaCount (🔗 see page 2182)	Number of defined schemata.

### IV.2.1 SchemaCount

#### C++

```
HRESULT get_SchemaCount(long*);
```

#### C#

```
public long SchemaCount { get; }
```

#### Visual Basic

```
Public ReadOnly SchemaCount As long
```

#### Description

Number of defined schemata.

#### Version

Available since version 8.2.

## IV.3 IRobotParamSchemaMngr Methods

The methods of the IRobotParamSchemaMngr class are listed here.

### Public Methods

	Name	Description
💡	Clear (🔗 see page 2183)	Function deletes all parameter schemata defined in the project. It involves simultaneously deletion of all external parameters defined in the project.
💡	Create (🔗 see page 2183)	Function creates a schema with the specified name. If the schema with this name already exists, the function returns its definition.
💡	Exist (🔗 see page 2184)	Function checks if the schema with the specified name exists.
💡	Get (🔗 see page 2184)	Function returns the parameter schema with the specified index.
💡	GetByName (🔗 see page 2184)	Function returns the schema with the specified name.
💡	Remove (🔗 see page 2185)	Function deletes the schema with the specified index.



RemoveByName (see page 2185) Function deletes the schema with the specified name.

### IV.3.1 Clear

#### C++

```
HRESULT Clear();
```

#### C#

```
public void Clear();
```

#### Visual Basic

```
Public Sub Clear()
```

#### Description

Function deletes all parameter schemata defined in the project. It involves simultaneously deletion of all external parameters defined in the project.

#### Version

Available since version 8.2.

### IV.3.2 Create

#### C++

```
HRESULT Create(BSTR _schema_name, IRobotParamSchema** ret);
```

#### C#

```
public IRobotParamSchema Create(String _schema_name);
```

#### Visual Basic

```
Public Function Create(_schema_name As String) As IRobotParamSchema
```

#### Description

Function creates a schema with the specified name. If the schema with this name already exists, the function returns its definition.

#### Version

Available since version 8.2.

### IV.3.3 Exist

#### C++

```
HRESULT Exist(BSTR _schema_name, VARIANT_BOOL* ret);
```

#### C#

```
public bool Exist(String _schema_name);
```

#### Visual Basic

```
Public Function Exist(_schema_name As String) As Boolean
```

#### Description

Function checks if the schema with the specified name exists.

#### Version

Available since version 8.2.

#### IV.3.4 Get

##### C++

```
HRESULT Get(long _idx, IRobotParamSchema** ret);
```

##### C#

```
public IRobotParamSchema Get(long _idx);
```

##### Visual Basic

```
Public Function Get(_idx As long) As IRobotParamSchema
```

##### Description

Function returns the parameter schema with the specified index.

##### Version

Available since version 8.2.

#### IV.3.5 GetByName

##### C++

```
HRESULT GetByName(BSTR _schema_name, IRobotParamSchema** ret);
```

##### C#

```
public IRobotParamSchema GetByName(String _schema_name);
```

##### Visual Basic

```
Public Function GetByName(_schema_name As String) As IRobotParamSchema
```

##### Description

Function returns the schema with the specified name.

##### Version

Available since version 8.2.

#### IV.3.6 Remove

##### C++

```
HRESULT Remove(long _idx);
```

##### C#

```
public void Remove(long _idx);
```

##### Visual Basic

```
Public Sub Remove(_idx As long)
```

##### Description

Function deletes the schema with the specified index.

##### Version

Available since version 8.2.

### IV.3.7 RemoveByName

#### C++

```
HRESULT RemoveByName(BSTR _schema_name);
```

#### C#

```
public void RemoveByName(String _schema_name);
```

#### Visual Basic

```
Public Sub RemoveByName(_schema_name As String)
```

#### Description

Function deletes the schema with the specified name.

#### Version

Available since version 8.2.

## V IRobotParamDef

#### Class Hierarchy

#### C++

```
interface IRobotParamDef : IDispatch;
```

#### C#

```
public interface IRobotParamDef;
```

#### Visual Basic

```
Public Interface IRobotParamDef
```

#### Description

Complete definition of an external parameter.

#### Version

Available since version 8.2.

## V.1 IRobotParamDef Members

The following tables list the members exposed by IRobotParamDef.

#### Public Fields

	Name	Description
❖	DefaultValue (see page 2187)	Default parameter value.
❖	GuiReadOnly (see page 2187)	Flag indicating if a given parameter can be edited in the parameter browser.
❖	GuiVisible (see page 2187)	Flag indicating if a given parameter should be visible in the parameter browser.
❖	Name (see page 2188)	Parameter name - should be unique in the schema.
❖	UniquelId (see page 2188)	Automatically-assigned parameter identifier, unique in the structure.
❖	ValueList (see page 2188)	List of possible values, if the parameter value is limited only to the specified list.
❖	ValueType (see page 2188)	Parameter value type.

## Public Methods

	Name	Description
✖	GuiGetName (see page 2189)	Function returns the attribute name in the specified language.
✖	GuiSetName (see page 2189)	Function sets a language specific attribute name displayed in the attribute browser.

## V.2 IRobotParamDef Fields

The fields of the IRobotParamDef class are listed here.

### Public Fields

	Name	Description
❖	DefaultValue (see page 2187)	Default parameter value.
❖	GuiReadOnly (see page 2187)	Flag indicating if a given parameter can be edited in the parameter browser.
❖	GuiVisible (see page 2187)	Flag indicating if a given parameter should be visible in the parameter browser.
❖	Name (see page 2188)	Parameter name - should be unique in the schema.
❖	Uniqueld (see page 2188)	Automatically-assigned parameter identifier, unique in the structure.
❖	ValueList (see page 2188)	List of possible values, if the parameter value is limited only to the specified list.
❖	ValueType (see page 2188)	Parameter value type.

### V.2.1 DefaultValue

#### C++

```
HRESULT get_DefaultValue(BSTR* );
HRESULT put_DefaultValue(BSTR);
```

#### C#

```
public String DefaultValue { get; set; }
```

#### Visual Basic

```
Public DefaultValue As String
```

#### Description

Default parameter value.

#### Version

Available since version 8.2.

### V.2.2 GuiReadOnly

#### C++

```
HRESULT get_GuiReadOnly(VARIANT_BOOL* );
HRESULT put_GuiReadOnly(VARIANT_BOOL);
```

#### C#

```
public bool GuiReadOnly { get; set; }
```

#### Visual Basic

```
Public GuiReadOnly As Boolean
```

#### Description

Flag indicating if a given parameter can be edited in the parameter browser.

**Version**

Available since version 8.2.

### V.2.3 GuiVisible

**C++**

```
HRESULT get_GuiVisible(VARIANT_BOOL* );
HRESULT put_GuiVisible(VARIANT_BOOL);
```

**C#**

```
public bool GuiVisible { get; set; }
```

**Visual Basic**

```
Public GuiVisible As Boolean
```

**Description**

Flag indicating if a given parameter should be visible in the parameter browser.

**Version**

Available since version 8.2.

### V.2.4 Name

**C++**

```
HRESULT get_Name(BSTR* );
HRESULT put_Name(BSTR);
```

**C#**

```
public String Name { get; set; }
```

**Visual Basic**

```
Public Name As String
```

**Description**

Parameter name - should be unique in the schema.

**Version**

Available since version 8.2.

### V.2.5 UniqueId

**C++**

```
HRESULT get_UniqueId(BSTR* );
```

**C#**

```
public String UniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly UniqueId As String
```

**Description**

Automatically-assigned parameter identifier, unique in the structure.

**Version**

Available since version 8.2.

## V.2.6 ValueList

### C++

```
HRESULT get_ValueList(IRobotNamesArray**);
```

### C#

```
public IRobotNamesArray ValueList { get; }
```

### Visual Basic

```
Public ReadOnly ValueList As IRobotNamesArray
```

### Description

List of possible values, if the parameter value is limited only to the specified list.

### Version

Available since version 8.2.

## V.2.7 ValueType

### C++

```
HRESULT get_ValueType(IRobotParamValueType*);  
HRESULT put_ValueType(IRobotParamValueType);
```

### C#

```
public IRobotParamValueType ValueType { get; set; }
```

### Visual Basic

```
Public ValueType As IRobotParamValueType
```

### Description

Parameter value type.

### Version

Available since version 8.2.

## V.3 IRobotParamDef Methods

The methods of the IRobotParamDef class are listed here.

### Public Methods

	Name	Description
💡	GuiGetName (see page 2189)	Function returns the attribute name in the specified language.
💡	GuiSetName (see page 2189)	Function sets a language specific attribute name displayed in the attribute browser.

## V.3.1 GuiGetName

### C++

```
HRESULT GuiGetName(long _lang_id, BSTR* ret);
```

### C#

```
public String GuiGetName(long _lang_id);
```

### Visual Basic

```
Public Function GuiGetName(_lang_id As long) As String
```

**Description**

Function returns the attribute name in the specified language.

**Version**

Available since version 8.2.

**V.3.2 GuiSetName****C++**

```
HRESULT GuiSetName(long _lang_id, BSTR _name);
```

**C#**

```
public void GuiSetName(long _lang_id, String _name);
```

**Visual Basic**

```
Public Sub GuiSetName(_lang_id As long, _name As String)
```

**Description**

Function sets a language specific attribute name displayed in the attribute browser.

**Version**

Available since version 8.2.

**VI IRobotParamValueType****C++**

```
enum IRobotParamValueType;
```

**C#**

```
public enum IRobotParamValueType;
```

**Visual Basic**

```
Public Enum IRobotParamValueType
```

**Members**

Members	Description
I_PVT_TEXT = 1	Character string. Available since version 8.2.
I_PVT_INTEGER = 2	Integer. Available since version 8.2.
I_PVT_REAL = 3	Real number. Available since version 8.2.

**Description**

Available types of external parameter values.

**Version**

Available since version 8.2.

**VII IRobotParamSchema****Class Hierarchy**

**C++**

```
interface IRobotParamSchema : IDispatch;
```

**C#**

```
public interface IRobotParamSchema;
```

**Visual Basic**

```
Public Interface IRobotParamSchema
```

**Description**

Parameter schemata have been implemented on order to avoid a collision of names of parameters defined by different external applications. It is recommended that each external application should define its own separate parameter schema.

**Version**

Available since version 8.2.

## VII.1 IRobotParamSchema Members

The following tables list the members exposed by IRobotParamSchema.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Def ( <a href="#">see page 2191</a> )	Scheme definition.

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	GetAllParams ( <a href="#">see page 2192</a> )	Function enters to the collection all parameters from this schema assigned to the given structure element. The number of parameters is returned.
◆	GetObjectsWithParam ( <a href="#">see page 2192</a> )	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.
◆	GetObjectsWithParamVal ( <a href="#">see page 2193</a> )	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.
◆	GetParam ( <a href="#">see page 2193</a> )	Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value. .
◆	RemoveAllParams ( <a href="#">see page 2193</a> )	Function deletes all parameters belonging to this schema from the specified structure element.
◆	RemoveParam ( <a href="#">see page 2194</a> )	Function deletes a determined parameter from the specified structure element.
◆	ResetParam ( <a href="#">see page 2194</a> )	Function assigns a default value of the determined parameter to the specified structure element.
◆	SetParam ( <a href="#">see page 2194</a> )	Function assigns a determined parameter value to the specified structure element.

## VII.2 IRobotParamSchema Fields

The fields of the IRobotParamSchema class are listed here.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	Def ( <a href="#">see page 2191</a> )	Scheme definition.

## VII.2.1 Def

### C++

```
HRESULT get_Def(IRobotParamSchemaDef**);
```

### C#

```
public IRobotParamSchemaDef Def { get; }
```

### Visual Basic

```
Public ReadOnly Def As IRobotParamSchemaDef
```

### Description

Scheme definition.

### Version

Available since version 8.2.

## VII.3 IRobotParamSchema Methods

The methods of the IRobotParamSchema class are listed here.

### Public Methods

	Name	Description
⊕	GetAllParams (see page 2192)	Function enters to the collection all parameters from this schema assigned to the given structure element. The number of parameters is returned.
⊕	GetObjectsWithParam (see page 2192)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.
⊕	GetObjectsWithParamVal (see page 2193)	Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.
⊕	GetParam (see page 2193)	Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value. .
⊕	RemoveAllParams (see page 2193)	Function deletes all parameters belonging to this schema from the specified structure element.
⊕	RemoveParam (see page 2194)	Function deletes a determined parameter from the specified structure element.
⊕	ResetParam (see page 2194)	Function assigns a default value of the determined parameter to the specified structure element.
⊕	SetParam (see page 2194)	Function assigns a determined parameter value to the specified structure element.

### VII.3.1 GetAllParams

#### C++

```
HRESULT GetAllParams(long _unique_obj_id, IRobotParamCollection* _params_col, long* ret);
```

#### C#

```
public long GetAllParams(long _unique_obj_id, IRobotParamCollection _params_col);
```

#### Visual Basic

```
Public Function GetAllParams(_unique_obj_id As long, ByRef _params_col As IRobotParamCollection) As long
```

## Description

Function enters to the collection all parameters from this schema assigned to the given structure element. The number of parameters is returned.

## Version

Available since version 8.2.

### VII.3.2 GetObjectsWithParam

#### C++

```
HRESULT GetObjectsWithParam(BSTR _param_name, IRobotNumbersArray* _obj_ids, long* ret);
```

#### C#

```
public long GetObjectsWithParam(String _param_name, IRobotNumbersArray _obj_ids);
```

#### Visual Basic

```
Public Function GetObjectsWithParam(_param_name As String, ByRef _obj_ids As IRobotNumbersArray) As long
```

## Description

Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter. Function returns the number of added identifiers.

## Version

Available since version 8.2.

### VII.3.3 GetObjectsWithParamVal

#### C++

```
HRESULT GetObjectsWithParamVal(BSTR _param_name, BSTR _param_val, IRobotNumbersArray* _obj_ids, long* ret);
```

#### C#

```
public long GetObjectsWithParamVal(String _param_name, String _param_val, IRobotNumbersArray _obj_ids);
```

#### Visual Basic

```
Public Function GetObjectsWithParamVal(_param_name As String, _param_val As String, ByRef _obj_ids As IRobotNumbersArray) As long
```

## Description

Function adds to the table unique identifiers of all structure elements which were assigned the determined parameter with the specified value. Function returns the number of added identifiers.

## Version

Available since version 8.2.

### VII.3.4 GetParam

#### C++

```
HRESULT GetParam(long _unique_obj_id, BSTR _param_name, BSTR _ret_param_val, VARIANT_BOOL* ret);
```

#### C#

```
public bool GetParam(long _unique_obj_id, String _param_name, String _ret_param_val);
```

**Visual Basic**

```
Public Function GetParam(_unique_obj_id As long, _param_name As String, ByRef
_ret_param_val As String) As Boolean
```

**Description**

Function gives access to a value of the specified parameter assigned to a determined structure element. If a given parameter has not been assigned to the structure element, function returns the False value..

**Version**

Available since version 8.2.

**VII.3.5 RemoveAllParams****C++**

```
HRESULT RemoveAllParams(long _unique_obj_id);
```

**C#**

```
public void RemoveAllParams(long _unique_obj_id);
```

**Visual Basic**

```
Public Sub RemoveAllParams(_unique_obj_id As long)
```

**Description**

Function deletes all parameters belonging to this schema from the specified structure element.

**Version**

Available since version 8.2.

**VII.3.6 RemoveParam****C++**

```
HRESULT RemoveParam(long _unique_obj_id, BSTR _param_name);
```

**C#**

```
public void RemoveParam(long _unique_obj_id, String _param_name);
```

**Visual Basic**

```
Public Sub RemoveParam(_unique_obj_id As long, _param_name As String)
```

**Description**

Function deletes a determined parameter from the specified structure element.

**Version**

Available since version 8.2.

**VII.3.7 ResetParam****C++**

```
HRESULT ResetParam(long _unique_obj_id, BSTR _param_name);
```

**C#**

```
public void ResetParam(long _unique_obj_id, String _param_name);
```

**Visual Basic**

```
Public Sub ResetParam(_unique_obj_id As long, _param_name As String)
```

**Description**

Function assigns a default value of the determined parameter to the specified structure element.

**Version**

Available since version 8.2.

**VII.3.8 SetParam****C++**

```
HRESULT SetParam(long _unique_obj_id, BSTR _param_name, BSTR _param_val);
```

**C#**

```
public void SetParam(long _unique_obj_id, String _param_name, String _param_val);
```

**Visual Basic**

```
Public Sub SetParam(_unique_obj_id As long, _param_name As String, _param_val As String)
```

**Description**

Function assigns a determined parameter value to the specified structure element.

**Version**

Available since version 8.2.

**Robot Kernel**

Available since version 2.5.

**Interfaces**

	Name	Description
→	IRobotKernel ( <a href="#">see page 2195</a> )	Main interface of the Robot program kernel.
→	IRobotKernelPreferences ( <a href="#">see page 2203</a> )	Interface describing parameters of the Robot program kernel.

**IRobotKernel****Class Hierarchy****C++**

```
interface IRobotKernel : IDispatch;
```

**C#**

```
public interface IRobotKernel;
```

**Visual Basic**

```
Public Interface IRobotKernel
```

**Description**

Main interface of the Robot program kernel.

## Version

Available since version 2.5.

## I.1 IRobotKernel Members

The following tables list the members exposed by IRobotKernel.

### Public Fields

	Name	Description
◆	CalcEngine ( <a href="#">see page 2197</a> )	
◆	CmpntFactory ( <a href="#">see page 2197</a> )	
◆	ConcrReinfEngine ( <a href="#">see page 2197</a> )	Calculation module for the concrete reinforcement.
◆	Preferences ( <a href="#">see page 2197</a> )	Settings for the Robot program kernel .
◆	ProgramName ( <a href="#">see page 2198</a> )	Program name.
◆	ProgramPath ( <a href="#">see page 2198</a> )	Full path to the main executable file for process to which RobotKernel has been loaded.
◆	ProgramVersion ( <a href="#">see page 2198</a> )	Designation of the program version.
◆	ProjectActiveModel ( <a href="#">see page 2199</a> )	.
◆	ProjectPreferences ( <a href="#">see page 2199</a> )	Current project settings.
◆	ProjectUniqueId ( <a href="#">see page 2199</a> )	Unique project identifier.
◆	Structure ( <a href="#">see page 2200</a> )	Interface describing the entire structure.
◆	Version ( <a href="#">see page 2200</a> )	Version number.

### Public Methods

	Name	Description
◆	GetExtension ( <a href="#">see page 2201</a> )	Function returns the interface for the module extending functionality of the Robot program kernel. If the extension with the indicated name has not been installed, then the function returns zero value (Nothing).
◆	ProjectNew ( <a href="#">see page 2201</a> )	Function creates a new project of the indicated type. .
◆	ProjectNewFromTemplate ( <a href="#">see page 2201</a> )	Function creates a new document based on the template defined by the specified file. .
◆	ProjectOpen ( <a href="#">see page 2202</a> )	Function opens the project saved in the indicated RTD format file. .
◆	ProjectSave ( <a href="#">see page 2202</a> )	Function saves the current project to an RTD format file. .
◆	ProjectSaveAs ( <a href="#">see page 2202</a> )	Function saves the current project to the indicated RTD format file. .

## I.2 IRobotKernel Fields

The fields of the IRobotKernel class are listed here.

### Public Fields

	Name	Description
◆	CalcEngine ( <a href="#">see page 2197</a> )	
◆	CmpntFactory ( <a href="#">see page 2197</a> )	
◆	ConcrReinfEngine ( <a href="#">see page 2197</a> )	Calculation module for the concrete reinforcement.
◆	Preferences ( <a href="#">see page 2197</a> )	Settings for the Robot program kernel .
◆	ProgramName ( <a href="#">see page 2198</a> )	Program name.
◆	ProgramPath ( <a href="#">see page 2198</a> )	Full path to the main executable file for process to which RobotKernel has been loaded.

❖	ProgramVersion (see page 2198)	Designation of the program version.
❖	ProjectActiveModel (see page 2199)	.
❖	ProjectPreferences (see page 2199)	Current project settings.
❖	ProjectUniqueId (see page 2199)	Unique project identifier.
❖	Structure (see page 2200)	Interface describing the entire structure.
❖	Version (see page 2200)	Version number.

## I.2.1 CalcEngine

**C++**

```
HRESULT get_CalcEngine(IRobotCalcEngine**);
```

**C#**

```
public IRobotCalcEngine CalcEngine { get; }
```

**Visual Basic**

```
Public ReadOnly CalcEngine As IRobotCalcEngine
```

**Version**

Available since version 2.5.

## I.2.2 CmpntFactory

**C++**

```
HRESULT get_CmpntFactory(IRobotComponentFactory**);
```

**C#**

```
public IRobotComponentFactory CmpntFactory { get; }
```

**Visual Basic**

```
Public ReadOnly CmpntFactory As IRobotComponentFactory
```

**Version**

Available since version 2.5.

## I.2.3 ConcrReinfEngine

**C++**

```
HRESULT get_ConcrReinfEngine(IRConcrCalcEngine**);
```

**C#**

```
public IRConcrCalcEngine ConcrReinfEngine { get; }
```

**Visual Basic**

```
Public ReadOnly ConcrReinfEngine As IRConcrCalcEngine
```

**Description**

Calculation module for the concrete reinforcement.

**Version**

Available since version 5.5.

## I.2.4 Preferences

### C++

```
HRESULT get_Preferences(IRobotKernelPreferences**);
```

### C#

```
public IRobotKernelPreferences Preferences { get; }
```

### Visual Basic

```
Public ReadOnly Preferences As IRobotKernelPreferences
```

### Description

Settings for the Robot program kernel.

### Version

Available since version 2.5.

## I.2.5 ProgramName

### C++

```
HRESULT get_ProgramName(BSTR*);
```

### C#

```
public String ProgramName { get; }
```

### Visual Basic

```
Public ReadOnly ProgramName As String
```

### Description

Program name.

### Version

Available since version 5.5.

## I.2.6 ProgramPath

### C++

```
HRESULT get_ProgramPath(BSTR*);
```

### C#

```
public String ProgramPath { get; }
```

### Visual Basic

```
Public ReadOnly ProgramPath As String
```

### Description

Full path to the main executable file for process to which RobotKernel has been loaded.

### Version

Available since version 13.4.

## I.2.7 ProgramVersion

### C++

```
HRESULT get_ProgramVersion(BSTR*);
```

**C#**

```
public String ProgramVersion { get; }
```

**Visual Basic**

```
Public ReadOnly ProgramVersion As String
```

**Description**

Designation of the program version.

**Version**

Available since version 5.5.

## I.2.8 ProjectActiveModel

**C++**

```
HRESULT get_ProjectActiveModel(IRobotActiveModelType* );
HRESULT put_ProjectActiveModel(IRobotActiveModelType);
```

**C#**

```
public IRobotActiveModelType ProjectActiveModel { get; set; }
```

**Visual Basic**

```
Public ProjectActiveModel As IRobotActiveModelType
```

**Description****Version**

Available since version 15.

## I.2.9 ProjectPreferences

**C++**

```
HRESULT get_ProjectPreferences(IRobotProjectPreferences** );
```

**C#**

```
public IRobotProjectPreferences ProjectPreferences { get; }
```

**Visual Basic**

```
Public ReadOnly ProjectPreferences As IRobotProjectPreferences
```

**Description**

Current project settings.

**Version**

Available since version 2.5.

## I.2.10 ProjectUniqueId

**C++**

```
HRESULT get_ProjectUniqueId(BSTR* );
```

**C#**

```
public String ProjectUniqueId { get; }
```

**Visual Basic**

```
Public ReadOnly ProjectUniqueId As String
```

**Description**

Unique project identifier.

**Version**

Available since version 2.5.

**I.2.11 Structure****C++**

```
HRESULT get_Structure(IRobotStructure**);
```

**C#**

```
public IRobotStructure Structure { get; }
```

**Visual Basic**

```
Public ReadOnly Structure As IRobotStructure
```

**Description**

Interface describing the entire structure.

**Version**

Available since version 2.5.

**I.2.12 Version****C++**

```
HRESULT get_Version(double*);
```

**C#**

```
public double Version { get; }
```

**Visual Basic**

```
Public ReadOnly Version As Double
```

**Description**

Version number.

**Version**

Available since version 2.5.

**I.3 IRobotKernel Methods**

The methods of the IRobotKernel class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
✳	GetExtension (see page 2201)	Function returns the interface for the module extending functionality of the Robot program kernel. If the extension with the indicated name has not been installed, then the function returns zero value (Nothing).
✳	ProjectNew (see page 2201)	Function creates a new project of the indicated type. .
✳	ProjectNewFromTemplate (see page 2201)	Function creates a new document based on the template defined by the specified file. .
✳	ProjectOpen (see page 2202)	Function opens the project saved in the indicated RTD format file. .
✳	ProjectSave (see page 2202)	Function saves the current project to an RTD format file. .
✳	ProjectSaveAs (see page 2202)	Function saves the current project to the indicated RTD format file. .

### I.3.1 GetExtension

**C++**

```
HRESULT GetExtension(BSTR _extmdl_name, IDispatch* ret);
```

**C#**

```
public IDispatch GetExtension(String _extmdl_name);
```

**Visual Basic**

```
Public Function GetExtension(_extmdl_name As String) As IDispatch
```

**Description**

Function returns the interface for the module extending functionality of the Robot program kernel. If the extension with the indicated name has not been installed, then the function returns zero value (Nothing).

**Version**

Available since version 2.5.

### I.3.2 ProjectNew

**C++**

```
HRESULT ProjectNew(IRobotProjectType _prj_type, VARIANT_BOOL* ret);
```

**C#**

```
public bool ProjectNew(IRobotProjectType _prj_type);
```

**Visual Basic**

```
Public Function ProjectNew(_prj_type As IRobotProjectType) As Boolean
```

**Description**

Function creates a new project of the indicated type. .

**Version**

Available since version 2.5.

### I.3.3 ProjectNewFromTemplate

**C++**

```
HRESULT ProjectNewFromTemplate(BSTR _tmpl_file, VARIANT_BOOL* ret);
```

**C#**

```
public bool ProjectNewFromTemplate(String _tmpl_file);
```

**Visual Basic**

```
Public Function ProjectNewFromTemplate(_tmpl_file As String) As Boolean
```

**Description**

Function creates a new document based on the template defined by the specified file. .

**Version**

Available since version 3.5.

### I.3.4 ProjectOpen

**C++**

```
HRESULT ProjectOpen(BSTR _file_path, VARIANT_BOOL* ret);
```

**C#**

```
public bool ProjectOpen(String _file_path);
```

**Visual Basic**

```
Public Function ProjectOpen(_file_path As String) As Boolean
```

**Description**

Function opens the project saved in the indicated RTD format file. .

**Version**

Available since version 2.5.

### I.3.5 ProjectSave

**C++**

```
HRESULT ProjectSave();
```

**C#**

```
public void ProjectSave();
```

**Visual Basic**

```
Public Sub ProjectSave()
```

**Description**

Function saves the current project to an RTD format file. .

**Version**

Available since version 2.5.

### I.3.6 ProjectSaveAs

**C++**

```
HRESULT ProjectSaveAs(BSTR _file_path);
```

**C#**

```
public void ProjectSaveAs(String _file_path);
```

**Visual Basic**

```
Public Sub ProjectSaveAs(_file_path As String)
```

**Description**

Function saves the current project to the indicated RTD format file. .

**Version**

Available since version 2.5.

## II IRobotKernelPreferences

### Class Hierarchy

#### C++

```
interface IRobotKernelPreferences : IDispatch;
```

#### C#

```
public interface IRobotKernelPreferences;
```

### Visual Basic

```
Public Interface IRobotKernelPreferences
```

### Description

Interface describing parameters of the Robot program kernel.

### Version

Available since version 2.5.

## II.1 IRobotKernelPreferences Members

The following tables list the members exposed by IRobotKernelPreferences.

### Public Fields

	Name	Description
◆	Multiprocessing (see page 2204)	Flag activating parallel processing - multiprocessing.

### Public Methods

	Name	Description
♫	GetDirectory (see page 2204)	Function returns the full path to the selected folder.
♫	GetDirectoryExt (see page 2205)	Function returns folder extension. If for a specified folder no extension has been defined, function returns an empty character string. .
♫	GetLanguage (see page 2205)	Function returns the current setting of the indicated language.
♫	SetDirectoryExt (see page 2205)	Function defines a folder extension. If an empty character string is given a as a full path to the extension, it will result in deleting the extension. Meaning of the folder extension is best illustrated by the example. If the SetDirectoryExt( I_DE_TEMPLATE, "C:My Templates") function is activated, then from this moment on template files will be searched first in the C:My Templates folder and next, in the standard folder with template files. .
♫	SetLanguage (see page 2206)	

## II.2 IRobotKernelPreferences Fields

The fields of the IRobotKernelPreferences class are listed here.

### Public Fields

	Name	Description
◆	Multiprocessing (see page 2204)	Flag activating parallel processing - multiprocessing.

### II.2.1 Multiprocessing

#### C++

```
HRESULT get_Multiprocessing(VARIANT_BOOL* );
```

```
HRESULT put_Multiprocessing(VARIANT_BOOL);
```

**C#**

```
public bool Multiprocessing { get; set; }
```

**Visual Basic**

```
Public Multiprocessing As Boolean
```

**Description**

Flag activating parallel processing - multiprocessing.

**Version**

Available since version 8.5.

## II.3 IRobotKernelPreferences Methods

The methods of the IRobotKernelPreferences class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	GetDirectory (see page 2204)	Function returns the full path to the selected folder.
💡	GetDirectoryExt (see page 2205)	Function returns folder extension. If for a specified folder no extension has been defined, function returns an empty character string..
💡	GetLanguage (see page 2205)	Function returns the current setting of the indicated language.
💡	SetDirectoryExt (see page 2205)	Function defines a folder extension. If an empty character string is given a as a full path to the extension, it will result in deleting the extension. Meaning of the folder extension is best illustrated by the example. If the SetDirectoryExt( I_DE_TEMPLATE, "C:My Templates") function is activated, then from this moment on template files will be searched first in the C:My Templates folder and next, in the standard folder with template files. .
💡	SetLanguage (see page 2206)	

### II.3.1 GetDirectory

**C++**

```
HRESULT GetDirectory(IRobotDirectory _rob_dir, BSTR* ret);
```

**C#**

```
public String GetDirectory(IRobotDirectory _rob_dir);
```

**Visual Basic**

```
Public Function GetDirectory(_rob_dir As IRobotDirectory) As String
```

**Description**

Function returns the full path to the selected folder.

**Version**

Available since version 2.5.

### II.3.2 GetDirectoryExt

**C++**

```
HRESULT GetDirectoryExt(IRobotDirectoryExtension _dir_ext, BSTR* ret);
```

**C#**

```
public String GetDirectoryExt(IRobotDirectoryExtension _dir_ext);
```

**Visual Basic**

```
Public Function GetDirectoryExt(_dir_ext As IRobotDirectoryExtension) As String
```

**Description**

Function returns folder extension. If for a specified folder no extension has been defined, function returns an empty character string. .

**Version**

Available since version 3.5.

**II.3.3 GetLanguage****C++**

```
HRESULT GetLanguage(IRobotLanguage _lang_id, int* ret);
```

**C#**

```
public int GetLanguage(IRobotLanguage _lang_id);
```

**Visual Basic**

```
Public Function GetLanguage(_lang_id As IRobotLanguage) As int
```

**Description**

Function returns the current setting of the indicated language.

**Version**

Available since version 2.5.

**II.3.4 SetDirectoryExt****C++**

```
HRESULT SetDirectoryExt(IRobotDirectoryExtension _dir_ext, BSTR _full_path);
```

**C#**

```
public void SetDirectoryExt(IRobotDirectoryExtension _dir_ext, String _full_path);
```

**Visual Basic**

```
Public Sub SetDirectoryExt(_dir_ext As IRobotDirectoryExtension, _full_path As String)
```

**Description**

Function defines a folder extension. If an empty character string is given a as a full path to the extension, it will result in deleting the extension.

Meaning of the folder extension is best illustrated by the example. If the SetDirectoryExt( I\_DE\_TEMPLATE, "C:My Templates") function is activated, then from this moment on template files will be searched first in the C:My Templates folder and next, in the standard folder with template files. .

**Version**

Available since version 3.5.

## II.3.5 SetLanguage

**C++**

```
HRESULT SetLanguage(IRobotLanguage _lang_id, int _lang_num);
```

**C#**

```
public void SetLanguage(IRobotLanguage _lang_id, int _lang_num);
```

**Visual Basic**

```
Public Sub SetLanguage(_lang_id As IRobotLanguage, _lang_num As int)
```

**Version**

Available since version 2.5.

# STR files analyzer

Available since version 2.5.

**Interfaces**

	Name	Description
»	IRobotSTRFileAnalyser (see page 2206)	Analyzer of STR format text files.

## I IRobotSTRFileAnalyser

**Class Hierarchy**

**C++**

```
interface IRobotSTRFileAnalyser : IDispatch;
```

**C#**

```
public interface IRobotSTRFileAnalyser;
```

**Visual Basic**

```
Public Interface IRobotSTRFileAnalyser
```

**Description**

Analyzer of STR format text files.

**Version**

Available since version 2.5.

## I.1 IRobotSTRFileAnalyser Members

The following tables list the members exposed by IRobotSTRFileAnalyser.

**Public Fields**

	Name	Description
◆	InsertParams (see page 2207)	Parameters of inserting the structure defined in an STR file to the current project.
◆	Params (see page 2207)	Parameters defined in the STR file.

## Public Methods

	Name	Description
💡	InsertToProject (see page 2208)	Function inserts definition of the structure described in the indicated STR file to the current project. If the parameters of the inserted structure have been read out and modified earlier, then their new values will be considered during structure insertion..
💡	ReadParams (see page 2208)	Function reads in the parameters defined in the indicated STR file. Once read in, the parameters are available by means of the Params (see page 2207) component and may be edited. If read-out of variables fails, then zero value will be returned (False). .

## I.2 IRobotSTRFileAnalyser Fields

The fields of the IRobotSTRFileAnalyser class are listed here.

### Public Fields

	Name	Description
💡	InsertParams (see page 2207)	Parameters of inserting the structure defined in an STR file to the current project.
💡	Params (see page 2207)	Parameters defined in the STR file.

### I.2.1 InsertParams

#### C++

```
HRESULT get_InsertParams(IRobotFileInsertParams**);
```

#### C#

```
public IRobotFileInsertParams InsertParams { get; }
```

#### Visual Basic

```
Public ReadOnly InsertParams As IRobotFileInsertParams
```

#### Description

Parameters of inserting the structure defined in an STR file to the current project.

#### Version

Available since version 2.5.

### I.2.2 Params

#### C++

```
HRESULT get_Params(IRobotSTRParams**);
```

#### C#

```
public IRobotSTRParams Params { get; }
```

#### Visual Basic

```
Public ReadOnly Params As IRobotSTRParams
```

#### Description

Parameters defined in the STR file.

#### Version

Available since version 2.5.

## I.3 IRobotSTRFileAnalyser Methods

The methods of the IRobotSTRFileAnalyser class are listed here.

### Public Methods

	Name	Description
💡	InsertToProject (see page 2208)	Function inserts definition of the structure described in the indicated STR file to the current project. If the parameters of the inserted structure have been read out and modified earlier, then their new values will be considered during structure insertion. .
💡	ReadParams (see page 2208)	Function reads in the parameters defined in the indicated STR file. Once read in, the parameters are available by means of the Params (see page 2207) component and may be edited. If read-out of variables fails, then zero value will be returned (False). .

### I.3.1 InsertToProject

#### C++

```
HRESULT InsertToProject(BSTR _file_path, VARIANT_BOOL _ignore_warnings, VARIANT_BOOL _only_geometry, VARIANT_BOOL* ret);
```

#### C#

```
public bool InsertToProject(String _file_path, bool _ignore_warnings, bool _only_geometry);
```

#### Visual Basic

```
Public Function InsertToProject(_file_path As String, _ignore_warnings As Boolean, _only_geometry As Boolean) As Boolean
```

#### Description

Function inserts definition of the structure described in the indicated STR file to the current project. If the parameters of the inserted structure have been read out and modified earlier, then their new values will be considered during structure insertion. .

#### Version

Available since version 2.5.

### I.3.2 ReadParams

#### C++

```
HRESULT ReadParams(BSTR _file_path, VARIANT_BOOL* ret);
```

#### C#

```
public bool ReadParams(String _file_path);
```

#### Visual Basic

```
Public Function ReadParams(_file_path As String) As Boolean
```

#### Description

Function reads in the parameters defined in the indicated STR file. Once read in, the parameters are available by means of the Params (see page 2207) component and may be edited. If read-out of variables fails, then zero value will be returned (False). .

#### Version

Available since version 2.5.

# Structure correction

Available since version 3.5.

## Enumerations

	Name	Description
	IRobotBarIntersectRelationship (see page 2209)	Relations between bars resulting from bar intersections. .

## Interfaces

	Name	Description
	IRobotStructureGeoAnalyser (see page 2210)	Tool component performing analysis of structure geometry and its automatic correction in order to obtain a correct calculation model.

## I IRobotBarIntersectRelationship

### C++

```
enum IRobotBarIntersectRelationship;
```

### C#

```
public enum IRobotBarIntersectRelationship;
```

### Visual Basic

```
Public Enum IRobotBarIntersectRelationship
```

## Members

Members	Description
I_BIR_UNKNOWN = 0	It is not known if bars remaining in this relation intersect or not. Available since version 3.5.
I_BIR_NOT_INTERSECT = 1	Bars remaining in this relation do not intersect. Available since version 3.5.
I_BIR_INTERSECT_MID_MID = 2	Bars remaining in this relation intersect. Available since version 3.5.
I_BIR_INTERSECT_MID_END = 3	Available since version 3.5.
I_BIR_INTERSECT_END_MID = 4	Available since version 3.5.
I_BIR_INTERSECT_END_END = 5	Available since version 3.5.

## Description

Relations between bars resulting from bar intersections. .

## Version

Available since version 3.5.

## II IRobotStructureGeoAnalyser

### Class Hierarchy

### C++

```
interface IRobotStructureGeoAnalyser : IDispatch;
```

**C#**

```
public interface IRobotStructureGeoAnalyser;
```

**Visual Basic**

```
Public Interface IRobotStructureGeoAnalyser
```

**Description**

Tool component performing analysis of structure geometry and its automatic correction in order to obtain a correct calculation model.

**Version**

Available since version 3.5.

## II.1 IRobotStructureGeoAnalyser Members

The following tables list the members exposed by IRobotStructureGeoAnalyser.

**Public Fields**

	<b>Name</b>	<b>Description</b>
◆	CanEliminateIsolatedNodes ( <a href="#">see page 2211</a> )	Flag steering with work of the Correct ( <a href="#">see page 2213</a> ) function; the True value indicates that it is possible to delete isolated nodes in order to perform structure correction.
◆	CanExtendBars ( <a href="#">see page 2211</a> )	Flag managing operation of the Correct ( <a href="#">see page 2213</a> ) function; True (default) value means that to carry out structure correction bar length may be increased.
◆	CanUseRigidLinks ( <a href="#">see page 2212</a> )	Flag managing operation of the Correct ( <a href="#">see page 2213</a> ) function; True (default) value means that to carry out structure correction rigid links may be defined.
◆	Precision ( <a href="#">see page 2212</a> )	Precision applied by the Correct ( <a href="#">see page 2213</a> ) function; setting the value (-1) indicates adoption of a default value .

**Public Methods**

	<b>Name</b>	<b>Description</b>
◆	Correct ( <a href="#">see page 2213</a> )	Function carries out geometry correction for a whole structure or for a structure part based on the parameters set earlier. .
◆	DefineIntersection ( <a href="#">see page 2213</a> )	Function defines a relation of intersection between the specified bars. This information will be used by the function of structure geometry correction. .
◆	SetEdgeSize ( <a href="#">see page 2213</a> )	Function defines the "thickness" of a geometric object edge.
◆	SetNodeSize ( <a href="#">see page 2214</a> )	Function defines the "size" of a node. Structure elements that are located in a distance equal or less than the specified radius from the node will be treated as adjoining to the node.
◆	StartCollectInfo ( <a href="#">see page 2214</a> )	Function runs analysis of operations performed on the structure to make subsequent correction of its geometry easier. .
◆	StopCollectInfo ( <a href="#">see page 2214</a> )	

## II.2 IRobotStructureGeoAnalyser Fields

The fields of the IRobotStructureGeoAnalyser class are listed here.

## Public Fields

	Name	Description
◆	CanEliminateIsolatedNodes (see page 2211)	Flag steering with work of the Correct (see page 2213) function; the True value indicates that it is possible to delete isolated nodes in order to perform structure correction.
◆	CanExtendBars (see page 2211)	Flag managing operation of the Correct (see page 2213) function; True (default) value means that to carry out structure correction bar length may be increased.
◆	CanUseRigidLinks (see page 2212)	Flag managing operation of the Correct (see page 2213) function; True (default) value means that to carry out structure correction rigid links may be defined.
◆	Precision (see page 2212)	Precision applied by the Correct (see page 2213) function; setting the value (-1) indicates adoption of a default value .

### II.2.1 CanEliminateIsolatedNodes

#### C++

```
HRESULT get_CanEliminateIsolatedNodes(VARIANT_BOOL* );
HRESULT put_CanEliminateIsolatedNodes(VARIANT_BOOL);
```

#### C#

```
public bool CanEliminateIsolatedNodes { get; set; }
```

#### Visual Basic

```
Public CanEliminateIsolatedNodes As Boolean
```

#### Description

Flag steering with work of the Correct (see page 2213) function; the True value indicates that it is possible to delete isolated nodes in order to perform structure correction.

#### Version

Available since version 8.5.

### II.2.2 CanExtendBars

#### C++

```
HRESULT get_CanExtendBars(VARIANT_BOOL* );
HRESULT put_CanExtendBars(VARIANT_BOOL);
```

#### C#

```
public bool CanExtendBars { get; set; }
```

#### Visual Basic

```
Public CanExtendBars As Boolean
```

#### Description

Flag managing operation of the Correct (see page 2213) function; True (default) value means that to carry out structure correction bar length may be increased.

#### Version

Available since version 3.5.

### II.2.3 CanUseRigidLinks

#### C++

```
HRESULT get_CanUseRigidLinks(VARIANT_BOOL* );
HRESULT put_CanUseRigidLinks(VARIANT_BOOL);
```

**C#**

```
public bool CanUseRigidLinks { get; set; }
```

**Visual Basic**

```
Public CanUseRigidLinks As Boolean
```

**Description**

Flag managing operation of the Correct (see page 2213) function; True (default) value means that to carry out structure correction rigid links may be defined.

**Version**

Available since version 3.5.

**II.2.4 Precision****C++**

```
HRESULT get_Precision(double* );
HRESULT put_Precision(double);
```

**C#**

```
public double Precision { get; set; }
```

**Visual Basic**

```
Public Precision As Double
```

**Description**

Precision applied by the Correct (see page 2213) function; setting the value (-1) indicates adoption of a default value .

**Version**

Available since version 3.5.

**II.3 IRobotStructureGeoAnalyser Methods**

The methods of the IRobotStructureGeoAnalyser class are listed here.

**Public Methods**

	<b>Name</b>	<b>Description</b>
💡	Correct (see page 2213)	Function carries out geometry correction for a whole structure or for a structure part based on the parameters set earlier. .
💡	DefineIntersection (see page 2213)	Function defines a relation of intersection between the specified bars. This information will be used by the function of structure geometry correction. .
💡	SetEdgeSize (see page 2213)	Function defines the "thickness" of a geometric object edge.
💡	SetNodeSize (see page 2214)	Function defines the "size" of a node. Structure elements that are located in a distance equal or less than the specified radius from the node will be treated as adjoining to the node.
💡	StartCollectInfo (see page 2214)	Function runs analysis of operations performed on the structure to make subsequent correction of its geometry easier. .
💡	StopCollectInfo (see page 2214)	

**II.3.1 Correct****C++**

```
HRESULT Correct(BSTR _bar_sel = "");
```

**C#**

```
public void Correct(string _bar_sel = "");
```

**Visual Basic**

```
Public Sub Correct(Optional _bar_sel As String = "")
```

**Description**

Function carries out geometry correction for a whole structure or for a structure part based on the parameters set earlier. .

**Version**

Available since version 3.5.

**II.3.2 DefineIntersection****C++**

```
HRESULT DefineIntersection(long _bar1, long _bar2, IRobotBarIntersectRelationship  
_intersect_rel);
```

**C#**

```
public void DefineIntersection(long _bar1, long _bar2, IRobotBarIntersectRelationship  
_intersect_rel);
```

**Visual Basic**

```
Public Sub DefineIntersection(_bar1 As long, _bar2 As long, _intersect_rel As  
IRobotBarIntersectRelationship)
```

**Description**

Function defines a relation of intersection between the specified bars. This information will be used by the function of structure geometry correction. .

**Version**

Available since version 3.5.

**II.3.3 SetEdgeSize****C++**

```
HRESULT SetEdgeSize(long _obj_num, double _radius);
```

**C#**

```
public void SetEdgeSize(long _obj_num, double _radius);
```

**Visual Basic**

```
Public Sub SetEdgeSize(_obj_num As long, _radius As double)
```

**Description**

Function defines the "thickness" of a geometric object edge.

**II.3.4 SetNodeSize****C++**

```
HRESULT SetNodeSize(long _node_num, double _radius);
```

**C#**

```
public void SetNodeSize(long _node_num, double _radius);
```

**Visual Basic**

```
Public Sub SetNodeSize(_node_num As long, _radius As double)
```

**Description**

Function defines the "size" of a node. Structure elements that are located in a distance equal or less than the specified radius from the node will be treated as adjoining to the node.

## II.3.5 StartCollectInfo

**C++**

```
HRESULT StartCollectInfo();
```

**C#**

```
public void StartCollectInfo();
```

**Visual Basic**

```
Public Sub StartCollectInfo()
```

**Description**

Function runs analysis of operations performed on the structure to make subsequent correction of its geometry easier. .

**Version**

Available since version 3.5.

## II.3.6 StopCollectInfo

**C++**

```
HRESULT StopCollectInfo();
```

**C#**

```
public void StopCollectInfo();
```

**Visual Basic**

```
Public Sub StopCollectInfo()
```

**Version**

Available since version 3.5.

# Index

\_VVST\_STRUCTURAL\_AXIS enumeration member 1399

## A

Add-ins Manager 1540  
Angle connection 2019  
Application - main object of the model 1527

## B

Backgrounds 1318  
Bars 533  
Basic definitions 2  
Beam-principal beam connection 2111  
Buckling analysis parameters 356

## C

Cables 534  
Calculation module 1552  
Calculation results for bars 864  
Calculation results for dynamic analyses 919  
Calculation results for finite elements 930  
Calculation results for nodes 847  
CB71 Code 1844  
Claddings 609  
Code combinations 147  
Column base connections 1960  
Compatible nodes 485  
Complex attributes of a bar 534  
Complex attributes of a node 475  
Complex attributes of an object 697  
Complex attributes shared by objects of different types 654  
Concrete 1656  
Concrete column base 1979  
Connection module 1929  
Continuous Footing 1163

## D

Data Server 1

Data structures containing complex calculation results 879  
Design of RC members 1779  
Direct Analysis Method 1587  
Drawing 1259

## E

EC3 Code 1800  
Edit operations 1042  
Elastic ground 537  
EPF\_HTML enumeration member 1510  
EPF\_MS\_OFFICE enumeration member 1510  
EPF\_OPEN\_OFFICE enumeration member 1510  
External parameters 2168

## F

FE mesh generator 804  
Finite elements 803  
Fixed column base 1970  
Footfall analysis parameters 395  
FRF analysis parameters 392

## G

Generating 3D loads 81  
Geometric data types 1601  
Geometrical imperfections 583  
Global data types 1600  
Graphical structure views 1372  
Grouping of objects 843  
Gusset plate connections 2048

## H

Harmonic analysis parameters 387

## I

I\_3PRV\_LOCAL\_SYSTEM enumeration member 44  
I\_3PRV\_N1 enumeration member 44  
I\_3PRV\_N2 enumeration member 44  
I\_3PRV\_N3 enumeration member 44  
I\_3PRV\_PROJECTION enumeration member 44  
I\_3PRV\_PX1 enumeration member 44  
I\_3PRV\_PX2 enumeration member 44

I_3PRV_PX3 enumeration member 44	I_ACI318_ST_75 enumeration member 1740
I_3PRV_PY1 enumeration member 44	I_AMT_DAM enumeration member 1351
I_3PRV_PY2 enumeration member 44	I_AMT_MAIN enumeration member 1351
I_3PRV_PY3 enumeration member 44	I_AST_COLUMN_CIRCULAR enumeration member 513
I_3PRV_PZ1 enumeration member 44	I_AST_COLUMN_RECTANGULAR enumeration member 513
I_3PRV_PZ2 enumeration member 44	I_AST_NODAL enumeration member 513
I_3PRV_PZ3 enumeration member 44	I_AST_WALL_CONCRETE enumeration member 513
I_3PRV_TX1 enumeration member 47	I_AST_WALL_MASONRY enumeration member 513
I_3PRV_TX2 enumeration member 47	I_BAM_BLOCK_SUBSPACE_ITERATION enumeration member 360
I_3PRV_TX3 enumeration member 47	I_BAM_SUBSPACE_ITERATION enumeration member 360
I_3PRV_TZ1 enumeration member 47	I_BBT_BIAXIAL enumeration member 1689
I_3PRV_TZ2 enumeration member 47	I_BBT_FALSE enumeration member 1689
I_3PRV_TZ3 enumeration member 47	I_BBT_SIMPLE enumeration member 1689
I_ACI318_BR_10 enumeration member 1742	I_BBT_UNIAXIAL enumeration member 1689
I_ACI318_BR_11 enumeration member 1742	I_BCAPT_ELONGATION_DL enumeration member 537
I_ACI318_BR_14 enumeration member 1742	I_BCAPT_ELONGATION_DL_RELATIVE enumeration member 537
I_ACI318_BR_18 enumeration member 1742	I_BCAPT_FORCE_FO enumeration member 537
I_ACI318_BR_3 enumeration member 1742	I_BCAPT_LENGTH_L enumeration member 537
I_ACI318_BR_4 enumeration member 1742	I_BCAPT_STRESSES_SIGMA enumeration member 537
I_ACI318_BR_5 enumeration member 1742	I_BCCT_MEMBER enumeration member 1687
I_ACI318_BR_6 enumeration member 1742	I_BCCT_PLATE enumeration member 1687
I_ACI318_BR_7 enumeration member 1742	I_BCE_BCKL_COLUMN enumeration member 1677
I_ACI318_BR_8 enumeration member 1742	I_BCE_NO_CALC enumeration member 1677
I_ACI318_BR_9 enumeration member 1742	I_BCE_NO_ERRORS enumeration member 1677
I_ACI318_LAPT_12 enumeration member 1740	I_BCE_SECTION_SHEAR enumeration member 1677
I_ACI318_LAPT_3 enumeration member 1740	I_BCE_SECTION_TOO_SMALL enumeration member 1677
I_ACI318_LAPT_6 enumeration member 1740	I_BCE_USER_ERROR enumeration member 1677
I_ACI318_LAPT_60 enumeration member 1740	I_BCPDDV_CONCRETE_CREEP enumeration member 1669
I_ACI318_MBD_10 enumeration member 1743	I_BCPDDV_CONCRETE_FC enumeration member 1669
I_ACI318_MBD_12 enumeration member 1743	I_BCPDDV_COVER_SIZE enumeration member 1669
I_ACI318_MBD_14 enumeration member 1743	I_BCPDDV_FLT2F enumeration member 1669
I_ACI318_MBD_16 enumeration member 1743	I_BCPDDV_LONG_COMPRESSION_BAR_DIAM enumeration member 1669
I_ACI318_MBD_20 enumeration member 1743	I_BCPDDV_LONG_STEEL_FE enumeration member 1669
I_ACI318_MBD_25 enumeration member 1743	I_BCPDDV_LONG_TENSION_BAR_DIAM enumeration member 1669
I_ACI318_MBD_32 enumeration member 1743	I_BCPDDV_MAIN_RNF_COORD_X enumeration member 1669
I_ACI318_MBD_40 enumeration member 1743	I_BCPDDV_MAIN_RNF_COORD_Y enumeration member 1669
I_ACI318_MBD_5 enumeration member 1743	I_BCPDDV_MAIN_RNF_COORD_Z enumeration member 1669
I_ACI318_MBD_6 enumeration member 1743	I_BCPDDV_MAX_BREAK enumeration member 1669
I_ACI318_MBD_8 enumeration member 1743	
I_ACI318_ST_40 enumeration member 1740	
I_ACI318_ST_50 enumeration member 1740	
I_ACI318_ST_60 enumeration member 1740	

I_BCPDDV_MAX_CRACK enumeration member 1669	I_BCT_RIGIDITY enumeration member 1688
I_BCPDDV_MAX_DEFLECTION enumeration member 1669	I_BCW_BREAK_MAX enumeration member 1678
I_BCPDDV_REDISTRIBUTION enumeration member 1669	I_BCW_CONCRETE_EXC_QPR enumeration member 1678
I_BCPDDV_SAND_CONTENT enumeration member 1669	I_BCW_CONCRETE_EXC_RAR enumeration member 1678
I_BCPDDV_TRAN_BAR_DIAM enumeration member 1669	I_BCW_COT_THETA_CHANGED enumeration member 1678
I_BCPDDV_TRAN_INCLINATION enumeration member 1669	I_BCW_LAMBDA_MAX enumeration member 1678
I_BCPDDV_TRAN_STEEL_FE enumeration member 1669	I_BCW_NO_WARNINGS enumeration member 1678
I_BCPDDV_USER_VALUE enumeration member 1669	I_BCW_RO_MAX enumeration member 1678
I_BCPDIV_ADJUST_DEFLECTION enumeration member 1670	I_BCW_RO_MIN enumeration member 1678
I_BCPDIV_CALC_CRACK enumeration member 1670	I_BCW_SPACE_BY_CODE enumeration member 1678
I_BCPDIV_CALC_DEFLECTION enumeration member 1670	I_BCW_STEEL_EXC_RAR enumeration member 1678
I_BCPDIV_CONCRETE_BY_CLASS enumeration member 1670	I_BCW_USER_WARNING enumeration member 1678
I_BCPDIV_CONCRETE_BY_CONSTRUCTION enumeration member 1670	I_BD_CB71_CANTILEVER enumeration member 1878
I_BCPDIV_CONCRETE_LIGHT enumeration member 1670	I_BD_CB71_INTERNAL_BRACINGS enumeration member 1878
I_BCPDIV_CRACKING enumeration member 1670	I_BD_CB71_NONE enumeration member 1878
I_BCPDIV_FLT2F enumeration member 1670	I_BD_CB71_PINNED_PINNED_1_0 enumeration member 1878
I_BCPDIV_LEGS_NUM enumeration member 1670	I_BD_CB71_STIFF_PINNED_0_8 enumeration member 1878
I_BCPDIV_LONG_BAR_DIAM_THE_SAME enumeration member 1670	I_BD_CB71_STIFF_STIFF_0_65 enumeration member 1878
I_BCPDIV_LONG_STEEL_BY_CLASS enumeration member 1670	I_BD_X enumeration member 1689
I_BCPDIV_MAIN_RNF_COORD_SYSTEM enumeration member 1670	I_BD_Y enumeration member 1689
I_BCPDIV_SEISMICS enumeration member 1670	I_BD_Z enumeration member 1689
I_BCPDIV_TRAN_STEEL_BY_CLASS enumeration member 1670	I_BDPDV_AS enumeration member 1689
I_BCPDIV_TRAN_TYPE enumeration member 1670	I_BDPDV_AS_BOTTOM enumeration member 1689
I_BCPDIV_TRAN_ZONES_NUM enumeration member 1670	I_BDPDV_AS_LEFT enumeration member 1689
I_BCPDIV_TRAN_ZONES_OPTIMALIZATION enumeration member 1670	I_BDPDV_AS_RIGHT enumeration member 1689
I_BCPDIV_USER_VALUE enumeration member 1670	I_BDPDV_AS_TOP enumeration member 1689
I_BCPDSV_CONCRETE_CLASS enumeration member 1671	I_BDPDV_CONCRETE_FC enumeration member 1689
I_BCPDSV_LONG_REINF_CLASS enumeration member 1671	I_BDPDV_DIR_ALPHA enumeration member 1689
I_BCPDSV_NAME enumeration member 1671	I_BDPDV_DIR_R enumeration member 1689
I_BCPDSV_TRAN_REINF_CLASS enumeration member 1671	I_BDPDV_DIR_THETA enumeration member 1689
I_BCPDSV_USER_VALUE enumeration member 1671	I_BDPDV_DIR_X enumeration member 1689
I_BCS_CARTESIAN enumeration member 1690	I_BDPDV_DIR_Y enumeration member 1689
I_BCS_POLAR enumeration member 1690	I_BDPDV_DIR_Z enumeration member 1689
I_BCT_BAR_SPACING enumeration member 1688	I_BDPIV_BEND enumeration member 1681
I_BCT_CRACK_WIDTH enumeration member 1688	I_BDPIV_BEND_BIAXIAL enumeration member 1681
I_BCT_DIMENSIONING enumeration member 1688	I_BDPIV_BEND_SIMPLE enumeration member 1681
	I_BDPIV_BEND_UNIAXIAL enumeration member 1681
	I_BDPIV_CALCULATION_TYPE enumeration member 1681
	I_BDPIV_DIR_Y enumeration member 1681
	I_BDPIV_DIR_Z enumeration member 1681
	I_BDPIV_DIRECTION enumeration member 1681

I_BDPIV_LEVEL enumeration member 1681	I_BFCRV_OFFSET_Z enumeration member 41
I_BDPIV_SHEAR enumeration member 1681	I_BFCRV_REL enumeration member 41
I_BDPIV_TORSION enumeration member 1681	I_BFCRV_X enumeration member 41
I_BDPIV_USER_VALUE enumeration member 1681	I_BFDDV_COEF_LONG enumeration member 1690
I_BDRV_DIL enumeration member 40	I_BFDDV_COEF_LONG2TOT enumeration member 1690
I_BDRV_ENTIRE_STRUCTURE enumeration member 43	I_BFDIV_SLS_COMB_TYPE enumeration member 1690
I_BDRV_REL enumeration member 40	I_BFDSLS_COMB_TYPE_FRE enumeration member 1691
I_BDRV_X enumeration member 43	I_BFDSLS_COMB_TYPE_NONE enumeration member 1691
I_BDRV_Y enumeration member 43	I_BFDSLS_COMB_TYPE_QPR enumeration member 1691
I_BDRV_Z enumeration member 43	I_BFDSLS_COMB_TYPE_RAR enumeration member 1691
I_BEBDV_HEIGHT enumeration member 615	I_BGIA_Y enumeration member 587
I_BEBDV_LENGTH enumeration member 615	I_BGIA_Z enumeration member 587
I_BEBDV_THICKNESS_1 enumeration member 615	I_BIR_INTERSECT_END_END enumeration member 2209
I_BEBDV_THICKNESS_2 enumeration member 615	I_BIR_INTERSECT_END_MID enumeration member 2209
I_BEBDV_WIDTH enumeration member 615	I_BIR_INTERSECT_MID_END enumeration member 2209
I_BEBT_PLATES enumeration member 615	I_BIR_INTERSECT_MID_MID enumeration member 2209
I_BEBT_SECTION enumeration member 615	I_BIR_NOT_INTERSECT enumeration member 2209
I_BERV_ELASTIC enumeration member 575	I_BIR_UNKNOWN enumeration member 2209
I_BERV_ELASTIC_MINUS enumeration member 575	I_BL_DOWN enumeration member 1689
I_BERV_ELASTIC_PLUS enumeration member 575	I_BL_UP enumeration member 1689
I_BERV_ELASTIC_REDUCED enumeration member 575	I_BLN_BEAM_NATURE_DEAD enumeration member 1159
I_BERV_ELASTIC_REDUCED_MINUS enumeration member 575	I_BLN_BEAM_NATURE_LIVE enumeration member 1159
I_BERV_ELASTIC_REDUCED_PLUS enumeration member 575	I_BLN_BEAM_NATURE_SEISMIC enumeration member 1159
I_BERV_MINUS enumeration member 575	I_BLN_BEAM_NATURE_SNOW enumeration member 1159
I_BERV_NONE enumeration member 575	I_BLN_BEAM_NATURE_WIND enumeration member 1159
I_BERV_NONLINEAR enumeration member 575	I_BMDDV_BUCK_COEF_Y enumeration member 1674
I_BERV_PLUS enumeration member 575	I_BMDDV_BUCK_COEF_Z enumeration member 1674
I_BERV_STD enumeration member 575	I_BMDDV_LENGTH_Y enumeration member 1674
I_BFCRV_ALPHA enumeration member 41	I_BMDDV_LENGTH_Z enumeration member 1674
I_BFCRV_BETA enumeration member 41	I_BMDDV_SUPPORT_WIDTH_1 enumeration member 1674
I_BFCRV_CX enumeration member 41	I_BMDDV_SUPPORT_WIDTH_2 enumeration member 1674
I_BFCRV_CY enumeration member 41	I_BMDDV_USER_VALUE enumeration member 1674
I_BFCRV_CZ enumeration member 41	I_BMDIV_CALC_AS_SLENDER_Y enumeration member 1675
I_BFCRV_FX enumeration member 41	I_BMDIV_CALC_AS_SLENDER_Z enumeration member 1675
I_BFCRV_FY enumeration member 41	I_BMDIV_SWAY_Y enumeration member 1675
I_BFCRV_FZ enumeration member 41	I_BMDIV_SWAY_Z enumeration member 1675
I_BFCRV_GAMMA enumeration member 41	I_BMDIV_TYPE enumeration member 1675
I_BFCRV_GENERATE_CALC_NODE enumeration member 41	I_BMDIV_USER_VALUE enumeration member 1675
I_BFCRV_LOC enumeration member 41	I_BMDRV_ALPHA enumeration member 41
I_BFCRV_OFFSET_Y enumeration member 41	I_BMDRV_BETA enumeration member 41

I_BMDRV_GAMMA enumeration member 41	1676
I_BMDRV_MX enumeration member 41	I_BRDV_LAMBDA_Y enumeration member 1676
I_BMDRV_MY enumeration member 41	I_BRDV_LAMBDA_Z enumeration member 1676
I_BMDRV_MZ enumeration member 41	I_BRDV_RIGIDITY enumeration member 1676
I_BMDSV_USER_VALUE enumeration member 1685	I_BRDV_RO enumeration member 1676
I_BMT_BEAM enumeration member 1685	I_BRDV_RO_MAX enumeration member 1676
I_BMT_COLUMN enumeration member 1685	I_BRDV_RO_MIN enumeration member 1676
I_BMT_PLATE enumeration member 1685	I_BRDV_STIFFNESS_BOTTOM enumeration member 1676
I_BOAP_0_0 enumeration member 618	I_BRDV_STIFFNESS_TOP enumeration member 1676
I_BOAP_0_VPZ enumeration member 618	I_BRDV_STIRR_DENSITY enumeration member 1676
I_BOAP_0_VZ enumeration member 618	I_BRDV_STIRR_SPACE enumeration member 1676
I_BOAP_VPY_0 enumeration member 618	I_BRDV_STIRR_SPACE_MAX enumeration member 1676
I_BOAP_VPY_VPZ enumeration member 618	I_BRDV_STIRR_SPACE_MIN enumeration member 1676
I_BOAP_VPY_VZ enumeration member 618	I_BRDV_STIRR_SPACE_MOD enumeration member 1676
I_BOAP_VY_0 enumeration member 618	I_BRDV_USER_VALUE enumeration member 1676
I_BOAP_VY_VPZ enumeration member 618	I_BRIV_BEND_CASE_IDX enumeration member 1677
I_BOAP_VY_VZ enumeration member 618	I_BRIV_BEND_ERROR_NUM enumeration member 1677
I_BOML_IN_THE_AXIS enumeration member 618	I_BRIV_BEND_WARNING_NUM enumeration member 1677
I_BOML_INCREASE_LENGTH enumeration member 618	I_BRIV_SHEAR_CASE_IDX enumeration member 1677
I_BOML_REDUCE_LENGTH enumeration member 618	I_BRIV_SHEAR_ERROR_NUM enumeration member 1677
I_BRDV_AS_BOTTOM enumeration member 1676	I_BRIV_SHEAR_WARNING_NUM enumeration member 1677
I_BRDV_AS_LEFT enumeration member 1676	I_BRIV_TORSION_CASE_IDX enumeration member 1677
I_BRDV_AS_MAX enumeration member 1676	I_BRIV_TORSION_ERROR_NUM enumeration member 1677
I_BRDV_AS_MIN enumeration member 1676	I_BRIV_TORSION_WARNING_NUM enumeration member 1677
I_BRDV_AS_MIN_BOTTOM enumeration member 1676	I_BRIV_USER_VALUE enumeration member 1677
I_BRDV_AS_MIN_TOP enumeration member 1676	I_BRSV_BEND_ERROR_NAME enumeration member 1678
I_BRDV_AS_PROVIDED_BOTTOM_MY enumeration member 1676	I_BRSV_BEND_WARNING_NAME enumeration member 1678
I_BRDV_AS_PROVIDED_BOTTOM_MZ enumeration member 1676	I_BRSV_SHEAR_ERROR_NAME enumeration member 1678
I_BRDV_AS_PROVIDED_TOP_MY enumeration member 1676	I_BRSV_SHEAR_WARNING_NAME enumeration member 1678
I_BRDV_AS_PROVIDED_TOP_MZ enumeration member 1676	I_BRSV_TORSION_ERROR_NAME enumeration member 1678
I_BRDV_AS_RIGHT enumeration member 1676	I_BRSV_TORSION_WARNING_NAME enumeration member 1678
I_BRDV_AS_TOP enumeration member 1676	I_BRSV_USER_VALUE enumeration member 1678
I_BRDV_BAR_SPACING enumeration member 1676	I_BS8110_CA_1 enumeration member 1750
I_BRDV_CRACK_WIDTH enumeration member 1676	I_BS8110_CA_28 enumeration member 1750
I_BRDV_CRACK_WIDTH_LT_BOTTOM enumeration member 1676	I_BS8110_CA_3 enumeration member 1750
I_BRDV_CRACK_WIDTH_LT_TOP enumeration member 1676	I_BS8110_CA_365 enumeration member 1750
I_BRDV_CRACK_WIDTH_ST_BOTTOM enumeration member 1676	I_BS8110_CA_7 enumeration member 1750
I_BRDV_CRACK_WIDTH_ST_TOP enumeration member	I_BS8110_CA_90 enumeration member 1750
	I_BS8110(CG_AUTO enumeration member 1751

I\_BS8110(CG\_C12\_15 enumeration member 1751  
I\_BS8110(CG\_C16\_20 enumeration member 1751  
I\_BS8110(CG\_C20\_25 enumeration member 1751  
I\_BS8110(CG\_C25\_30 enumeration member 1751  
I\_BS8110(CG\_C30\_37 enumeration member 1751  
I\_BS8110(CG\_C35\_45 enumeration member 1751  
I\_BS8110(CG\_C40\_50 enumeration member 1751  
I\_BS8110(CG\_C45\_55 enumeration member 1751  
I\_BS8110(CG\_C50\_60 enumeration member 1751  
I\_BS8110(ER\_MILD enumeration member 1750  
I\_BS8110(ER\_MODERATE enumeration member 1750  
I\_BS8110(ER\_MOST\_SEVERE enumeration member 1750  
I\_BS8110(ER\_SEVERE enumeration member 1750  
I\_BS8110(ER VERY\_SEVERE enumeration member 1750  
I\_BS8110(PSF\_1985 enumeration member 1751  
I\_BS8110(PSF\_1997 enumeration member 1751  
I\_BSCCP\_LEFT enumeration member 547  
I\_BSCCP\_NONE enumeration member 547  
I\_BSCCP\_RIGHT enumeration member 547  
I\_BSCDV\_BEAM\_B enumeration member 546  
I\_BSCDV\_BEAM\_BG1 enumeration member 546  
I\_BSCDV\_BEAM\_BG2 enumeration member 546  
I\_BSCDV\_BEAM\_EL1 enumeration member 546  
I\_BSCDV\_BEAM\_EL2 enumeration member 546  
I\_BSCDV\_BEAM\_ER1 enumeration member 546  
I\_BSCDV\_BEAM\_ER2 enumeration member 546  
I\_BSCDV\_BEAM\_H enumeration member 546  
I\_BSCDV\_BEAM\_HG1 enumeration member 546  
I\_BSCDV\_BEAM\_HG2 enumeration member 546  
I\_BSCDV\_BEAM\_I\_B enumeration member 546  
I\_BSCDV\_BEAM\_I\_B1 enumeration member 546  
I\_BSCDV\_BEAM\_I\_B2 enumeration member 546  
I\_BSCDV\_BEAM\_I\_H enumeration member 546  
I\_BSCDV\_BEAM\_I\_HF1 enumeration member 546  
I\_BSCDV\_BEAM\_I\_HF2 enumeration member 546  
I\_BSCDV\_BEAM\_PL enumeration member 546  
I\_BSCDV\_BEAM\_PR enumeration member 546  
I\_BSCDV\_BEAM\_RECT\_B enumeration member 546  
I\_BSCDV\_BEAM\_RECT\_H enumeration member 546  
I\_BSCDV\_BEAM\_RL enumeration member 546  
I\_BSCDV\_BEAM\_RR enumeration member 546  
I\_BSCDV\_BEAM\_T\_B enumeration member 546  
I\_BSCDV\_BEAM\_T\_BF enumeration member 546  
I\_BSCDV\_BEAM\_T\_H enumeration member 546  
I\_BSCDV\_BEAM\_T\_HF enumeration member 546  
I\_BSCDV\_COL\_B enumeration member 546  
I\_BSCDV\_COL\_DE enumeration member 546  
I\_BSCDV\_COL\_H enumeration member 546  
I\_BSCDV\_COL\_H1 enumeration member 546  
I\_BSCDV\_COL\_H2 enumeration member 546  
I\_BSCDV\_COL\_L1 enumeration member 546  
I\_BSCDV\_COL\_L2 enumeration member 546  
I\_BSCDV\_COL\_N enumeration member 546  
I\_BSCS\_C enumeration member 571  
I\_BSCS\_I enumeration member 571  
I\_BSCS\_L enumeration member 571  
I\_BSCS\_UNDEFINED enumeration member 571  
I\_BSD\_BEAM\_SECTION\_DIM\_B enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_BG1 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_BG2 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_BG3 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_BG4 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_CL enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_CR enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_EL1 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_EL2 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_ER1 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_ER2 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_H1 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_H2 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_HG1 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_HG2 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_HG3 enumeration member 1161  
I\_BSD\_BEAM\_SECTION\_DIM\_HG4 enumeration member 1161

I_BSD_BEAM_SECTION_DIM_L enumeration member 1161	I_BSDV_DIM1 enumeration member 544
I_BSD_BEAM_SECTION_DIM_OFFY1 enumeration member 1161	I_BSDV_DIM2 enumeration member 544
I_BSD_BEAM_SECTION_DIM_OFFY2 enumeration member 1161	I_BSDV_DIM3 enumeration member 544
I_BSD_BEAM_SECTION_DIM_OFFZ1 enumeration member 1161	I_BSDV_GAMMA enumeration member 544
I_BSD_BEAM_SECTION_DIM_OFFZ2 enumeration member 1161	I_BSDV_IOMEGA enumeration member 544
I_BSD_BEAM_SECTION_DIM_PL enumeration member 1161	I_BSDV_IX enumeration member 544
I_BSD_BEAM_SECTION_DIM_PL_CALC enumeration member 1161	I_BSDV_IY enumeration member 544
I_BSD_BEAM_SECTION_DIM_PR enumeration member 1161	I_BSDV_IZ enumeration member 544
I_BSD_BEAM_SECTION_DIM_PR_CALC enumeration member 1161	I_BSDV_P1_LENGTH enumeration member 544
I_BSD_BEAM_SECTION_DIM_RL enumeration member 1161	I_BSDV_P1_THICKNESS enumeration member 544
I_BSD_BEAM_SECTION_DIM_RR enumeration member 1161	I_BSDV_P2_LENGTH enumeration member 544
I_BSD_CFOOT_SECTION_DIM_B enumeration member 1178	I_BSDV_P2_THICKNESS enumeration member 544
I_BSD_CFOOT_SECTION_DIM_EL1 enumeration member 1178	I_BSDV_P3_LENGTH enumeration member 544
I_BSD_CFOOT_SECTION_DIM_ER1 enumeration member 1178	I_BSDV_P3_THICKNESS enumeration member 544
I_BSD_CFOOT_SECTION_DIM_H1 enumeration member 1178	I_BSDV_P4_LENGTH enumeration member 544
I_BSD_CFOOT_SECTION_DIM_H2 enumeration member 1178	I_BSDV_P4_THICKNESS enumeration member 544
I_BSD_CFOOT_SECTION_DIM_L enumeration member 1178	I_BSDV_RA enumeration member 544
I_BSD_CFOOT_SECTION_DIM_PL enumeration member 1178	I_BSDV_RI enumeration member 544
I_BSD_CFOOT_SECTION_DIM_PR enumeration member 1178	I_BSDV_S enumeration member 544
I_BSD_CFOOT_SECTION_DIM_RL enumeration member 1178	I_BSDV_SURFACE enumeration member 544
I_BSD_CFOOT_SECTION_DIM_RR enumeration member 1178	I_BSDV_TF enumeration member 544
I_BSDV_ANGLE1 enumeration member 544	I_BSDV_TF2 enumeration member 544
I_BSDV_ANGLE2 enumeration member 544	I_BSDV_TW enumeration member 544
I_BSDV_AX enumeration member 544	I_BSDV_VPY enumeration member 544
I_BSDV_AY enumeration member 544	I_BSDV_VPZ enumeration member 544
I_BSDV_AZ enumeration member 544	I_BSDV_VY enumeration member 544
I_BSDV_BF enumeration member 544	I_BSDV_VZ enumeration member 544
I_BSDV_BF2 enumeration member 544	I_BSDV_WEIGHT enumeration member 544
I_BSDV_D enumeration member 544	I_BSDV_WX enumeration member 544
	I_BSDV_WY enumeration member 544
	I_BSDV_WZ enumeration member 544
	I_BSDV_ZY enumeration member 544
	I_BSDV_ZZ enumeration member 544
	I_BSM_BEAM_SUPPORT_CONCRETE enumeration member 1162
	I_BSM_BEAM_SUPPORT_MASONRY enumeration member 1162
	I_BSM_CFOOT_SUPPORT_CONCRETE enumeration member 1178
	I_BSM_CFOOT_SUPPORT_MASONRY enumeration member 1178
	I_BSNDV_BOX_2_B enumeration member 546
	I_BSNDV_BOX_2_B1 enumeration member 546
	I_BSNDV_BOX_2_H enumeration member 546

I\_BSNDV\_BOX\_2\_TF enumeration member 546  
I\_BSNDV\_BOX\_2\_TW enumeration member 546  
I\_BSNDV\_BOX\_3\_B enumeration member 546  
I\_BSNDV\_BOX\_3\_B1 enumeration member 546  
I\_BSNDV\_BOX\_3\_B2 enumeration member 546  
I\_BSNDV\_BOX\_3\_H enumeration member 546  
I\_BSNDV\_BOX\_3\_TF enumeration member 546  
I\_BSNDV\_BOX\_3\_TF2 enumeration member 546  
I\_BSNDV\_BOX\_3\_TW enumeration member 546  
I\_BSNDV\_BOX\_B enumeration member 546  
I\_BSNDV\_BOX\_H enumeration member 546  
I\_BSNDV\_BOX\_TF enumeration member 546  
I\_BSNDV\_BOX\_TW enumeration member 546  
I\_BSNDV\_C\_B enumeration member 546  
I\_BSNDV\_C\_H enumeration member 546  
I\_BSNDV\_C\_TF enumeration member 546  
I\_BSNDV\_C\_TW enumeration member 546  
I\_BSNDV\_CROSS\_P1\_L enumeration member 546  
I\_BSNDV\_CROSS\_P1\_T enumeration member 546  
I\_BSNDV\_CROSS\_P2\_L enumeration member 546  
I\_BSNDV\_CROSS\_P2\_T enumeration member 546  
I\_BSNDV\_CROSS\_P3\_L enumeration member 546  
I\_BSNDV\_CROSS\_P3\_T enumeration member 546  
I\_BSNDV\_CROSS\_P4\_L enumeration member 546  
I\_BSNDV\_CROSS\_P4\_T enumeration member 546  
I\_BSNDV\_DRECT\_B enumeration member 546  
I\_BSNDV\_DRECT\_D enumeration member 546  
I\_BSNDV\_DRECT\_H enumeration member 546  
I\_BSNDV\_H\_B enumeration member 546  
I\_BSNDV\_H\_B1 enumeration member 546  
I\_BSNDV\_H\_H enumeration member 546  
I\_BSNDV\_H\_TF enumeration member 546  
I\_BSNDV\_H\_TW enumeration member 546  
I\_BSNDV\_HOLE\_B enumeration member 546  
I\_BSNDV\_HOLE\_H enumeration member 546  
I\_BSNDV\_HOLE\_X enumeration member 546  
I\_BSNDV\_HOLE\_Z enumeration member 546  
I\_BSNDV\_I\_B enumeration member 546  
I\_BSNDV\_I\_H enumeration member 546  
I\_BSNDV\_I\_TF enumeration member 546  
I\_BSNDV\_I\_TW enumeration member 546  
I\_BSNDV\_II\_B1 enumeration member 546  
I\_BSNDV\_II\_B2 enumeration member 546  
I\_BSNDV\_II\_H enumeration member 546  
I\_BSNDV\_II\_TF1 enumeration member 546  
I\_BSNDV\_II\_TF2 enumeration member 546  
I\_BSNDV\_II\_TW enumeration member 546  
I\_BSNDV\_L\_B enumeration member 546  
I\_BSNDV\_L\_H enumeration member 546  
I\_BSNDV\_L\_TF enumeration member 546  
I\_BSNDV\_L\_TW enumeration member 546  
I\_BSNDV\_POLYGONAL\_D enumeration member 546  
I\_BSNDV\_POLYGONAL\_D\_IS\_INT enumeration member 546  
I\_BSNDV\_POLYGONAL\_N enumeration member 546  
I\_BSNDV\_POLYGONAL\_T enumeration member 546  
I\_BSNDV\_RECT\_B enumeration member 546  
I\_BSNDV\_RECT\_H enumeration member 546  
I\_BSNDV\_RECT\_T enumeration member 546  
I\_BSNDV\_T\_B enumeration member 546  
I\_BSNDV\_T\_H enumeration member 546  
I\_BSNDV\_T\_TF enumeration member 546  
I\_BSNDV\_T\_TW enumeration member 546  
I\_BSNDV\_TUBE\_D enumeration member 546  
I\_BSNDV\_TUBE\_T enumeration member 546  
I\_BSNDV\_XI\_B enumeration member 546  
I\_BSNDV\_XI\_H enumeration member 546  
I\_BSNDV\_XI\_H1 enumeration member 546  
I\_BSNDV\_XI\_TF enumeration member 546  
I\_BSNDV\_XI\_TW enumeration member 546  
I\_BSNDV\_XT\_B enumeration member 546  
I\_BSNDV\_XT\_H enumeration member 546  
I\_BSNDV\_XT\_H1 enumeration member 546  
I\_BSNDV\_XT\_TF enumeration member 546  
I\_BSNDV\_XT\_TW enumeration member 546  
I\_BSNDV\_Z\_B enumeration member 546  
I\_BSNDV\_Z\_H enumeration member 546  
I\_BSNDV\_Z\_TF enumeration member 546  
I\_BSNDV\_Z\_TW enumeration member 546  
I\_BSSDV\_A enumeration member 574  
I\_BSSDV\_B1 enumeration member 574  
I\_BSSDV\_B2 enumeration member 574  
I\_BSSDV\_BP enumeration member 574  
I\_BSSDV\_C enumeration member 574  
I\_BSSDV\_D enumeration member 574

I_BSST_COMP_4L_FACE_WELD enumeration member 542
I_BSST_COMP_CI enumeration member 542
I_BSST_COMP_CI_BACK enumeration member 542
I_BSST_COMP_CI_BACK_WELD enumeration member 542
I_BSST_COMP_CI_WELD enumeration member 542
I_BSST_CONCR_BEAM enumeration member 542
I_BSST_CONCR_BEAM_I enumeration member 542
I_BSST_CONCR_BEAM_RECT enumeration member 542
I_BSST_CONCR_BEAM_T enumeration member 542
I_BSST_CONCR_COL_C enumeration member 542
I_BSST_CONCR_COL_CH enumeration member 542
I_BSST_CONCR_COL_CQ enumeration member 542
I_BSST_CONCR_COL_L enumeration member 542
I_BSST_CONCR_COL_P enumeration member 542
I_BSST_CONCR_COL_R enumeration member 542
I_BSST_CONCR_COL_T enumeration member 542
I_BSST_CONCR_COL_Z enumeration member 542
I_BSST_CUAP enumeration member 542
I_BSST_DCEC enumeration member 542
I_BSST_DCED enumeration member 542
I_BSST_DCEP enumeration member 542
I_BSST_DCIG enumeration member 542
I_BSST_DCIP enumeration member 542
I_BSST_HEA enumeration member 542
I_BSST_HEAA enumeration member 542
I_BSST_HEB enumeration member 542
I_BSST_HEC enumeration member 542
I_BSST_HEM enumeration member 542
I_BSST_HER enumeration member 542
I_BSST_HHEA enumeration member 542
I_BSST_HHEB enumeration member 542
I_BSST_HHEM enumeration member 542
I_BSST_IIPE enumeration member 542
I_BSST_IPE enumeration member 542
I_BSST_IPEA enumeration member 542
I_BSST_IPEO enumeration member 542
I_BSST_IPER enumeration member 542
I_BSST_IPEV enumeration member 542
I_BSST_IPN enumeration member 542
I_BSST_JOIST_BG enumeration member 542
I_BSST_JOIST_DLH enumeration member 542
I_BSST_JOIST_G enumeration member 542
I_BSSDV_H enumeration member 574
I_BSSDV_HS enumeration member 574
I_BSSDV_TF1 enumeration member 574
I_BSSDV_TF2 enumeration member 574
I_BSSDV_TP enumeration member 574
I_BSSDV_TW enumeration member 574
I_BSSDV_W enumeration member 574
I_BSST_CAE enumeration member 542
I_BSST_CAEP enumeration member 542
I_BSST_CAI enumeration member 542
I_BSST_CAIP enumeration member 542
I_BSST_CCL enumeration member 542
I_BSST_CIRC_FILLED enumeration member 542
I_BSST_COLD_C_PLUS enumeration member 542
I_BSST_COLD_L_LIPS enumeration member 542
I_BSST_COLD_SIGMA enumeration member 542
I_BSST_COLD_SIGMA_SL enumeration member 542
I_BSST_COLD_Z enumeration member 542
I_BSST_COLD_Z_ROT enumeration member 542
I_BSST_COMP_2C_BACK enumeration member 542
I_BSST_COMP_2C_BACK_WELD enumeration member 542
I_BSST_COMP_2C_FACE enumeration member 542
I_BSST_COMP_2C_FACE_WELD enumeration member 542
I_BSST_COMP_2I enumeration member 542
I_BSST_COMP_2I_WELD enumeration member 542
I_BSST_COMP_2L_CROSS enumeration member 542
I_BSST_COMP_2L_CROSS_WELD enumeration member 542
I_BSST_COMP_2L_FACE_LONG enumeration member 542
I_BSST_COMP_2L_FACE_LONG_WELD enumeration member 542
I_BSST_COMP_2L_FACE_SHORT enumeration member 542
I_BSST_COMP_2L_FACE_SHORT_WELD enumeration member 542
I_BSST_COMP_2L_LONG enumeration member 542
I_BSST_COMP_2L_LONG_WELD enumeration member 542
I_BSST_COMP_2L_SHORT enumeration member 542
I_BSST_COMP_2L_SHORT_WELD enumeration member 542
I_BSST_COMP_2LI enumeration member 542
I_BSST_COMP_2LI_WELD enumeration member 542
I_BSST_COMP_4L_BACK enumeration member 542
I_BSST_COMP_4L_BACK_WELD enumeration member 542
I_BSST_COMP_4L_FACE enumeration member 542

I_BSST_JOIST_K enumeration member 542	I_BSST_USER_RECT enumeration member 542
I_BSST_JOIST_KCS enumeration member 542	I_BSST_USER_T_SHAPE enumeration member 542
I_BSST_JOIST_LH enumeration member 542	I_BSST_USER_TUBE enumeration member 542
I_BSST_JOIST_SLH enumeration member 542	I_BSST_UUAP enumeration member 542
I_BSST_JOIST_VG enumeration member 542	I_BSST_UUPN enumeration member 542
I_BSST_MHEA enumeration member 542	I_BSST_WOOD_CIRC enumeration member 542
I_BSST_MHEB enumeration member 542	I_BSST_WOOD_DRECT enumeration member 542
I_BSST_MHEM enumeration member 542	I_BSST_WOOD_RECT enumeration member 542
I_BSST_MIPE enumeration member 542	I_BST_COMPLEX enumeration member 545
I_BSST_PRS enumeration member 542	I_BST_JOIST enumeration member 545
I_BSST_RECT_FILLED enumeration member 542	I_BST_NS_BOX enumeration member 545
I_BSST_SPEC_CASTELLATED_WEB_HEXAGONAL_OPENI NGS enumeration member 542	I_BST_NS_BOX_2 enumeration member 545
I_BSST_SPEC_CASTELLATED_WEB_HEXAGONAL_OPENI NGS_SHIFTED enumeration member 542	I_BST_NS_BOX_3 enumeration member 545
I_BSST_SPEC_CASTELLATED_WEB_ROUND_OPENINGS enumeration member 542	I_BST_NS_C enumeration member 545
I_BSST_SPEC_CORRUGATED_WEB enumeration member 542	I_BST_NS_CROSS enumeration member 545
I_BSST_SPEC_IFBA enumeration member 542	I_BST_NS_DRECT enumeration member 545
I_BSST_SPEC_IFBB enumeration member 542	I_BST_NS_H enumeration member 545
I_BSST_SPEC_SFB enumeration member 542	I_BST_NS_HOLE enumeration member 545
I_BSST_TCAR enumeration member 542	I_BST_NS_I enumeration member 545
I_BSST_TEAE enumeration member 542	I_BST_NS_II enumeration member 545
I_BSST_TEAI enumeration member 542	I_BST_NS_L enumeration member 545
I_BSST_THEX enumeration member 542	I_BST_NS_LP enumeration member 545
I_BSST_TREC enumeration member 542	I_BST_NS_POLYGONAL enumeration member 545
I_BSST_TRND enumeration member 542	I_BST_NS_RECT enumeration member 545
I_BSST_TRON enumeration member 542	I_BST_NS_T enumeration member 545
I_BSST_UAP enumeration member 542	I_BST_NS_TUBE enumeration member 545
I_BSST_UNKNOWN enumeration member 542	I_BST_NS_XI enumeration member 545
I_BSST_UPAF enumeration member 542	I_BST_NS_XT enumeration member 545
I_BSST_UPN enumeration member 542	I_BST_NS_Z enumeration member 545
I_BSST_URND enumeration member 542	I_BST_NS_ZP enumeration member 545
I_BSST_USER_BOX enumeration member 542	I_BST_SPECIAL enumeration member 545
I_BSST_USER_BOX_2 enumeration member 542	I_BST_STANDARD enumeration member 545
I_BSST_USER_BOX_3 enumeration member 542	I_BTC_COMPRESSION_ONLY enumeration member 646
I_BSST_USER_C_SHAPE enumeration member 542	I_BTC_STANDARD enumeration member 646
I_BSST_USER_CIRC_FILLED enumeration member 542	I_BTRV_ALPHA enumeration member 42
I_BSST_USER_CROSS enumeration member 542	I_BTRV_BETA enumeration member 42
I_BSST_USER_I_BISYM enumeration member 542	I_BTRV_GAMMA enumeration member 42
I_BSST_USER_I_MONOSYM enumeration member 542	I_BTRV_LOCAL enumeration member 42
I_BSST_USER_POLYGONAL enumeration member 542	I_BTRV_PROJECTION enumeration member 42
	I_BTRV_PX1 enumeration member 42
	I_BTRV_PX2 enumeration member 42

I_BTRV_PY1 enumeration member 42	I_CAT_PUSH_OVER enumeration member 407
I_BTRV_PY2 enumeration member 42	I_CAT_STATIC_BUCKLING enumeration member 407
I_BTRV_PZ1 enumeration member 42	I_CAT_STATIC_ELF_SEISMIC enumeration member 407
I_BTRV_PZ2 enumeration member 42	I_CAT_STATIC_LINEAR enumeration member 407
I_BTRV_RELATIVE enumeration member 42	I_CAT_STATIC_LINEAR_AUXILIARY enumeration member 407
I_BTRV_TX enumeration member 43	I_CAT_STATIC_NONLINEAR enumeration member 407
I_BTRV_TY enumeration member 43	I_CAT_STATIC_NONLINEAR_BUCKLING enumeration member 407
I_BTRV_TZ enumeration member 43	I_CAT_TIME_HISTORY enumeration member 407
I_BTRV_X1 enumeration member 42	I_CBAELCG_C12_15 enumeration member 1731
I_BTRV_X2 enumeration member 42	I_CBAELCG_C16_20 enumeration member 1731
I_BURV_ALPHA enumeration member 41	I_CBAELCG_C20_25 enumeration member 1731
I_BURV_BETA enumeration member 41	I_CBAELCG_C25_30 enumeration member 1731
I_BURV_GAMMA enumeration member 41	I_CBAELCG_C30_37 enumeration member 1731
I_BURV_LOCAL enumeration member 41	I_CBAELCG_C35_45 enumeration member 1731
I_BURV_OFFSET_Y enumeration member 41	I_CBAELCG_C40_50 enumeration member 1731
I_BURV_OFFSET_Z enumeration member 41	I_CBAELCG_C45_55 enumeration member 1731
I_BURV_PROJECTION enumeration member 41	I_CBAELCG_C50_60 enumeration member 1731
I_BURV_PX enumeration member 41	I_CBAELCG_OTHER enumeration member 1731
I_BURV_PY enumeration member 41	I_CBAELCT_NOT_SEVERE enumeration member 1730
I_BURV_PZ enumeration member 41	I_CBAELCT_OTHER enumeration member 1730
I_BURV_RELATIVE enumeration member 41	I_CBAELCT_SEVERE enumeration member 1730
I_BVRT_AUTOMATIC enumeration member 1329	I_CBAELCT VERY_SEVERE enumeration member 1730
I_BVRT_UNLIMITED enumeration member 1329	I_CBAELET.Aggressive_WATER enumeration member 1731
I_BVRT_USER_DEFINED enumeration member 1329	I_CBAELET.NORMAL_WATER enumeration member 1731
I_CAMFT_LIST_OF_MODES enumeration member 433	I_CBAELET.DEFORMED enumeration member 1732
I_CAMFT_MASS_PERCENTAGE enumeration member 433	I_CBAELET.PLAIN enumeration member 1732
I_CAMFT_NON_ACTIVE enumeration member 433	I_CBAELET.EXCEPTIONAL enumeration member 1731
I_CAT_COMB enumeration member 407	I_CBAELET.HIGH enumeration member 1731
I_CAT_COMB_BUCKLING enumeration member 407	I_CBAELET.LOW enumeration member 1731
I_CAT_COMB_CODE enumeration member 407	I_CBC_BEAM_CRACKING.LIMITED enumeration member 1162
I_CAT_COMB_NONLINEAR enumeration member 407	I_CBC_BEAM_CRACKING.NOT_PERMISSIBLE enumeration member 1162
I_CAT_COMB_NONLINEAR_BUCKLING enumeration member 407	I_CBC_BEAM_CRACKING.PERMISSIBLE enumeration member 1162
I_CAT_DYNAMIC_FOOTFALL enumeration member 407	I_CBC_BEAM_CRACKING.UNDEFINED enumeration member 1162
I_CAT_DYNAMIC_FRF enumeration member 407	I_CBCPDT.POINT_COUNT enumeration member 1785
I_CAT_DYNAMIC_HARMONIC enumeration member 407	I_CBCPDT.POINT_SPACING enumeration member 1785
I_CAT_DYNAMIC_MODAL enumeration member 407	I_CBS_BAR_SUBTYPE.ANY enumeration member 1250
I_CAT_DYNAMIC_NONLINEAR_MODAL_WITH_STATIC_FO RCE enumeration member 407	I_CBS_BAR_SUBTYPE.BACK enumeration member 1250
I_CAT_DYNAMIC_SEISMIC enumeration member 407	I_CBS_BAR_SUBTYPE.DIR_HOR enumeration member
I_CAT_DYNAMIC_SPECTRAL enumeration member 407	
I_CAT_MOBILE_MAIN enumeration member 407	

1250	I_CBS_CFOOT_SECTION_RP enumeration member 1177
I_CBS_BAR_SUBTYPE_DIR_VER enumeration member 1250	I_CBSGT_C enumeration member 1790
I_CBS_BAR_SUBTYPE_DIST enumeration member 1250	I_CBSGT_C2 enumeration member 1790
I_CBS_BAR_SUBTYPE_DOWN enumeration member 1250	I_CBSGT_C4 enumeration member 1790
I_CBS_BAR_SUBTYPE_EXT enumeration member 1250	I_CBSGT_I enumeration member 1790
I_CBS_BAR_SUBTYPE_FOOT enumeration member 1250	I_CBSGT_L enumeration member 1790
I_CBS_BAR_SUBTYPE_FRONT enumeration member 1250	I_CBSGT_P enumeration member 1790
I_CBS_BAR_SUBTYPE_GROIN enumeration member 1250	I_CBSGT_R enumeration member 1790
I_CBS_BAR_SUBTYPE_INC_45 enumeration member 1250	I_CBSGT_T enumeration member 1790
I_CBS_BAR_SUBTYPE_INC_LEFT enumeration member 1250	I_CBSGT_TN enumeration member 1790
I_CBS_BAR_SUBTYPE_INC_RIGHT enumeration member 1250	I_CBSGT_UC enumeration member 1790
I_CBS_BAR_SUBTYPE_INT enumeration member 1250	I_CBSGT_UC2 enumeration member 1790
I_CBS_BAR_SUBTYPE_JOIN enumeration member 1250	I_CBSGT_UC4 enumeration member 1790
I_CBS_BAR_SUBTYPE_LEFT enumeration member 1250	I_CBSGT_UL enumeration member 1790
I_CBS_BAR_SUBTYPE_PIN enumeration member 1250	I_CBSGT_UP enumeration member 1790
I_CBS_BAR_SUBTYPE_RIGHT enumeration member 1250	I_CBSGT_UR enumeration member 1790
I_CBS_BAR_SUBTYPE_SHA enumeration member 1250	I_CBSGT_UT enumeration member 1790
I_CBS_BAR_SUBTYPE_SPEC enumeration member 1250	I_CBSGT_UZ enumeration member 1790
I_CBS_BAR_SUBTYPE_STRAIGHT enumeration member 1250	I_CBSGT_Z enumeration member 1790
I_CBS_BAR_SUBTYPE_TABLE enumeration member 1250	I_CBT_ALS enumeration member 429
I_CBS_BAR_SUBTYPE_UBAR enumeration member 1250	I_CBT_BAR_TYPE_LON_BOAT enumeration member 1250
I_CBS_BAR_SUBTYPE_UP enumeration member 1250	I_CBT_BAR_TYPE_LON_CONS enumeration member 1250
I_CBS_BAR_SUBTYPE_UP_RIGHT enumeration member 1250	I_CBT_BAR_TYPE_LON_CORN enumeration member 1250
I_CBS_BAR_SUBTYPE_WALL enumeration member 1250	I_CBT_BAR_TYPE_LON_DIST enumeration member 1250
I_CBS_BEAM_SECTION_LG_BP enumeration member 1160	I_CBT_BAR_TYPE_LON_FLAN enumeration member 1250
I_CBS_BEAM_SECTION_LG_LP enumeration member 1160	I_CBT_BAR_TYPE_LON_HOLE enumeration member 1250
I_CBS_BEAM_SECTION_LG_NP enumeration member 1160	I_CBT_BAR_TYPE_LON_JOIN enumeration member 1250
I_CBS_BEAM_SECTION_LG_RP enumeration member 1160	I_CBT_BAR_TYPE_LON_JOIN_STRUT enumeration member 1250
I_CBS_BEAM_SECTION_NG_BP enumeration member 1160	I_CBT_BAR_TYPE_LON_MAIN enumeration member 1250
I_CBS_BEAM_SECTION_NG_LP enumeration member 1160	I_CBT_BAR_TYPE_LON_MAIN_FIRST enumeration member 1250
I_CBS_BEAM_SECTION_NG_NP enumeration member 1160	I_CBT_BAR_TYPE_LON_SPEC enumeration member 1250
I_CBS_BEAM_SECTION_NG_RP enumeration member 1160	I_CBT_BAR_TYPE_LON_SURF enumeration member 1250
I_CBS_BEAM_SECTION_RG_BP enumeration member 1160	I_CBT_BAR_TYPE_LON_TORS enumeration member 1250
I_CBS_BEAM_SECTION_RG_LP enumeration member 1160	I_CBT_BAR_TYPE_TRA_CONS enumeration member 1250
I_CBS_BEAM_SECTION_RG_NP enumeration member 1160	I_CBT_BAR_TYPE_TRA_CORN enumeration member 1250
I_CBS_BEAM_SECTION_RG_RP enumeration member 1160	I_CBT_BAR_TYPE_TRA_HOLE enumeration member 1250
I_CBS_CFOOT_SECTION_BP enumeration member 1177	I_CBT_BAR_TYPE_TRA_JOIN enumeration member 1250
I_CBS_CFOOT_SECTION_LP enumeration member 1177	I_CBT_BAR_TYPE_TRA_MAIN enumeration member 1250
I_CBS_CFOOT_SECTION_NP enumeration member 1177	I_CBT_BAR_TYPE_TRA_SPEC enumeration member 1250
	I_CBT_BAR_TYPE_TRA_SURF enumeration member 1250
	I_CBT_BAR_TYPE_UNDEFINED enumeration member 1250

I\_CBT\_SLS enumeration member 429  
I\_CBT\_SLS\_EC\_FRE enumeration member 429  
I\_CBT\_SLS\_EC\_QPR enumeration member 429  
I\_CBT\_SLS\_EC\_RAR enumeration member 429  
I\_CBT\_SPC enumeration member 429  
I\_CBT\_ULS enumeration member 429  
I\_CCACT\_GA enumeration member 162  
I\_CCACT\_GS enumeration member 162  
I\_CCACT\_GU\_MAX enumeration member 162  
I\_CCACT\_GU\_MIN enumeration member 162  
I\_CCACT\_KSI\_MAX enumeration member 162  
I\_CCACT\_KSI\_MIN enumeration member 162  
I\_CCACT\_PSI0\_1 enumeration member 162  
I\_CCACT\_PSI0\_2 enumeration member 162  
I\_CCACT\_PSI0\_3 enumeration member 162  
I\_CCACT\_PSI0\_N enumeration member 162  
I\_CCACT\_PSI1 enumeration member 162  
I\_CCACT\_PSI2\_1 enumeration member 162  
I\_CCACT\_PSI2\_N enumeration member 162  
I\_CCACT\_PSIK enumeration member 162  
I\_CCBC\_BM\_VERIFICATION enumeration member 1798  
I\_CCCC\_CL\_VERIFICATION enumeration member 1795  
I\_CCCPT\_ACCIDENTAL enumeration member 165  
I\_CCCPT\_DEAD enumeration member 165  
I\_CCCPT\_LIVE enumeration member 165  
I\_CCCPT\_SEISMIC enumeration member 165  
I\_CCCT\_ACCIDENTAL enumeration member 166  
I\_CCCT\_EXTREMAL enumeration member 166  
I\_CCCT\_FREQUENT enumeration member 166  
I\_CCCT\_FUNDAMENTAL enumeration member 166  
I\_CCCT\_QUASI\_PERM enumeration member 166  
I\_CCCT\_RARE enumeration member 166  
I\_CCCT\_SEISMIC enumeration member 166  
I\_CCCT\_SIMPLIFIED enumeration member 166  
I\_CCCT\_STANDARD enumeration member 166  
I\_CCCT\_USER enumeration member 166  
I\_CCD\_COLUMN\_DIM\_A enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_B enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_C enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_DE enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_EPД enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_H1 enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_H2 enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_HSD enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_HSP enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_HSP\_X enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_HSP\_Y enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_L1 enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_L2 enumeration member 1195  
I\_CCD\_COLUMN\_DIM\_N enumeration member 1195  
I\_CCDVT\_DEFLECTION enumeration member 173  
I\_CCDVT\_FX enumeration member 173  
I\_CCDVT\_FXX\_MAX enumeration member 173  
I\_CCDVT\_FY enumeration member 173  
I\_CCDVT\_FZ enumeration member 173  
I\_CCDVT\_MX enumeration member 173  
I\_CCDVT\_MXX\_MAX enumeration member 173  
I\_CCDVT\_MY enumeration member 173  
I\_CCDVT\_MZ enumeration member 173  
I\_CCDVTREACTIONS enumeration member 173  
I\_CCDVT\_SIGMA\_X enumeration member 173  
I\_CCDVT\_SXX\_MAX enumeration member 173  
I\_CCDVT\_UX enumeration member 173  
I\_CCDVT\_UY enumeration member 173  
I\_CCDVT\_UZ enumeration member 173  
I\_CCF\_MAIN enumeration member 155  
I\_CCF\_MAX enumeration member 155  
I\_CCF\_MIN enumeration member 155  
I\_CCGT\_FULL enumeration member 186  
I\_CCGT\_MANUAL enumeration member 186  
I\_CCGT\_NONE enumeration member 186  
I\_CCGT\_SIMPLIFIED enumeration member 186  
I\_CCO\_AND enumeration member 176  
I\_CCO\_AND\_OR enumeration member 176  
I\_CCO\_EXCLUSIVE\_OR enumeration member 176  
I\_CCT\_COLUMN\_SEC\_C18 enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_C36 enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_C90 enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_L enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_P enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_R enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_TC enumeration member 1195  
I\_CCT\_COLUMN\_SEC\_Z enumeration member 1195  
I\_CEC2CG\_AUTOMATIC enumeration member 1766

I_CEC2CG_C12_15 enumeration member 1766	I_CFST_FOOT_SHAPE_TRAPEZOIDAL enumeration member 1114
I_CEC2CG_C16_20 enumeration member 1766	I_CFST_FOOT_SHAPE_TWO_CL enumeration member 1114
I_CEC2CG_C20_25 enumeration member 1766	I_CFT_FOOT_PIER_TYPE_BOLTED enumeration member 1123
I_CEC2CG_C25_30 enumeration member 1766	I_CFT_FOOT_PIER_TYPE_DOWELED enumeration member 1123
I_CEC2CG_C30_37 enumeration member 1766	I_CFT_FOOT_PIER_TYPEHINGE_1 enumeration member 1123
I_CEC2CG_C35_45 enumeration member 1766	I_CFT_FOOT_PIER_TYPEHINGE_2 enumeration member 1123
I_CEC2CG_C40_50 enumeration member 1766	I_CFT_FOOT_PIER_TYPE_PLAIN enumeration member 1123
I_CEC2CG_C45_55 enumeration member 1766	I_CFT_FOOT_PIER_TYPE_SOCKETED enumeration member 1123
I_CEC2CG_C50_60 enumeration member 1766	I_CFT_FOOT_TYPE_FOOTING enumeration member 1114
I_CEC2ER_1 enumeration member 1766	I_CFT_FOOT_TYPE_WALL enumeration member 1114
I_CEC2ER_2A enumeration member 1766	I_CM_CLOUD enumeration member 1586
I_CEC2ER_2B enumeration member 1766	I_CM_CLOUD_ASYNC enumeration member 1586
I_CEC2ER_3 enumeration member 1766	I_CM_LOCAL enumeration member 1586
I_CEC2ER_4A enumeration member 1766	I_CM_LOCAL_ASYNC enumeration member 1586
I_CEC2ER_4B enumeration member 1766	I_CM_TRAPEZOIDAL enumeration member 611
I_CEC2ER_5A enumeration member 1766	I_CMC_COHERENT_FE_MESH enumeration member 1318
I_CEC2ER_5B enumeration member 1766	I_CMC_USE_OF_KINEMATIC_CONSTRAINTS enumeration member 1318
I_CEC2ER_5C enumeration member 1766	I_CMRT_FOR_NONZERO_AS enumeration member 1771
I_CEC2ITSG_FE_B_22 enumeration member 1766	I_CMRT_FOR_WHOLE_PANEL enumeration member 1771
I_CEC2ITSG_FE_B_32 enumeration member 1766	I_CMRT_NONE enumeration member 1771
I_CEC2ITSG_FE_B_38 enumeration member 1766	I_CMS_CALCULATIONS enumeration member 1585
I_CEC2ITSG_FE_B_44 enumeration member 1766	I_CMS_VERIFICATION enumeration member 1585
I_CEC2NAD_BELGIAN enumeration member 1771	I_CMSL_ERROR enumeration member 1585
I_CEC2NAD_DUTCH enumeration member 1771	I_CMSL_NOTE enumeration member 1585
I_CEC2NAD_FINNISH enumeration member 1771	I_CMSL_WARNING enumeration member 1585
I_CEC2NAD_FRENCH enumeration member 1771	I_CN_ACCIDENTAL enumeration member 406
I_CEC2NAD_GERMAN enumeration member 1771	I_CN_EXPLOATATION enumeration member 406
I_CEC2NAD_ITALIAN enumeration member 1771	I_CN_PERMANENT enumeration member 406
I_CFDT_FOOT_DIM_A enumeration member 1113	I_CN_SEISMIC enumeration member 406
I_CFDT_FOOT_DIM_AP enumeration member 1113	I_CN_SNOW enumeration member 406
I_CFDT_FOOT_DIM_B enumeration member 1113	I_CN_TEMPERATURE enumeration member 406
I_CFDT_FOOT_DIM_BP enumeration member 1113	I_CN_WIND enumeration member 406
I_CFDT_FOOT_DIM_COL_A enumeration member 1113	I_CPN_MASS_CONVERSION_DYNAMIC enumeration member 474
I_CFDT_FOOT_DIM_COL_B enumeration member 1113	I_CPN_MASS_CONVERSION_GLOBAL enumeration member 474
I_CFDT_FOOT_DIM_EX enumeration member 1113	I_CPN84CG_ANOTHER enumeration member 1712
I_CFDT_FOOT_DIM_EY enumeration member 1113	
I_CFDT_FOOT_DIM_H1 enumeration member 1113	
I_CFDT_FOOT_DIM_H2 enumeration member 1113	
I_CFDT_FOOT_DIM_H3 enumeration member 1113	
I_CFDT_FOOT_DIM_H4 enumeration member 1113	
I_CFDT_FOOT_DIM_L enumeration member 1113	
I_CFST_FOOT_SHAPE_RECT enumeration member 1114	

I_CPN84CG_B10 enumeration member 1712	I_CPN84CG_B10 enumeration member 1712
I_CPN84CG_B12_5 enumeration member 1712	I_CPN84CG_B12_5 enumeration member 1712
I_CPN84CG_B15 enumeration member 1712	I_CPN84CG_B15 enumeration member 1712
I_CPN84CG_B17_5 enumeration member 1712	I_CPN84CG_B17_5 enumeration member 1712
I_CPN84CG_B20 enumeration member 1712	I_CPN84CG_B20 enumeration member 1712
I_CPN84CG_B25 enumeration member 1712	I_CPN84CG_B25 enumeration member 1712
I_CPN84CG_B30 enumeration member 1712	I_CPN84CG_B30 enumeration member 1712
I_CPN84CG_B35 enumeration member 1712	I_CPN84CG_B35 enumeration member 1712
I_CPN84CG_B40 enumeration member 1712	I_CPN84CG_B40 enumeration member 1712
I_CPN84CG_B50 enumeration member 1712	I_CPN84CG_B50 enumeration member 1712
I_CPN84CG_B55 enumeration member 1712	I_CPN84CG_B55 enumeration member 1712
I_CPN84CG_B60 enumeration member 1712	I_CPN84CG_B60 enumeration member 1712
I_CPN84CG_B65 enumeration member 1712	I_CPN84CG_B65 enumeration member 1712
I_CPN84ER_MILD enumeration member 1712	I_CPN84ER_MILD enumeration member 1712
I_CPN84ER_MODERATE enumeration member 1712	I_CPN84ER_MODERATE enumeration member 1712
I_CPN84ER_SEVERE_OR_LEAKPROOF enumeration member 1712	I_CPN84ER_SEVERE_OR_LEAKPROOF enumeration member 1712
I_CPN84HT_40 enumeration member 1732	I_CPN84HT_40 enumeration member 1732
I_CPN84HT_40_75 enumeration member 1732	I_CPN84HT_40_75 enumeration member 1732
I_CPN84HT_75 enumeration member 1732	I_CPN84HT_75 enumeration member 1732
I_CPN84HT_WATER enumeration member 1732	I_CPN84HT_WATER enumeration member 1732
I_CPN99CG_ANOTHER enumeration member 1721	I_CPN99CG_ANOTHER enumeration member 1721
I_CPN99CG_B15 enumeration member 1721	I_CPN99CG_B15 enumeration member 1721
I_CPN99CG_B20 enumeration member 1721	I_CPN99CG_B20 enumeration member 1721
I_CPN99CG_B25 enumeration member 1721	I_CPN99CG_B25 enumeration member 1721
I_CPN99CG_B30 enumeration member 1721	I_CPN99CG_B30 enumeration member 1721
I_CPN99CG_B37 enumeration member 1721	I_CPN99CG_B37 enumeration member 1721
I_CPN99CG_B45 enumeration member 1721	I_CPN99CG_B45 enumeration member 1721
I_CPN99CG_B50 enumeration member 1721	I_CPN99CG_B50 enumeration member 1721
I_CPN99CG_B55 enumeration member 1721	I_CPN99CG_B55 enumeration member 1721
I_CPN99CG_B60 enumeration member 1721	I_CPN99CG_B60 enumeration member 1721
I_CPN99CG_B65 enumeration member 1721	I_CPN99CG_B65 enumeration member 1721
I_CPN99CG_B70 enumeration member 1721	I_CPN99CG_B70 enumeration member 1721
I_CPN99ER_X0 enumeration member 1713	I_CPN99ER_X0 enumeration member 1713
I_CPN99ER_XA1_XA2_XA3 enumeration member 1713	I_CPN99ER_XA1_XA2_XA3 enumeration member 1713
I_CPN99ER_XC1_XC2_XC3_XC4 enumeration member 1713	I_CPN99ER_XC1_XC2_XC3_XC4 enumeration member 1713
I_CPN99ER_XD1_XD2_XD3 enumeration member 1713	I_CPN99ER_XD1_XD2_XD3 enumeration member 1713
I_CPN99ER_XF1_XF3 enumeration member 1713	I_CPN99ER_XF1_XF3 enumeration member 1713
I_CPN99ER_XF2_XF4 enumeration member 1713	I_CPN99ER_XF2_XF4 enumeration member 1713
I_CPN99ER_XS1_XS2_XS3 enumeration member 1713	I_CPN99ER_XS1_XS2_XS3 enumeration member 1713
I_CPN84CG_A_0 enumeration member 1711	I_CPN84CG_A_0 enumeration member 1711
I_CPN84CG_A_I enumeration member 1711	I_CPN84CG_A_I enumeration member 1711
	I_CPN84CG_A_II enumeration member 1711
	I_CPN84CG_A_III enumeration member 1711
	I_CPN84CG_A_IIN enumeration member 1711
	I_CRCT_BENDING_COMPRESSION_TENSION enumeration member 1757
	I_CRCT_COMPRESSION_TENSION enumeration member 1757
	I_CRCT_SIMPLE_BENDING enumeration member 1757
	I_CRD_ALONG_X enumeration member 1699
	I_CRD_ALONG_Y enumeration member 1699
	I_CRD_ALONG_Z enumeration member 1699
	I_CRD_AUTOMATIC enumeration member 1699
	I_CRD_CARTESIAN_ALONG_VECTOR enumeration member 1699
	I_CRD_POLAR_POINT enumeration member 1699
	I_CRVTFRF_FREQUENCY enumeration member 442
	I_CS_CANCELLED_BY_USER enumeration member 1586
	I_CS_COMPLETED enumeration member 1586
	I_CS_FAILED_CALCULATION enumeration member 1586
	I_CS_FAILED_GENERATION enumeration member 1586
	I_CS_FAILED_NO_ENTITLEMENT enumeration member 1586
	I_CS_FAILED_VERIFICATION enumeration member 1586
	I_CS_IN_PROGRESS enumeration member 1586
	I_CSNIPCG_B10 enumeration member 1763
	I_CSNIPCG_B12_5 enumeration member 1763
	I_CSNIPCG_B15 enumeration member 1763
	I_CSNIPCG_B20 enumeration member 1763
	I_CSNIPCG_B25 enumeration member 1763
	I_CSNIPCG_B30 enumeration member 1763
	I_CSNIPCG_B35 enumeration member 1763
	I_CSNIPCG_B40 enumeration member 1763
	I_CSNIPCG_B45 enumeration member 1763
	I_CSNIPCG_B50 enumeration member 1763
	I_CSNIPCG_B55 enumeration member 1763
	I_CSNIPCG_B60 enumeration member 1763
	I_CSNIPCG_B7_5 enumeration member 1763
	I_CSNIPCM_AUTOCLAVES enumeration member 1763
	I_CSNIPCM_NORMAL enumeration member 1763
	I_CSNIPCM_THERMAL_TREATMENT enumeration member 1763
	I_CSNIPECT_CELLULAR enumeration member 1763
	I_CSNIPECT_FINE_GRAINED_A enumeration member 1763
	I_CSNIPECT_FINE_GRAINED_B enumeration member 1763

I_CSNIPICT_FINE_GRAINED_V enumeration member 1763	I_CT_EMITTER enumeration member 1647
I_CSNIPICT_HEAVYWEIGHT enumeration member 1763	I_CT_EXTREME_PARAMS enumeration member 1647
I_CSNIPICT_LIGHTWEIGHT_NATURAL_FILLER enumeration member 1763	I_CT_FE_EXTREME_PARAMS enumeration member 1647
I_CSNIPICT_LIGHTWEIGHT_SYNTHETIC_COARSE_GRAINED enumeration member 1763	I_CT_FE_MULTI_RESULT_TYPE enumeration member 1647
I_CSNIPICT_LIGHTWEIGHT_SYNTHETIC_COMPACT enumeration member 1763	I_CT_FE_RESULT_PARAMS enumeration member 1647
I_CSNIPICT_POROUS enumeration member 1763	I_CT_FOUNDATIONS DESIGN enumeration member 1303
I_CSNIPE_EXTERNAL enumeration member 1765	I_CT_GEO_ARC enumeration member 1647
I_CSNIPE_GROUND_VARIABLE_WATER_LEVEL enumeration member 1765	I_CT_GEO_CIRCLE enumeration member 1647
I_CSNIPE_INTERNAL enumeration member 1765	I_CT_GEO_CONTOUR enumeration member 1647
I_CSNIPSG_A_I enumeration member 1763	I_CT_GEO_CURVE_DIV enumeration member 1647
I_CSNIPSG_A_II enumeration member 1763	I_CT_GEO_LAYER enumeration member 1647
I_CSNIPSG_A_III enumeration member 1763	I_CT_GEO_OBJECT enumeration member 1647
I_CSNIPSG_A_IIIB_Y enumeration member 1763	I_CT_GEO_POINT_2D enumeration member 1647
I_CSNIPSG_A_IIIB_YH enumeration member 1763	I_CT_GEO_POINT_3D enumeration member 1647
I_CSNIPSG_A_IV enumeration member 1763	I_CT_GEO_POINT_3D_COLLECTION enumeration member 1647
I_CSNIPSG_A_V enumeration member 1763	I_CT_GEO_POLYLINE enumeration member 1647
I_CSNIPSG_A_VI enumeration member 1763	I_CT_GEO_SEGMENT enumeration member 1647
I_CSNIPSG_B_II enumeration member 1763	I_CT_GEO_SEGMENT_ARC enumeration member 1647
I_CSNIPSG_BP_I enumeration member 1763	I_CT_GEO_SEGMENT_COLLECTION enumeration member 1647
I_CSNIPSG_BP_II enumeration member 1763	I_CT_GEO_SEGMENT_LINE enumeration member 1647
I_CSNIPSG_K_19 enumeration member 1763	I_CT_HTML_VIEW enumeration member 1647
I_CSNIPSG_K_7 enumeration member 1763	I_CT_JOINT_ANGLE_LOAD enumeration member 1647
I_CSR_SLAB_RNF_BAR_AND_NET enumeration member 1219	I_CT_JOINT_FIXED_LOAD enumeration member 1647
I_CSR_SLAB_RNF_BAR_ONLY enumeration member 1219	I_CT_JOINT_GUSSET_CROSS_LOAD enumeration member 1647
I_CSR_SLAB_RNF_NET_ONLY enumeration member 1219	I_CT_JOINT_GUSSET_FLANGE_LOAD enumeration member 1647
I_CSS_SLAB_SUPPORT_FIXED enumeration member 1220	I_CT_JOINT_GUSSET_SIMPLE_LOAD enumeration member 1647
I_CSS_SLAB_SUPPORT_PINNED enumeration member 1220	I_CT_JOINT_KNEE_LOAD enumeration member 1647
I_CST_STEEL_TYPE_LONG enumeration member 1252	I_CT_JOINT_LOAD enumeration member 1647
I_CST_STEEL_TYPE_SPEC enumeration member 1252	I_CT_JOINT_PINNED_LOAD enumeration member 1647
I_CST_STEEL_TYPE_TRAN enumeration member 1252	I_CT_JOINT_TUBE_LOAD enumeration member 1647
I_CST_STEEL_TYPE_WIRE enumeration member 1252	I_CT_MOBILE enumeration member 429
I_CT_CALCULATE_IN_CLOUD enumeration member 1540	I_CT_MODIF_EXTRUSION enumeration member 1647
I_CT_CALCULATE_LOCALLY enumeration member 1540	I_CT_MODIF_LATHE enumeration member 1647
I_CT_CASE_ANALYSIS_MODES_FILTER enumeration member 1647	I_CT_MODIF_PYRAMID enumeration member 1647
I_CT_CODE_COMBINATION enumeration member 429	I_CT_MOVING LOADS enumeration member 1303
I_CT_CODE_COMBINATIONS enumeration member 1303	I_CT_NAMES_ARRAY enumeration member 1647
I_CT_COMBINATION enumeration member 429	I_CT_NUMBERS_ARRAY enumeration member 1647
	I_CT_NUMBERS_DICTIONARY enumeration member 1647
	I_CT_OBJECTS_ARRAY enumeration member 1647

I_CT_OPER_MESHING enumeration member 1647	I_D_TEMPLATE enumeration member 1536
I_CT_OPER_ROTATION enumeration member 1647	I_D_USER_CONF enumeration member 1536
I_CT_OPER_SCALING enumeration member 1647	I_D_USER_MAIN enumeration member 1536
I_CT_OPER_TRANSLATION enumeration member 1647	I_D_USER_OUTPUT enumeration member 1536
I_CT_PARAM_DEF enumeration member 1647	I_D_USER_PROJECTS enumeration member 1536
I_CT_POINTS_ARRAY enumeration member 1647	I_D_USER_TEMPLATE enumeration member 1536
I_CT_RC_REAL_REINF enumeration member 1303	I_DADT_CONSTANT enumeration member 442
I_CT_RC_THEORETICAL_REINF enumeration member 1303	I_DADT_MODAL enumeration member 442
I_CT_RESULT_QUERY_PARAMS enumeration member 1647	I_DADT_NONE enumeration member 442
I_CT_RTF_VIEW enumeration member 1647	I_DADT_VARIABLE enumeration member 442
I_CT_SEISMIC_LOADS enumeration member 1303	I_DAT_ANSI_AISC_360_10_ASD enumeration member 1589
I_CT_SIMPLE enumeration member 429	I_DAT_ANSI_AISC_360_10_LRFD enumeration member 1589
I_CT_SNOW_WIND_LOADS enumeration member 1303	I_DAT_USER_DEFINED enumeration member 1589
I_CT_SPECTRAL_ANALYSIS_POINTS_COLLECTION enumeration member 1647	I_DB3D_EC3_AUTO enumeration member 1810
I_CT_SPECTRAL_ANALYSIS_SPECTRUM enumeration member 1647	I_DB3D_EC3_CANTILEVER_2_0 enumeration member 1810
I_CT_STEEL_CONNECTIONS enumeration member 1303	I_DB3D_EC3_INTERN_ADJBAR_6 enumeration member 1810
I_CT_STEEL_STRUCTURES enumeration member 1303	I_DB3D_EC3_INTERNAL_BRACINGS enumeration member 1810
I_CT_STRUCTURE_GEO_ANALYSER enumeration member 1647	I_DB3D_EC3_NO enumeration member 1810
I_CT_STRUCTURE_MERGE_DATA enumeration member 1647	I_DB3D_EC3_PINNED_ADJBAR_1 enumeration member 1810
I_CT_SW_STRUCT3D enumeration member 1647	I_DB3D_EC3_PINNED_ADJBAR_3 enumeration member 1810
I_CT_SW_STRUCT3D_ELEMENT enumeration member 1647	I_DB3D_EC3_PINNED_PINNED_1_0 enumeration member 1810
I_CT_SW_STRUCT3D_FRAME enumeration member 1647	I_DB3D_EC3_PINNED_STIFF_0_7 enumeration member 1810
I_CT_SW_STRUCT3D_GEN_PARAMS enumeration member 1647	I_DB3D_EC3_STIFF_ADJBAR_1 enumeration member 1810
I_CT_SW_STRUCT3D_PURLIN_GEN_PARAMS enumeration member 1647	I_DB3D_EC3_STIFF_ADJBAR_3 enumeration member 1810
I_CT_TABLE_SCREEN_CAPTURE_PARAMS enumeration member 1647	I_DB3D_EC3_STIFF_STIFF_0_5 enumeration member 1810
I_CT_TIMBER_STRUCTURES enumeration member 1303	I_DB3D_EC3_TRUSS_CHORD_0_9 enumeration member 1810
I_CT_VALUES_ARRAY enumeration member 1647	I_DB3D_EC3_TRUSS_DIAGONAL_0_8 enumeration member 1810
I_CT_VIEW_SCREEN_CAPTURE_PARAMS enumeration member 1647	I_DB3D_EC3_USER_DEFINED enumeration member 1810
I_CT_X enumeration member 610	I_DCSFT_CHECKSLEND enumeration member 1906
I_CT_XY enumeration member 610	I_DCSFT_FIRE enumeration member 1906
I_CT_Y enumeration member 610	I_DCSFT_POINTMIDDLE enumeration member 1906
I_D_CONF enumeration member 1536	I_DCSPCALCMETHOD enumeration member 1907
I_D_EXE enumeration member 1536	I_DCSPCALCTYPE enumeration member 1907
I_D_HELP enumeration member 1536	I_DCSPV_CALCMETHOD_1 enumeration member 1907
I_D_MAIN enumeration member 1536	I_DCSPV_CALCMETHOD_2 enumeration member 1907
I_D_RES enumeration member 1536	I_DCSPV_CALCMETHOD_3 enumeration member 1907
	I_DCSPV_DIMENSIONING enumeration member 1907
	I_DCSPV_EFFRATIO enumeration member 1907
	I_DCSPV_GROUP_VERIF enumeration member 1907

I_DCSPV_MANUAL_VERIF enumeration member 1907	I_DLL_EC3_UPP_SECT_PAR_LOADED enumeration member 1809
I_DCSPV_MAXSLEND enumeration member 1907	I_DLLCT_LATERAL_LOAD_DIR enumeration member 1594
I_DCSPV_MEMBERS_VERIF enumeration member 1907	I_DLLCT_ONE_DIR_ONLY enumeration member 1594
I_DCSPV_OPTIMIZATION enumeration member 1907	I_DLLCT_TWO_DIR_INDEPENDENTLY enumeration member 1594
I_DE_CONF enumeration member 1538	I_DLT_CB71_FORCE_AT_DISTANCE enumeration member 1879
I_DE_TEMPLATE enumeration member 1538	I_DLT_CB71_MOMENT_AT_END enumeration member 1879
I_DE_USER_CONF enumeration member 1538	I_DLT_CB71_UNIFORM_LOAD enumeration member 1879
I_DE_USER_TEMPLATE enumeration member 1538	I_DLT_EC3_CONCENTRATED_FORCE enumeration member 1808
I_DEDDT_Y enumeration member 1890	I_DLT_EC3_MOMENTS_AT_ENDS enumeration member 1808
I_DEDDT_Z enumeration member 1890	I_DLT_EC3_UNIFORM_LOAD enumeration member 1808
I_DEDIPT_BUCKLING_Y enumeration member 1890	I_DLT_EC3_UNIFORM_MOMENT enumeration member 1808
I_DEDIPT_BUCKLING_Z enumeration member 1890	I_DMCBC_Y enumeration member 1911
I_DEDIPT_LBUCKLING_L enumeration member 1890	I_DMCBC_Z enumeration member 1911
I_DEDIPT_LBUCKLING_U enumeration member 1890	I_DMCRV_CORRECT enumeration member 1910
I_DEDPT_ACC enumeration member 1889	I_DMCRV_INCORRECT enumeration member 1910
I_DEDPT_CASE_NO enumeration member 1889	I_DMCRV_INCORRECTDATA enumeration member 1910
I_DEDPT_COMP_NO enumeration member 1889	I_DMCRV_INSTABILITY enumeration member 1910
I_DEDPT_ELEM_NO enumeration member 1889	I_DMDBDT_BUCKLING_U enumeration member 1885
I_DEDPT_LOADCLASS enumeration member 1889	I_DMDBDT_BUCKLING_V enumeration member 1885
I_DEDPT_POINT_NO enumeration member 1889	I_DMDBDT_BUCKLING_Y enumeration member 1885
I_DEDPT_POINTS_NUM enumeration member 1889	I_DMDBDT_BUCKLING_Z enumeration member 1885
I_DI_BAR_SECTIONS enumeration member 1469	I_DMDDDT_DEFY enumeration member 1885
I_DI_PANEL_THICKNESS enumeration member 1469	I_DMDDDT_DEFZ enumeration member 1885
I_DI_SUPPORTS enumeration member 1469	I_DMDDDT_DISP_X enumeration member 1885
I_DLBCD_EC3_CANTILEVER_2_0 enumeration member 1809	I_DMDDDT_DISP_Y enumeration member 1885
I_DLBCD_EC3_INTERNAL_BRACINGS enumeration member 1809	I_DMDDEVT_HUMIDITY enumeration member 1900
I_DLBCD_EC3_NO enumeration member 1809	I_DMDDEVT_TRAIT enumeration member 1900
I_DLBCD_EC3_PINNED_PINNED_1_0 enumeration member 1809	I_DMDLDT_LENGTH_U enumeration member 1884
I_DLBCD_EC3_STIFF_STIFF_0_5 enumeration member 1809	I_DMDLDT_LENGTH_V enumeration member 1884
I_DLBCD_EC3_USER_DEFINED enumeration member 1809	I_DMDLDT_LENGTH_Y enumeration member 1884
I_DLBT_EC3_CANTILEVER enumeration member 1809	I_DMDLDT_LENGTH_Z enumeration member 1884
I_DLBT_EC3_NO enumeration member 1809	I_DMDLEV_CATEGORY enumeration member 1900
I_DLBT_EC3_SYMMETR_LOADED enumeration member 1809	I_DMDLEV_NATURE enumeration member 1900
I_DLL_EC3_CENTER_LOADED enumeration member 1809	I_DMDLEV_TIMB_TYPE enumeration member 1900
I_DLL_EC3_LOW_EDGE_LOADED enumeration member 1809	I_DMDMT_CONCRETE enumeration member 1884
I_DLL_EC3_LOW_SEC_PART_LOADED enumeration member 1809	I_DMDMT_STEEL enumeration member 1884
I_DLL_EC3_UPP_EDGE_LOADED enumeration member 1809	I_DMDMT_TIMBER enumeration member 1884
	I_DMDT_ALUMINIUM enumeration member 1899
	I_DMDT_BEAM enumeration member 1884

I_DMDT_COLUMN enumeration member 1884	I_DOF_UX enumeration member 1648
I_DMDT_MEMBER enumeration member 1884	I_DOF_UY enumeration member 1648
I_DMDT_NONE enumeration member 1899	I_DOF_UZ enumeration member 1648
I_DMDT_STEEL enumeration member 1899	I_DPDIT_STANDARD enumeration member 1904
I_DMDT_USER enumeration member 1884	I_DPDIT_VAR_BEGEND enumeration member 1904
I_DMDT_WOOD enumeration member 1899	I_DPDIT_VAR_IN_POINT enumeration member 1904
I_DMDVT_CS enumeration member 1899	I_DPDIT_VAR_MIDDLE enumeration member 1904
I_DMDVT_DAMPCOEF enumeration member 1899	I_DPDIT_CAE enumeration member 1903
I_DMDVT_E enumeration member 1899	I_DPDIT_CAEP enumeration member 1903
I_DMDVT_E_5 enumeration member 1899	I_DPDIT_CAI enumeration member 1903
I_DMDVT_FU enumeration member 1899	I_DPDIT_CAIP enumeration member 1903
I_DMDVT_G enumeration member 1899	I_DPDIT_CAISSON enumeration member 1903
I_DMDVT_LX enumeration member 1899	I_DPDIT_CROSS enumeration member 1903
I_DMDVT_NU enumeration member 1899	I_DPDIT_CUSER enumeration member 1903
I_DMDVT_PN_E_ADDITIONAL enumeration member 1899	I_DPDIT_DCEC enumeration member 1903
I_DMDVT_PN_E_TRANS enumeration member 1899	I_DPDIT_DCED enumeration member 1903
I_DMDVT_RE enumeration member 1899	I_DPDIT_DCEP enumeration member 1903
I_DMDVT_RE_AX_COMMR enumeration member 1899	I_DPDIT_DCIG enumeration member 1903
I_DMDVT_RE_AX_TENS enumeration member 1899	I_DPDIT_DCIP enumeration member 1903
I_DMDVT_RE_BENDING enumeration member 1899	I_DPDIT_DRECT enumeration member 1903
I_DMDVT_RE_SHEAR enumeration member 1899	I_DPDIT_FRTG enumeration member 1903
I_DMDVT_RE_TR_COMPRESSION enumeration member 1899	I_DPDIT_HEA enumeration member 1903
I_DMDVT_RE_TR_TENS enumeration member 1899	I_DPDIT_HEAA enumeration member 1903
I_DMDVT_RO enumeration member 1899	I_DPDIT_HEB enumeration member 1903
I_DMDVT_RT enumeration member 1899	I_DPDIT_HEC enumeration member 1903
I_DMDVT_TRANS enumeration member 1899	I_DPDIT_HEM enumeration member 1903
I_DMRTC_DESCRIPTION enumeration member 1911	I_DPDIT_HER enumeration member 1903
I_DMRTC_NAME enumeration member 1911	I_DPDIT_HHEA enumeration member 1903
I_DMRTC_PARAGRAPH enumeration member 1911	I_DPDIT_HHEB enumeration member 1903
I_DMRTC_UNIT enumeration member 1911	I_DPDIT_HHEM enumeration member 1903
I_DMRTC_VALUE enumeration member 1911	I_DPDIT_IYPE enumeration member 1903
I_DMRTLTL_1PARAM enumeration member 1911	I_DPDIT_INSYM enumeration member 1903
I_DMRTLTL_4HEADER enumeration member 1911	I_DPDIT_IPE enumeration member 1903
I_DMRTLTL_4PARAM enumeration member 1911	I_DPDIT_IPEA enumeration member 1903
I_DMRTLTL_5HEADER enumeration member 1911	I_DPDIT_IPEO enumeration member 1903
I_DMRTLTL_5PARAM enumeration member 1911	I_DPDIT_IPER enumeration member 1903
I_DMRTLTL_CAPTION enumeration member 1911	I_DPDIT_IPEV enumeration member 1903
I_DMRTLTL_PRINTCAPTION enumeration member 1911	I_DPDIT_IPN enumeration member 1903
I_DMRTLTL_SUBCAPTION enumeration member 1911	I_DPDIT_ISYM enumeration member 1903
I_DOF_RX enumeration member 1648	I_DPDIT_MHEA enumeration member 1903
I_DOF_RY enumeration member 1648	I_DPDIT_MHEB enumeration member 1903
I_DOF_RZ enumeration member 1648	I_DPDIT_MHEM enumeration member 1903

I_DPDVT_MIPE enumeration member 1903	I_DPDVT_VPY enumeration member 1904
I_DPDVT_NONE enumeration member 1903	I_DPDVT_VPZ enumeration member 1904
I_DPDVT_OSIE enumeration member 1903	I_DPDVT_VY enumeration member 1904
I_DPDVT_PRS enumeration member 1903	I_DPDVT_VZ enumeration member 1904
I_DPDVT_RECT enumeration member 1903	I_DRV_COEFF enumeration member 51
I_DPDVT_TCAR enumeration member 1903	I_DRV_ENTIRE_STRUCTURE enumeration member 51
I_DPDVT_TEAE enumeration member 1903	I_DRV_X enumeration member 51
I_DPDVT_TEAI enumeration member 1903	I_DRV_Y enumeration member 51
I_DPDVT_THEX enumeration member 1903	I_DRV_Z enumeration member 51
I_DPDVT_TREC enumeration member 1903	I_DSMRT_FACTOR_0_8 enumeration member 1597
I_DPDVT_TRON enumeration member 1903	I_DSMRT_FACTOR_0_8_AND_NL_INCREASE enumeration member 1597
I_DPDVT_TUBE enumeration member 1903	I_DSMRT_FACTOR_0_8_AND_TAU_B enumeration member 1597
I_DPDVT_TUSER enumeration member 1903	I_DST_DOUBLE enumeration member 1881
I_DPDVT_UAP enumeration member 1903	I_DST_LONG enumeration member 1881
I_DPDVT_UPN enumeration member 1903	I_DST_TEXT enumeration member 1881
I_DPDVT_UUAP enumeration member 1903	I_DT_ANCHOR_BOLTS enumeration member 1317
I_DPDVT_UUPN enumeration member 1903	I_DT_BOLTS enumeration member 1317
I_DPDVT_X enumeration member 1903	I_DT_BUILDING_SOILS enumeration member 1317
I_DPDVT_B enumeration member 1904	I_DT_REINFORCING_BARS enumeration member 1317
I_DPDVT_B2 enumeration member 1904	I_DT_SECTIONS enumeration member 1317
I_DPDVT_BD enumeration member 1904	I_DT_STANDARD_LOADS enumeration member 1317
I_DPDVT_BF enumeration member 1904	I_DT_VEHICLE_LOADS enumeration member 1317
I_DPDVT_BF2 enumeration member 1904	I_DT_WIRE_FABRICS enumeration member 1317
I_DPDVT_DIS enumeration member 1904	I_DUT_DISP_L enumeration member 1926
I_DPDVT_EA enumeration member 1904	I_DUT_FORCE enumeration member 1926
I_DPDVT_EA2 enumeration member 1904	I_DUT_LENGTH enumeration member 1926
I_DPDVT_ES enumeration member 1904	I_DUT_MOMENT enumeration member 1926
I_DPDVT_ES2 enumeration member 1904	I_DUT_NONE enumeration member 1926
I_DPDVT_H enumeration member 1904	I_DUT_SECDIMEN enumeration member 1926
I_DPDVT_HD enumeration member 1904	I_DUT_SECMI enumeration member 1926
I_DPDVT_HW enumeration member 1904	I_DUT_SECSUR enumeration member 1926
I_DPDVT_I enumeration member 1904	I_DUT_SECVOL enumeration member 1926
I_DPDVT_IY enumeration member 1904	I_DUT_STRESS enumeration member 1926
I_DPDVT_IZ enumeration member 1904	I_DYST_EC3_AVERAGE enumeration member 1844
I_DPDVT_MASSE enumeration member 1904	I_DYST_EC3_BASIC enumeration member 1844
I_DPDVT_MSY enumeration member 1904	I_EC8_SAGT_A enumeration member 298
I_DPDVT_MSZ enumeration member 1904	I_EC8_SAGT_B enumeration member 298
I_DPDVT_R enumeration member 1904	I_EC8_SAGT_C enumeration member 298
I_DPDVT_R2 enumeration member 1904	I_EC8_SAGT_D enumeration member 298
I_DPDVT_S enumeration member 1904	I_EC8_SAGT_E enumeration member 298
I_DPDVT_SY enumeration member 1904	I_EC8_SAST_TYPE_1 enumeration member 298
I_DPDVT_SZ enumeration member 1904	

I_EC8_SAST_TYPE_2 enumeration member 298	I_EVT_FORCE_BAR_MY enumeration member 910
I_EFF_DXF enumeration member 1266	I_EVT_FORCE_BAR_MZ enumeration member 910
I_EFF_STR enumeration member 1266	I_EVT_FRF_VX enumeration member 910
I_ESM_AUTO enumeration member 1559	I_EVT_FRF_VY enumeration member 910
I_ESM_FRONTAL enumeration member 1559	I_EVT_FRF_VZ enumeration member 910
I_ESM_ITERATIVE enumeration member 1559	I_EVT_MODAL_AVERAGE_PARTICIPATION_COEFF enumeration member 910
I_ESM_MULTI_THREADED enumeration member 1559	I_EVT_MODAL_CURMASS_RX enumeration member 910
I_ESM_SKYLINE enumeration member 1559	I_EVT_MODAL_CURMASS_RY enumeration member 910
I_ESM_SPARSE enumeration member 1559	I_EVT_MODAL_CURMASS_RZ enumeration member 910
I_ESM_SPARSE_M enumeration member 1559	I_EVT_MODAL_CURMASS_UX enumeration member 910
I_EVT_DEFLECTION_UX enumeration member 910	I_EVT_MODAL_CURMASS_UY enumeration member 910
I_EVT_DEFLECTION_UY enumeration member 910	I_EVT_MODAL_CURMASS_UZ enumeration member 910
I_EVT_DEFLECTION_UZ enumeration member 910	I_EVT_MODAL_DAMPING enumeration member 910
I_EVT_DISPLACEMENT_BAR_RX enumeration member 910	I_EVT_MODAL_EIGENVALUE enumeration member 910
I_EVT_DISPLACEMENT_BAR_RY enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RX enumeration member 910
I_EVT_DISPLACEMENT_BAR_RZ enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RX_1 enumeration member 910
I_EVT_DISPLACEMENT_BAR_UX enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RY enumeration member 910
I_EVT_DISPLACEMENT_BAR_UY enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RY_1 enumeration member 910
I_EVT_DISPLACEMENT_BAR_UZ enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RZ enumeration member 910
I_EVT_DISPLACEMENT_NODE_RX enumeration member 910	I_EVT_MODAL_EIGENVECTOR_RZ_1 enumeration member 910
I_EVT_DISPLACEMENT_NODE_RY enumeration member 910	I_EVT_MODAL_EIGENVECTOR_UX enumeration member 910
I_EVT_DISPLACEMENT_NODE_RZ enumeration member 910	I_EVT_MODAL_EIGENVECTOR_UY enumeration member 910
I_EVT_DISPLACEMENT_NODE_UX enumeration member 910	I_EVT_MODAL_EIGENVECTOR_UY_1 enumeration member 910
I_EVT_DISPLACEMENT_NODE_UY enumeration member 910	I_EVT_MODAL_EIGENVECTOR_UZ enumeration member 910
I_EVT_DISPLACEMENT_NODE_UZ enumeration member 910	I_EVT_MODAL_EIGENVECTOR_UZ_1 enumeration member 910
I_EVT_FOOTFALL_A enumeration member 910	I_EVT_MODAL_ENERGY enumeration member 910
I_EVT_FOOTFALL_EXCITATION_NODE enumeration member 910	I_EVT_MODAL_FREQUENCY enumeration member 910
I_EVT_FOOTFALL_FREQUENCY enumeration member 910	I_EVT_MODAL_MASSES_X enumeration member 910
I_EVT_FOOTFALL_RF_OVERALL enumeration member 910	I_EVT_MODAL_MASSES_Y enumeration member 910
I_EVT_FOOTFALL_RF_RESONANT enumeration member 910	I_EVT_MODAL_MASSES_Z enumeration member 910
I_EVT_FOOTFALL_RF_TRANSIENT enumeration member 910	I_EVT_MODAL_NODE_MASSES_X enumeration member 910
I_EVT_FOOTFALL_VRMQ enumeration member 910	I_EVT_MODAL_NODE_MASSES_Y enumeration member 910
I_EVT_FOOTFALL_VRMS enumeration member 910	
I_EVT_FORCE_BAR_FX enumeration member 910	
I_EVT_FORCE_BAR_FY enumeration member 910	
I_EVT_FORCE_BAR_FZ enumeration member 910	
I_EVT_FORCE_BAR_MX enumeration member 910	

I_EVT_MODAL_NODE_MASSES_Z enumeration member 910	910
I_EVT_MODAL_PARTICIPATION_COEFF_RX enumeration member 910	I_EVT_PSEUDOSTATIC_FORCE_TY enumeration member 910
I_EVT_MODAL_PARTICIPATION_COEFF_RY enumeration member 910	I_EVT_PSEUDOSTATIC_FORCE_TZ enumeration member 910
I_EVT_MODAL_PARTICIPATION_COEFF_RZ enumeration member 910	I_EVTREACTION_FX enumeration member 910
I_EVT_MODAL_PARTICIPATION_COEFF_UX enumeration member 910	I_EVTREACTION_FY enumeration member 910
I_EVT_MODAL_PARTICIPATION_COEFF_UY enumeration member 910	I_EVTREACTION_FZ enumeration member 910
I_EVT_MODAL_PARTICIPATION_COEFF_UX enumeration member 910	I_EVTREACTION_MX enumeration member 910
I_EVT_MODAL_PERIOD enumeration member 910	I_EVTREACTION_MY enumeration member 910
I_EVT_MODAL_PRECISION enumeration member 910	I_EVTREACTION_MZ enumeration member 910
I_EVT_MODAL_PULSATION enumeration member 910	I_EVT_STRESS_BAR_FX_SX enumeration member 910
I_EVT_MODAL_SUMMASS_RX enumeration member 910	I_EVT_STRESS_BAR_SMAX enumeration member 910
I_EVT_MODAL_SUMMASS_RY enumeration member 910	I_EVT_STRESS_BAR_SMAX_MY enumeration member 910
I_EVT_MODAL_SUMMASS_RZ enumeration member 910	I_EVT_STRESS_BAR_SMAX_MZ enumeration member 910
I_EVT_MODAL_SUMMASS_UX enumeration member 910	I_EVT_STRESS_BAR_SMIN enumeration member 910
I_EVT_MODAL_SUMMASS_UY enumeration member 910	I_EVT_STRESS_BAR_SMIN_MY enumeration member 910
I_EVT_MODAL_SUMMASS_UZ enumeration member 910	I_EVT_STRESS_BAR_SMIN_MZ enumeration member 910
I_EVT_MODAL_TOTMASS_RX enumeration member 910	I_EVT_STRESS_BAR_T enumeration member 910
I_EVT_MODAL_TOTMASS_RY enumeration member 910	I_EVT_STRESS_BAR_TY enumeration member 910
I_EVT_MODAL_TOTMASS_RZ enumeration member 910	I_EVT_STRESS_BAR_TZ enumeration member 910
I_EVT_MODAL_TOTMASS_UX enumeration member 910	I_EVT_TIME_ARX enumeration member 910
I_EVT_MODAL_TOTMASS_UY enumeration member 910	I_EVT_TIME_ARY enumeration member 910
I_EVT_MODAL_TOTMASS_UZ enumeration member 910	I_EVT_TIME_ARZ enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_FX enumeration member 910	I_EVT_TIME_AX enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_FY enumeration member 910	I_EVT_TIME_AY enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_FZ enumeration member 910	I_EVT_TIME_AZ enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_GX enumeration member 910	I_EVT_TIME_VRX enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_GY enumeration member 910	I_EVT_TIME_VRY enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_GZ enumeration member 910	I_EVT_TIME_VRZ enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_MX enumeration member 910	I_EVT_TIME_VX enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_MY enumeration member 910	I_EVT_TIME_VY enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_MZ enumeration member 910	I_EVT_TIME_VZ enumeration member 910
I_EVT_PSEUDOSTATIC_FORCE_TX enumeration member	I_EXT_ANGLE2BEAM enumeration member 2167
	I_EXT_NONE enumeration member 2167
	I_FAEF_CONCRETE_CENTRE enumeration member 395
	I_FAEF_SCI_P354 enumeration member 395
	I_FAEM_FULL_EXCITATION enumeration member 395
	I_FAEM_SELF_EXCITATION enumeration member 395
	I_FANST_ALL_NODES enumeration member 397
	I_FANST_NODES_BELONGING_TO_SELECTED_PANELS enumeration member 397
	I_FANST_SELECTED_NODES enumeration member 397

I_FET_Q4 enumeration member 831	I_FRT_COMPLEX_MXX_TOP_NEN enumeration member 979
I_FET_Q8 enumeration member 831	I_FRT_COMPLEX_MXX_TOP_WA enumeration member 979
I_FET_T3 enumeration member 831	I_FRT_COMPLEX_MYY_BOTTOM_NEN enumeration member 979
I_FET_T6 enumeration member 831	I_FRT_COMPLEX_MYY_BOTTOM_WA enumeration member 979
I_FET_VOL_B20 enumeration member 831	I_FRT_COMPLEX_MYY_TOP_NEN enumeration member 979
I_FET_VOL_B8 enumeration member 831	I_FRT_COMPLEX_MYY_TOP_WA enumeration member 979
I_FET_VOL_T10 enumeration member 831	I_FRT_COMPLEX_N_MISES enumeration member 979
I_FET_VOL_T4 enumeration member 831	I_FRT_COMPLEX_S_MISES enumeration member 979
I_FET_VOL_W15 enumeration member 831	I_FRT_DETAILED_MXX enumeration member 979
I_FET_VOL_W6 enumeration member 831	I_FRT_DETAILED_MXY enumeration member 979
IFLT_ABSOLUTE_MAXIMUM enumeration member 932	I_FRT_DETAILED_MYY enumeration member 979
IFLT_ARBITRARY enumeration member 932	I_FRT_DETAILED_NXX enumeration member 979
IFLT_LOWER enumeration member 932	I_FRT_DETAILED_NXY enumeration member 979
IFLT_MAXIMUM enumeration member 932	I_FRT_DETAILED_NYY enumeration member 979
IFLT_MIDDLE enumeration member 932	I_FRT_DETAILED_PNORM enumeration member 979
IFLT_MINIMUM enumeration member 932	I_FRT_DETAILED_QXX enumeration member 979
IFLT_UPPER enumeration member 932	I_FRT_DETAILED_QYY enumeration member 979
I_FMP_CB71_AUTOMATIC enumeration member 1880	I_FRT_DETAILED_RNORM enumeration member 979
I_FMP_CB71_HORIZONTAL enumeration member 1880	I_FRT_DETAILED_RXX enumeration member 979
I_FMP_CB71_VERTICAL enumeration member 1880	I_FRT_DETAILED_RYY enumeration member 979
I_FPT_CB71_LESS enumeration member 1880	I_FRT_DETAILED_SXX enumeration member 979
I_FPT_CB71_MORE enumeration member 1880	I_FRT_DETAILED_SXY enumeration member 979
I_FRK_ELEMENT_CENTER enumeration member 981	I_FRT_DETAILED_SYY enumeration member 979
I_FRK_ELEMENT_NODE enumeration member 981	I_FRT_DETAILED_TXX enumeration member 979
I_FRK_PANEL_NODE enumeration member 981	I_FRT_DETAILED_TYY enumeration member 979
I_FRRCP_HORIZONTAL_BOTTOM enumeration member 972	I_FRT_DETAILED_UXX enumeration member 979
I_FRRCP_HORIZONTAL_MIDDLE enumeration member 972	I_FRT_DETAILED_UYY enumeration member 979
I_FRRCP_HORIZONTAL_TOP enumeration member 972	I_FRT_DETAILED_WNORM enumeration member 979
I_FRRCP_VERTICAL_LEFT enumeration member 972	I_FRT_FOOT_RES_AX enumeration member 1122
I_FRRCP_VERTICAL_MIDDLE enumeration member 972	I_FRT_FOOT_RES_AY enumeration member 1122
I_FRRCP_VERTICAL_RIGHT enumeration member 972	I_FRT_PRINCIPAL_M1 enumeration member 979
I_FRS_GLOBAL_SMOOTHING enumeration member 980	I_FRT_PRINCIPAL_M1_2 enumeration member 979
I_FRS_IN_ELEMENT_CENTER enumeration member 980	I_FRT_PRINCIPAL_M2 enumeration member 979
I_FRS_NO_SMOOTHING enumeration member 980	I_FRT_PRINCIPAL_MAL enumeration member 979
I_FRS_SMOOTHING_ACCORDING_TO_SELECTION enumeration member 980	I_FRT_PRINCIPAL_N1 enumeration member 979
I_FRS_SMOOTHING_WITHIN_A_PANEL enumeration member 980	I_FRT_PRINCIPAL_N1_2 enumeration member 979
I_FRT_COMPLEX_M_MISES enumeration member 979	I_FRT_PRINCIPAL_N2 enumeration member 979
I_FRT_COMPLEX_MXX_BOTTOM_NEN enumeration member 979	I_FRT_PRINCIPAL_NAL enumeration member 979
I_FRT_COMPLEX_MXX_BOTTOM_WA enumeration member 979	I_FRT_PRINCIPAL_Q1_2 enumeration member 979
	I_FRT_PRINCIPAL_S1 enumeration member 979

I_FRT_PRINCIPAL_S1_2 enumeration member 979	I_GCS_GLOBAL enumeration member 1607
I_FRT_PRINCIPAL_S2 enumeration member 979	I_GCS_LOCAL_AFTER_OFFSET enumeration member 1607
I_FRT_PRINCIPAL_SAL enumeration member 979	I_GCS_LOCAL_ORIGINAL enumeration member 1607
I_FRT_PRINCIPAL_T1_2 enumeration member 979	I_GOT_ARC enumeration member 1608
I_FRT_PRINCIPAL_U enumeration member 979	I_GOT_CIRCLE enumeration member 1608
I_FRT_PRINCIPAL_UGX enumeration member 979	I_GOT_CONTOUR enumeration member 1608
I_FRT_PRINCIPAL_UGY enumeration member 979	I_GOT_INTERSECTION enumeration member 1608
I_FRT_PRINCIPAL_UGZ enumeration member 979	I_GOT_NONE enumeration member 1608
I_FRT_REDUCED_HEIGHT enumeration member 979	I_GOT_POLYLINE enumeration member 1608
I_FRT_REDUCED_LENGTH enumeration member 979	I_GP_XY enumeration member 1627
I_FRT_REDUCED_MY enumeration member 979	I_GP_XZ enumeration member 1627
I_FRT_REDUCED_MZ enumeration member 979	I_GP_YZ enumeration member 1627
I_FRT_REDUCED_NX enumeration member 979	I_GST_ARC enumeration member 1607
I_FRT_REDUCED_SE enumeration member 979	I_GST_LINE enumeration member 1607
I_FRT_REDUCED_SO enumeration member 979	I_GST_NONE enumeration member 1607
I_FRT_REDUCED_T enumeration member 979	I_ICRV_LOCAL enumeration member 48
I_FRT_REDUCED_TY enumeration member 979	I_ICRV_NPOINTS enumeration member 48
I_FRT_REDUCED_TZ enumeration member 979	I_ICRV_PROJECTION enumeration member 48
I_FRT_REINF_A_MIN enumeration member 979	I_ICRV_PX1 enumeration member 48
I_FRT_REINF_AX enumeration member 979	I_ICRV_PX2 enumeration member 48
I_FRT_REINF_AX_BOTTOM enumeration member 979	I_ICRV_PX3 enumeration member 48
I_FRT_REINF_AX_TOP enumeration member 979	I_ICRV_PY1 enumeration member 48
I_FRT_REINF_AY enumeration member 979	I_ICRV_PY2 enumeration member 48
I_FRT_REINF_AY_BOTTOM enumeration member 979	I_ICRV_PY3 enumeration member 48
I_FRT_REINF_AY_TOP enumeration member 979	I_ICRV_PZ1 enumeration member 48
I_FRT_REINF_E_AX_BOTTOM enumeration member 979	I_ICRV_PZ2 enumeration member 48
I_FRT_REINF_E_AX_TOP enumeration member 979	I_ICRV_PZ3 enumeration member 48
I_FRT_REINF_E_AY_BOTTOM enumeration member 979	I_IPT_CHOLESKY enumeration member 1566
I_FRT_REINF_E_AY_TOP enumeration member 979	I_IPT_DIAGONAL enumeration member 1566
I_FRT_REINF_F enumeration member 979	I_IPT_GAUSS enumeration member 1566
I_GADM_BEGIN_MIDDLE_END enumeration member 1626	I_IPT_ICCF enumeration member 1566
I_GADM_CENTER_BEGIN_END enumeration member 1626	I_IS2000_ER_MILD enumeration member 1771
I_GCA_NONE enumeration member 1606	I_IS2000_ER_MODERATE enumeration member 1771
I_GCA_OX enumeration member 1606	I_IS2000_ER_MOST_SEVERE enumeration member 1771
I_GCA_OY enumeration member 1606	I_IS2000_ER_SEVERE enumeration member 1771
I_GCA_OZ enumeration member 1606	I_IS2000_ER VERY_SEVERE enumeration member 1771
I_GCAS_OX_MINUS enumeration member 1626	I_ISM_0 enumeration member 1570
I_GCAS_OX_PLUS enumeration member 1626	I_ISM_1 enumeration member 1570
I_GCAS_OY_MINUS enumeration member 1626	I_ISM_2 enumeration member 1570
I_GCAS_OY_PLUS enumeration member 1626	I_ISMU_1_2 enumeration member 1571
I_GCAS_OZ_MINUS enumeration member 1626	I_ISMU_1_4 enumeration member 1571
I_GCAS_OZ_PLUS enumeration member 1626	I_ISMU_MAX enumeration member 1571

I\_ISMU\_MIN enumeration member 1571  
I\_JAPP\_LONG\_FLANGE\_TO\_COLUMN enumeration member 2030  
I\_JAPP\_SHORT\_FLANGE\_TO\_COLUMN enumeration member 2030  
I\_JAPT\_RECT enumeration member 2009  
I\_JAPT\_TUBE enumeration member 2009  
I\_JAT\_PIN enumeration member 1993  
I\_JAT\_PLATE enumeration member 1993  
I\_JAT\_ROD enumeration member 1993  
I\_JAT\_SIMPLE enumeration member 1993  
I\_JAU\_DEGREES enumeration member 2167  
I\_JAU\_RADIANS enumeration member 2167  
I\_JBT\_HIGH\_TENSION enumeration member 2157  
I\_JBT\_NORMAL enumeration member 2157  
I\_JCBPCM\_ELASTIC enumeration member 1988  
I\_JCBPCM\_PLASTIC enumeration member 1988  
I\_JCDT\_IN\_STRUCTURE enumeration member 2144  
I\_JCDT\_STANDALONE enumeration member 2144  
I\_JCMT\_BEAMANGLES enumeration member 2168  
I\_JCMT\_BEAM\_BRACKETS enumeration member 2168  
I\_JCMT\_BEAM\_MAIN enumeration member 2168  
I\_JCT\_ANGLE enumeration member 2144  
I\_JCT\_BEAM\_GIRDER enumeration member 2144  
I\_JCT\_BOLTS enumeration member 2140  
I\_JCT\_COL\_CONCR enumeration member 2144  
I\_JCT\_COL\_FIXED enumeration member 2144  
I\_JCT\_COL\_PINNED enumeration member 2144  
I\_JCT\_GUSSET\_CROSS enumeration member 2144  
I\_JCT\_GUSSET\_FLANGE enumeration member 2144  
I\_JCT\_GUSSET\_SIMPLE enumeration member 2144  
I\_JCT\_KNEE\_BOLTED enumeration member 2144  
I\_JCT\_KNEE\_WELDED enumeration member 2144  
I\_JCT\_TRNS enumeration member 2144  
I\_JCT\_TUBE enumeration member 2144  
I\_JCT\_UNKNOWN enumeration member 2144  
I\_JCT\_WELDS enumeration member 2140  
I\_JET\_LANGLES\_RANGLES\_FLANGE enumeration member 2140  
I\_JET\_LANGLES\_RANGLES\_WEB enumeration member 2140  
I\_JET\_LANGLES\_RPLATES enumeration member 2140  
I\_JET\_LPLATES\_RANGLES enumeration member 2140  
I\_JET\_LPLATES\_RPLATES enumeration member 2140  
I\_JET\_L\_SEATSANGLES\_RSEATS enumeration member 2140  
I\_JET\_L\_SEATSANGLES\_RSEATSANGLES enumeration member 2140  
I\_JET\_L\_SEATS\_RSEATS enumeration member 2140  
I\_JET\_L\_SEATS\_RSEATSANGLES enumeration member 2140  
I\_JET\_L\_STIFF\_LONG\_R\_STIFF\_LONG enumeration member 2140  
I\_JET\_L\_STIFF\_LONG\_R\_STIFF\_SHORT enumeration member 2140  
I\_JET\_L\_STIFF\_SHORT\_R\_STIFF\_LONG enumeration member 2140  
I\_JET\_L\_STIFF\_SHORT\_R\_STIFF\_SHORT enumeration member 2140  
I\_JET\_RANGLES\_FLANGE enumeration member 2140  
I\_JET\_RANGLES\_WEB enumeration member 2140  
I\_JET\_RPLATES enumeration member 2140  
I\_JET\_RSEATS enumeration member 2140  
I\_JET\_RSEATSANGLES enumeration member 2140  
I\_JET\_RSTIFF\_LONG enumeration member 2140  
I\_JET\_RSTIFF\_SHORT enumeration member 2140  
I\_JFCBST\_COMPLEX enumeration member 1979  
I\_JFCBST\_NONE enumeration member 1979  
I\_JFCBST\_SIMPLE enumeration member 1979  
I\_JFPT\_ANGLE enumeration member 1999  
I\_JFPT\_NONE enumeration member 1999  
I\_JFPT\_WELD enumeration member 1999  
I\_JFST\_1 enumeration member 2019  
I\_JFST\_2 enumeration member 2019  
I\_JFST\_3 enumeration member 2019  
I\_JFST\_4 enumeration member 2019  
I\_JFST\_5 enumeration member 2019  
I\_JGCPC\_ALL\_CUTTED enumeration member 2075  
I\_JGCPC\_CONTINUE\_LEFTLOWER\_RIGHTUPPER enumeration member 2075  
I\_JGCPC\_CONTINUE\_LEFTUPPER\_RIGHTLOWER enumeration member 2075  
I\_JGCT\_ACUTE enumeration member 2067  
I\_JGCT\_CUT\_IN enumeration member 2067  
I\_JGCT\_CUT\_OUT enumeration member 2067  
I\_JGDPT\_LONG\_FLANGE enumeration member 2111  
I\_JGDPT\_LONG\_FLANGE\_INV enumeration member 2111  
I\_JGDPT\_SHORT\_FLANGE enumeration member 2111  
I\_JGDPT\_SHORT\_FLANGE\_INV enumeration member 2111  
I\_JGPC\_CONTINUOUS enumeration member 2090

I_JGFPC_CUTTED enumeration member 2090	I_JPCBST_RDIAG enumeration member 1961
I_JGFPRT_IRREGULAR enumeration member 2111	I_JPCBST_RI enumeration member 1961
I_JGFPRT_REGULAR enumeration member 2111	I_JSFT_ABRUPT_CHANGES_HEIGHT enumeration member 1983
I_JGFT_BOLTED enumeration member 2057	I_JSFT_RECT_CROSS_SECT enumeration member 1983
I_JGFT_WELDED enumeration member 2057	I_JSFT_TAPERED enumeration member 1983
I_JGSPP_DIAGONAL enumeration member 2057	I_JST_ALONG_BEARER enumeration member 2025
I_JGSPP_VERTICAL enumeration member 2057	I_JST_ALONG_SECOND enumeration member 2025
I_JKDST_DOUBLE enumeration member 1955	I_JTPT_RECT enumeration member 2036
I_JKDST_LEFT enumeration member 1955	I_JTPT_RECT_RECT enumeration member 2036
I_JKDST_RIGHT enumeration member 1955	I_JTPT_RECT_TUBE enumeration member 2036
I_JKF1_BOLTED enumeration member 1941	I_JTPT_TUBE enumeration member 2036
I_JKF1_WELDED enumeration member 1941	I_JTPT_TUBE_TUBE enumeration member 2036
I_JKM_FLAN_PLATE_LOWER enumeration member 1960	I_JTPT_UNKNOWN enumeration member 2036
I_JKM_FLAN_PLATE_UPPER enumeration member 1960	I_JTT_K enumeration member 2036
I_JKM_PLATE enumeration member 1960	I_JTT_KT enumeration member 2036
I_JKM_PLATE_LEFT enumeration member 1960	I_JTT_N enumeration member 2036
I_JKM_TENSION_PLATE_LOWER enumeration member 1960	I_JTT_NONE enumeration member 2036
I_JKM_TENSION_PLATE_UPPER enumeration member 1960	I_JTT_T enumeration member 2036
I_JKM_WEB_BEAM_STIFF_V_LOWER enumeration member 1960	I_JTT_X enumeration member 2036
I_JKM_WEB_BEAM_STIFF_V_UPPER enumeration member 1960	I_JTT_Y enumeration member 2036
I_JKM_WEB_COLUMN_STIFF_H_LOWER enumeration member 1960	I_JWFRP_FLANGE_LONG enumeration member 2147
I_JKM_WEB_COLUMN_STIFF_H_UPPER enumeration member 1960	I_JWFRP_FLANGE_TRAN enumeration member 2147
I_JKRT_BRACKET enumeration member 1941	I_JWFRP_OTHER enumeration member 2147
I_JKRT_NONE enumeration member 1941	I_JWFRP_WEB_LONG enumeration member 2147
I_JKRT_STIFF enumeration member 1941	I_JWFRP_WEB_TRAN enumeration member 2147
I_JKT_BEAM2BEAM enumeration member 1941	I_JWT_DOUBLE enumeration member 2157
I_JKT_BEAM2COLUMN enumeration member 1941	I_JWT_I enumeration member 2003
I_JKT_BEAM2COLUMN_CONTINUE enumeration member 1941	I_JWT_L enumeration member 2003
I_JKWST_DIAG enumeration member 1955	I_JWT_NONE enumeration member 2003
I_JKWST_NONE enumeration member 1955	I_JWT_SINGLE enumeration member 2157
I_JKWST_PLATE enumeration member 1955	I_JWT_T enumeration member 2003
I_JLT_MANUAL_FORCES enumeration member 2167	I_JWT_X enumeration member 2003
I_JLT_ROBOT_FORCES enumeration member 2167	I_L_PRINTOUT enumeration member 1537
I_JPCBST_LDIAG enumeration member 1961	I_L_REGIONAL enumeration member 1537
I_JPCBST_LI enumeration member 1961	I_L_WORK enumeration member 1537
I_JPCBST_LRDIAG enumeration member 1961	I_L3DRV_GAMMA enumeration member 46
I_JPCBST_LRI enumeration member 1961	I_L3DRV_LOCAL enumeration member 46
I_JPCBST_NONE enumeration member 1961	I_L3DRV_MX1 enumeration member 46
	I_L3DRV_MX2 enumeration member 46
	I_L3DRV_MY1 enumeration member 46
	I_L3DRV_MY2 enumeration member 46
	I_L3DRV_MZ1 enumeration member 46

I_L3DRV_MZ2 enumeration member 46	I_LI_RESULTS_STRESS_STRUCTURE enumeration member 1468
I_L3DRV_PX1 enumeration member 46	I_LI_TOOL_FINAL_DRAWING enumeration member 1468
I_L3DRV_PX2 enumeration member 46	I_LI_TOOL_SECTION_DEFINITION enumeration member 1468
I_L3DRV_PY1 enumeration member 46	I_LI_TOOL_TEXT_FILE enumeration member 1468
I_L3DRV_PY2 enumeration member 46	I_LL_CB71_BOTTOM enumeration member 1879
I_L3DRV_PZ1 enumeration member 46	I_LL_CB71_MIDDLE enumeration member 1879
I_L3DRV_PZ2 enumeration member 46	I_LL_CB71_UP enumeration member 1879
I_LBT_CB71_CANTILEVER enumeration member 1879	I_LLT_BEAM_LOAD_CONTINUOUS enumeration member 1159
I_LBT_CB71_FIXED_SUPPORTS enumeration member 1879	I_LLT_BEAM_LOAD_SELF_WEIGHT enumeration member 1159
I_LBT_CB71_NONE enumeration member 1879	I_LLT_BEAM_LOAD_TRAPEZOIDAL1 enumeration member 1159
I_LBT_CB71_PINNED_SUPPORTS enumeration member 1879	I_LLT_BEAM_LOAD_TRAPEZOIDAL2 enumeration member 1159
I_LE_CLOUD_SOLVE enumeration member 1539	I_LLT_BEAM_LOAD_TRAPEZOIDAL3 enumeration member 1159
I_LE_LOCAL_SOLVE enumeration member 1539	I_LLT_BEAM_LOAD_UNIFORM enumeration member 1159
ILES_ENTITLED enumeration member 1539	I_LOERV_GAMMA enumeration member 50
ILES_NOT_ENTITLED enumeration member 1539	I_LOERV_LOCAL_SYSTEM enumeration member 50
ILES_NOT_SIGNED_IN enumeration member 1539	I_LOERV_MX enumeration member 50
ILES_UNKNOWN enumeration member 1539	I_LOERV_MY enumeration member 50
I_LI DESIGN CONNECTIONS enumeration member 1468	I_LOERV_MZ enumeration member 50
I_LI DESIGN_RC_MEMBERS enumeration member 1468	I_LOERV_PX enumeration member 50
I_LI DESIGN_STEEL_ALUMINUM enumeration member 1468	I_LOERV_PY enumeration member 50
I_LI DESIGN_WOOD enumeration member 1468	I_LOERV_PZ enumeration member 50
I_LI FOOTING_DEFINITION enumeration member 1468	I_LRDT_MINUS enumeration member 701
I_LI FOUNDATION_DEFINITION enumeration member 1468	I_LRDT_NONE enumeration member 701
I_LI MODEL_BARS enumeration member 1468	I_LRDT_PLUS enumeration member 701
I_LI MODEL_GEOMETRY enumeration member 1468	I_LRDT_RELEASED enumeration member 701
I_LI MODEL LOADS enumeration member 1468	I_LRT_BAR_DILATATION enumeration member 54
I_LI MODEL_NODES enumeration member 1468	I_LRT_BAR_FORCE_CONCENTRATED enumeration member 54
I_LI MODEL_PROPERTIES enumeration member 1468	I_LRT_BAR_FORCE_CONCENTRATED_MASS enumeration member 54
I_LI MODEL_SUPPORTS enumeration member 1468	I_LRT_BAR_MOMENT_DISTRIBUTED enumeration member 54
I_LI RC_BEAM_DEFINITION enumeration member 1468	I_LRT_BAR_THERMAL enumeration member 54
I_LI RC_BEAM_WALL_DEFINITION enumeration member 1468	I_LRT_BAR_TRAPEZOIDALE enumeration member 54
I_LI RC_COLUMN_DEFINITION enumeration member 1468	I_LRT_BAR_TRAPEZOIDALE_MASS enumeration member 54
I_LI RC_SLAB_PROVIDED_REINF enumeration member 1468	I_LRT_BAR_UNIFORM enumeration member 54
I_LI RC_SLAB_REINFORCEMENT enumeration member 1468	I_LRT_BAR_UNIFORM_MASS enumeration member 54
I_LI RC_SLAB_REQUIRED_REINF enumeration member 1468	I_LRT_DEAD enumeration member 54
I_LI RESULTS_DETAILED enumeration member 1468	
I_LI RESULTS_DIAGRAMS enumeration member 1468	
I_LI RESULTS_MAPS enumeration member 1468	
I_LI RESULTS_STRESS_BARS enumeration member 1468	

I_LRT_IN_3_POINTS enumeration member 54	I_LT_CLADDING enumeration member 17
I_LRT_IN_CONTOUR enumeration member 54	I_LT_LINEAR_RELEASE enumeration member 17
I_LRT_LINEAR enumeration member 54	I_LT_MATERIAL enumeration member 17
I_LLRT_LINEAR_3D enumeration member 54	I_LT_MEMBER_REINFORCEMENT_PARAMS enumeration member 17
I_LLRT_LINEAR_ON_EDGES enumeration member 54	I_LT_MEMBER_TYPE enumeration member 17
I_LLRT_MASS_ACTIVATION enumeration member 54	I_LT_NODE_COMPATIBILITY enumeration member 17
I_LLRT_MOBILE_DISTRIBUTED enumeration member 54	I_LT_NODE_RELEASE enumeration member 17
I_LLRT_MOBILE_POINT_FORCE enumeration member 54	I_LT_NODE_RIGID_LINK enumeration member 17
I_LLRT_NODE_ACCELERATION enumeration member 54	I_LT_PANEL_CALC_MODEL enumeration member 17
I_LLRT_NODE_DISPLACEMENT enumeration member 54	I_LT_PANEL_REINFORCEMENT enumeration member 17
I_LLRT_NODE_FORCE enumeration member 54	I_LT_PANEL_THICKNESS enumeration member 17
I_LLRT_NODE_FORCE_IN_POINT enumeration member 54	I_LT_SOLID_PROPERTIES enumeration member 17
I_LLRT_NODE_FORCE_MASS enumeration member 54	I_LT_SUPPORT enumeration member 17
I_LLRT_NODE_VELOCITY enumeration member 54	I_LT_UNKNOWN enumeration member 17
I_LLRT_POINT_AUXILIARY enumeration member 54	I_LT_VEHICLE enumeration member 17
I_LLRT_PRESSURE enumeration member 54	I_MAA_BASE_REDUCTION enumeration member 191
I_LLRT_SPECTRUM_VALUE enumeration member 54	I_MAA_BLOCK_SUBSPACE_ITERATION enumeration member 191
I_LLRT_SURFACE_ON_OBJECT enumeration member 54	I_MAA_LANCZOS enumeration member 191
I_LLRT_THERMAL enumeration member 54	I_MAA_PCG enumeration member 191
I_LLRT_THERMAL_IN_3_POINTS enumeration member 54	I_MAA_PCG_RITZ enumeration member 191
I_LLRT_UNIFORM enumeration member 54	I_MAA_SUBSPACE_ITERATION enumeration member 191
I_LRV_LOCAL enumeration member 45	I_MALT_FREQUENCY enumeration member 192
I_LRV_MX1 enumeration member 45	I_MALT_PERIOD enumeration member 192
I_LRV_MX2 enumeration member 45	I_MALT_PULSATION enumeration member 192
I_LRV_MY1 enumeration member 45	I_MAM_MODAL enumeration member 190
I_LRV_MY2 enumeration member 45	I_MAM_SEISMIC enumeration member 190
I_LRV_NODE_1 enumeration member 45	I_MAM_SEISMIC_PSEUDO enumeration member 190
I_LRV_NODE_2 enumeration member 45	I_MAMMT_CONSISTENT enumeration member 191
I_LRV_PZ1 enumeration member 45	I_MAMMT_LUMPED enumeration member 191
I_LRV_PZ2 enumeration member 45	I_MAMMT_LUMPED_WITH_ROTATIONS enumeration member 191
I_LS_ALS enumeration member 431	I_MARV_ACTIVATION_DIR enumeration member 46
I_LS_SLS enumeration member 431	I_MARV_CASE_NUM enumeration member 46
I_LS_SPC enumeration member 431	I_MARV_FACTOR enumeration member 46
I_LS_ULS enumeration member 431	I_MARV_INPUT_DIR_X enumeration member 46
I_LT_BAR_CABLE enumeration member 17	I_MARV_INPUT_DIR_Y enumeration member 46
I_LT_BAR_ELASTIC_GROUND enumeration member 17	I_MARV_INPUT_DIR_Z enumeration member 46
I_LT_BAR_GEO_IMPERFECTIONS enumeration member 17	I_MARV_SIGN enumeration member 46
I_LT_BAR_NONLINEAR_HINGE enumeration member 17	I_MAT_SURFACE_AND_VOLUMETRIC enumeration member 827
I_LT_BAR_OFFSET enumeration member 17	I_MAT_SURFACE_ONLY enumeration member 827
I_LT_BAR_RELEASE enumeration member 17	I_MAT_VOLUMETRIC_ONLY enumeration member 827
I_LT_BAR_SECTION enumeration member 17	
I_LT_BAREND_BRACKET enumeration member 17	

I_MCAPT_AUTOMATIC enumeration member 340	I_MM_MOHR_COULOMB enumeration member 667
I_MCAPT_SELECTION enumeration member 340	I_MM_RANKIN enumeration member 667
I_MCF_MAIN enumeration member 332	I_MMT_COONS enumeration member 808
I_MCF_MAX enumeration member 332	I_MMT_DELAUNAY enumeration member 808
I_MCF_MIN enumeration member 332	I_MPDT_SQUARE_IN_RECT enumeration member 806
I_MCT_10P enumeration member 1035	I_MPDT_TRIANG_AND_SQUARE_IN_TRIANG enumeration member 806
I_MCT_2SM enumeration member 1035	I_MPDT_TRIANG_AND_TRAPEZ_IN_TRIANG enumeration member 806
I_MCT_ALL enumeration member 1035	I_MPDT_TRIANG_IN_RECT enumeration member 806
I_MCT_CQC enumeration member 1035	I_MPDT_TRIANG_IN_TRIANG enumeration member 806
I_MCT_NONE enumeration member 1035	I_MPFRV_FX enumeration member 52
I_MCT_SRSS enumeration member 1035	I_MPFRV_FY enumeration member 52
I_MDRV_DX enumeration member 52	I_MPFRV_FZ enumeration member 52
I_MDRV_DY enumeration member 52	I_MPFRV_X enumeration member 52
I_MDRV_PX enumeration member 52	I_MPFRV_Y enumeration member 52
I_MDRV_PY enumeration member 52	I_MPFRV_Z enumeration member 52
I_MDRV_PZ enumeration member 52	I_MRT_DOUBLE enumeration member 827
I_MDRV_X enumeration member 52	I_MRT_SIMPLE enumeration member 827
I_MDRV_Y enumeration member 52	I_MRT_TRIPLE enumeration member 827
I_MDRV_Z enumeration member 52	I_MSFET_3NODE_TRIANG enumeration member 807
I_MDT_DELAUNAY enumeration member 808	I_MSFET_4NODE_QUADRIL enumeration member 807
I_MDT_DELAUNAY_AND_KANG enumeration member 808	I_MSFET_6NODE_TRIANG enumeration member 807
I_MDT_KANG enumeration member 808	I_MSFET_8NODE_QUADRIL enumeration member 807
I_MET_PERFECTLY_PLASTIC enumeration member 668	I_MST_ALL enumeration member 1035
I_MET_PLASTIC_WITH_HARDENING enumeration member 668	I_MST_FIRST_N enumeration member 1035
I_MEUM_DAMAGE enumeration member 670	I_MST_NONE enumeration member 1035
I_MEUM_ELASTIC enumeration member 670	I_MST_SINGLE enumeration member 1035
I_MEUM_MIXED enumeration member 670	I_MT_ALL enumeration member 667
I_MEUM_PLASTIC enumeration member 670	I_MT_ALUMINIUM enumeration member 667
I_MFR_ANY enumeration member 805	I_MT_COARSE enumeration member 805
I_MFR_FORCED enumeration member 805	I_MT_CONCRETE enumeration member 667
I_MFR_NONE enumeration member 805	I_MT_FINE enumeration member 805
I_MFR_PROPOSED enumeration member 805	I_MT_FLOORS_COARSE enumeration member 805
I_MFR_RECOMMENDED enumeration member 805	I_MT_FLOORS_NORMAL enumeration member 805
I_MGT_AUTOMATIC enumeration member 809	I_MT_NORMAL enumeration member 805
I_MGT_ELEMENT_SIZE enumeration member 809	I_MT_OTHER enumeration member 667
I_MGT_USER enumeration member 809	I_MT_STEEL enumeration member 667
I_MIR_NEVER enumeration member 806	I_MT_TIMBER enumeration member 667
I_MIR_OFTEN enumeration member 806	I_MT_USER enumeration member 805
I_MIR_RARELY enumeration member 806	I_MT_WALLS_COARSE enumeration member 805
I_MM_DRUCKER_PRAGER enumeration member 667	I_MT_WALLS_NORMAL enumeration member 805
I_MM_ELASTIC enumeration member 667	I_MTT_GLUE_LAMINATED enumeration member 667
I_MM_HUBER_MISES enumeration member 667	

I_MTT_KERTO_Q enumeration member 667	I_NFIPRV_FZ enumeration member 49
I_MTT_KERTO_Q_OLD enumeration member 667	I_NFIPRV_GAMMA enumeration member 49
I_MTT_KERTO_S enumeration member 667	I_NFIPRV_MX enumeration member 49
I_MTT_KERTO_S_OLD enumeration member 667	I_NFIPRV_MY enumeration member 49
I_MTT_NORMAL enumeration member 667	I_NFIPRV_MZ enumeration member 49
I_MVFET_4NODE_TETRAHED enumeration member 807	I_NFIPRV_POINT_X enumeration member 49
I_MVFET_8NODE_HEXAHEDE enumeration member 807	I_NFIPRV_POINT_Y enumeration member 49
I_NAAT_ARC_LENGTH_METHOD enumeration member 314	I_NFIPRV_POINT_Z enumeration member 49
I_NAAT_DIRECT_ITERATION_METHOD enumeration member 314	I_NFRV_ALPHA enumeration member 39
I_NAAT_INCREMENTAL_METHOD enumeration member 314	I_NFRV_BETA enumeration member 39
I_NAAT_PREDICTOR_CORRECTOR_METHOD enumeration member 314	I_NFRV_CX enumeration member 39
I_NACRV_ALPHA enumeration member 54	I_NFRV_CY enumeration member 39
I_NACRV_BETA enumeration member 54	I_NFRV_CZ enumeration member 39
I_NACRV_GAMMA enumeration member 54	I_NFRV_FX enumeration member 39
I_NACRV_UX enumeration member 54	I_NFRV_FY enumeration member 39
I_NACRV_UY enumeration member 54	I_NFRV_FZ enumeration member 39
I_NACRV_UZ enumeration member 54	I_NFRV_GAMMA enumeration member 39
I_NARV_ALPHA enumeration member 40	I_NHCT_FX enumeration member 604
I_NARV_BETA enumeration member 40	I_NHCT_FY enumeration member 604
I_NARV_CX enumeration member 40	I_NHCT_FZ enumeration member 604
I_NARV_CY enumeration member 40	I_NHCT_MX enumeration member 604
I_NARV_CZ enumeration member 40	I_NHCT_MY enumeration member 604
I_NARV_FX enumeration member 40	I_NHCT_MZ enumeration member 604
I_NARV_FY enumeration member 40	I_NHCT_SX enumeration member 604
I_NARV_FZ enumeration member 40	I_NHMT_FORCE_DISPLACEMENT enumeration member 598
I_NARV_GAMMA enumeration member 40	I_NHMT_MOMENT_ROTATION enumeration member 598
I_NARV_NR enumeration member 40	I_NHMT_STRESS_PAIN enumeration member 598
I_NDRV_ALPHA enumeration member 40	I_NHMUT_DAMAGE enumeration member 600
I_NDRV_BETA enumeration member 40	I_NHMUT_ELASTIC enumeration member 600
I_NDRV_GAMMA enumeration member 40	I_NHMUT_MIXED enumeration member 600
I_NDRV_RX enumeration member 40	I_NHMUT_PLASTIC enumeration member 600
I_NDRV_RY enumeration member 40	I_NLCT_B_LINEAR enumeration member 681
I_NDRV_RZ enumeration member 40	I_NLCT_CUSTOM enumeration member 681
I_NDRV_UX enumeration member 40	I_NLCT_GAP_HOOK enumeration member 681
I_NDRV_UY enumeration member 40	I_NLCT_LINEAR enumeration member 681
I_NDRV_UZ enumeration member 40	I_NLCT_PARABOLIC enumeration member 681
I_NFIPRV_ALPHA enumeration member 49	I_NLCT_PARABOLIC_EC2 enumeration member 681
I_NFIPRV_BETA enumeration member 49	I_NLCT_PERFECTLY_PLASTIC enumeration member 681
I_NFIPRV_FX enumeration member 49	I_NLCT_PLASTIC_WITH_HARDENING enumeration member 681
I_NFIPRV_FY enumeration member 49	I_NLMT_FORCE_DISPLACEMENT enumeration member 693
	I_NLMT_MOMENT_ROTATION enumeration member 693

I_NLSAT_ANY enumeration member 693	I_OST_WALL enumeration member 846
I_NLSAT_NEGATIVE enumeration member 693	I_OT_BAR enumeration member 36
I_NLSAT_POSITIVE enumeration member 693	I_OT_CASE enumeration member 36
I_NSFD_RX enumeration member 510	I_OT_FAMILY enumeration member 36
I_NSFD_RY enumeration member 510	I_OTFINITE_ELEMENT enumeration member 36
I_NSFD_RZ enumeration member 510	I_OT_GEOMETRY enumeration member 36
I_NSFD_UX enumeration member 510	I_OT_NODE enumeration member 36
I_NSFD_UY enumeration member 510	I_OT_OBJECT enumeration member 36
I_NSFD_UZ enumeration member 510	I_OT_PANEL enumeration member 36
I_NSODFT_MINUS enumeration member 510	I_OT_UNDEFINED enumeration member 36
I_NSODFT_NONE enumeration member 510	I_OT_VOLUME enumeration member 36
I_NSODFT_PLUS enumeration member 510	I_PARV_ALPHA enumeration member 44
I_NVRV_ALPHA enumeration member 53	I_PARV_BETA enumeration member 44
I_NVRV_BETA enumeration member 53	I_PARV_CX enumeration member 44
I_NVRV_GAMMA enumeration member 53	I_PARV_CY enumeration member 44
I_NVRV_UX enumeration member 53	I_PARV_CZ enumeration member 44
I_NVRV_UY enumeration member 53	I_PARV_FX enumeration member 44
I_NVRV_UZ enumeration member 53	I_PARV_FY enumeration member 44
I_OFF_HTML enumeration member 1495	I_PARV_FZ enumeration member 44
I_OFF_PNG enumeration member 1495	I_PARV_GAMMA enumeration member 44
I_OFF_RTF enumeration member 1495	I_PARV_X enumeration member 44
I_OFF_RTF_JPEG enumeration member 1495	I_PARV_Y enumeration member 44
I_OFF_TEXT enumeration member 1495	I_PARV_Z enumeration member 44
I_OLXDDT_CARTESIAN enumeration member 788	I_PCDT_FULL_PLANE enumeration member 978
I_OLXDDT_POLAR enumeration member 788	I_PCDT_LIMITED_PLANE enumeration member 978
I_OLXDDT_UNDEFINED enumeration member 788	I_PCMD_FLEXIBLE enumeration member 713
I_OMP_EXTRUSION enumeration member 739	I_PCMD_NONE enumeration member 713
I_OMP_LATHE enumeration member 739	I_PCMD_RIGID enumeration member 713
I_OMP_NONE enumeration member 739	I_PCMFET_NONE enumeration member 712
I_OMP_PYRAMID enumeration member 739	I_PCMFET_SHELL enumeration member 712
I_OOT_MESH enumeration member 739	I_PCMLT_ANALYTICALFINITE_ELEMENTS enumeration member 713
I_OOT_NONE enumeration member 739	I_PCMLT_SIMPLIFIED_ONE_WAY enumeration member 713
I_OOT_ROTATE enumeration member 739	I_PCMLT_SIMPLIFIED_TWO_WAY enumeration member 713
I_OOT_SCALE enumeration member 739	I_PCT_BEAM enumeration member 1261
I_OOT_TRANSLATE enumeration member 739	I_PCT_BWALL enumeration member 1261
I_OPT_MAIN enumeration member 773	I_PCT_COLUMN enumeration member 1261
I_OPT_REFERENCE enumeration member 773	I_PCT_CONTINUOUS_FOOT enumeration member 1261
I_OPT_SIDE enumeration member 773	I_PCT_DRAWING enumeration member 1261
I_OST_BEAM enumeration member 846	I_PCT_FOOT enumeration member 1261
I_OST_COLUMN enumeration member 846	I_PCT_JOINT enumeration member 1261
I_OST_SLAB enumeration member 846	I_PCT_RETAINING_WALL enumeration member 1261
I_OST_UNDEFINED enumeration member 846	

I_PCT_SLAB enumeration member 1261	I_POLDM_ACCELERATION enumeration member 391
I_PL_MEMBER_REINFORCEMENT_PARAMS_STANDARD enumeration member 28	I_POLDM_USER_DEFINED enumeration member 391
I_PL_MEMBER_TYPE_BEAM enumeration member 28	I_PRV_DIRECTION enumeration member 45
I_PL_MEMBER_TYPE_COLUMN enumeration member 28	I_PRV_H enumeration member 45
I_PL_MEMBER_TYPE_RC_BEAM enumeration member 28	I_PRV_P enumeration member 45
I_PL_MEMBER_TYPE_RC_COLUMN enumeration member 28	I_PRV_RO enumeration member 45
I_PL_MEMBER_TYPE_SIMPLE_BAR enumeration member 28	I_PS_BAR_INACTIVE enumeration member 1035
I_PL_MEMBER_TYPE_TIMBER_BEAM enumeration member 28	I_PS_CASE_CODE_COMBINATIONS enumeration member 1035
I_PL_MEMBER_TYPE_TIMBER_COLUMN enumeration member 28	I_PS_CASE_COMBINATIONS enumeration member 1035
I_PL_MEMBER_TYPE_TIMBER_MEMBER enumeration member 28	I_PS_CASE_SIMPLE_CASES enumeration member 1035
I_PL_PANEL_CALC_MODEL_CURTAIN_WALL enumeration member 28	I_PS_NODE_CALC_NODES enumeration member 1035
I_PL_PANEL_CALC_MODEL_DECK_SLAB enumeration member 28	I_PS_NODE_SUPPORTED enumeration member 1035
I_PL_PANEL_CALC_MODEL_SHELL enumeration member 28	I_PS_NODE_USER_NODES enumeration member 1035
I_PL_PANEL_CALC_MODEL_SLAB_FLEXIBLE_DIAPHRAGM enumeration member 28	I_PSF_ANF enumeration member 1349
I_PL_PANEL_CALC_MODEL_SLAB_RIGID_DIAPHRAGM enumeration member 28	I_PSF_DWG enumeration member 1349
I_PL_PANEL_CALC_MODEL_SLAB_XY_DIAPHRAGM enumeration member 28	I_PSF_DXF enumeration member 1349
I_PL_PANEL_REINFORCEMENT_RC_FLOOR enumeration member 28	I_PSF_DXF_V14 enumeration member 1349
I_PL_PANEL_REINFORCEMENT_RC_SHELL enumeration member 28	I_PSF_RTID enumeration member 1349
I_PL_PANEL_REINFORCEMENT_RC_WALL enumeration member 28	I_PSF_RTID_NORESULTS enumeration member 1349
I_PLT_BEAM_LOAD_AXIAL_FORCE enumeration member 1160	I_PSF_S enumeration member 1349
I_PLT_BEAM_LOAD_FORCE enumeration member 1160	I_PSF_SAT enumeration member 1349
I_PLT_BEAM_LOAD_MOMENT enumeration member 1160	I_PSF_STP_CIM enumeration member 1349
I_PLT_BEAM_LOAD_TORSION_MOMENT enumeration member 1160	I_PSF_STP_DSTV enumeration member 1349
I_PLT_BEAM_LOAD_UNIFORM_TORSION_MOMENT enumeration member 1160	I_PSF_STR enumeration member 1349
I_POD_UX_MINUS enumeration member 392	I_PSF_WRL enumeration member 1349
I_POD_UX_PLUS enumeration member 392	I_PSFT_FRAME enumeration member 1507
I_POD_UY_MINUS enumeration member 392	I_PSFT_NONE enumeration member 1507
I_POD_UY_PLUS enumeration member 392	I_PSFT_SEPARATION_LINE enumeration member 1507
I_POD_UZ_MINUS enumeration member 392	I_PSO_LANDSCAPE enumeration member 1507
I_POD_UZ_PLUS enumeration member 392	I_PSO_PORTAIT enumeration member 1507
	I_PSTL_BEGINNING enumeration member 1509
	I_PSTL_END enumeration member 1509
	I_PT_AXISYMMETRIC enumeration member 1333
	I_PT_BUILDING enumeration member 1333
	I_PT_CONCRETE_BEAM enumeration member 1333
	I_PT_CONCRETE_COLUMN enumeration member 1333
	I_PT_CONCRETE_DEEP_BEAM enumeration member 1333
	I_PT_FOUNDATION enumeration member 1333
	I_PT_FRAME_2D enumeration member 1333
	I_PT_FRAME_3D enumeration member 1333
	I_PT_GRILLAGE enumeration member 1333
	I_PT_PARAMETRIZED enumeration member 1333

I_PT_PLANE_DEFORIFICATION enumeration member 1333	I_RPT_BAR_ELEMENT_DIV_COUNT enumeration member 1019
I_PT_PLANE_STRESS enumeration member 1333	I_RPT_BAR_ELEMENT_DIV_DISCONTINUITY enumeration member 1019
I_PT_PLATE enumeration member 1333	I_RPT_BAR_ELEMENT_DIV_DISCONTINUITY_OFFSET enumeration member 1019
I_PT_SECTION enumeration member 1333	I_RPT_BAR_ELEMENT_DIV_POINT enumeration member 1019
I_PT_SHELL enumeration member 1333	I_RPT_BAR_RELATIVE_POINT enumeration member 1019
I_PT_STEEL_CONNECTION enumeration member 1333	I_RPT_BUFFER_CHUNK_SIZE enumeration member 1019
I_PT_TRUSS_2D enumeration member 1333	I_RPT_CALC_POINT enumeration member 1019
I_PT_TRUSS_3D enumeration member 1333	I_RPT_DIR_X enumeration member 1019
I_PT_VOLUMETRIC enumeration member 1333	I_RPT_DIR_X_DEFTYPE enumeration member 1019
I_PVT_INTEGER enumeration member 2190	I_RPT_ELEMENT enumeration member 1019
I_PVT_REAL enumeration member 2190	I_RPT_ITERATE_LOAD_CASES enumeration member 1019
I_PVT_TEXT enumeration member 2190	I_RPT_LAYER enumeration member 1019
I_QO_DISCARD_CHANGES enumeration member 1537	I_RPT_LAYER_ARBITRARY_VALUE enumeration member 1019
I_QO_PROMPT_TO_SAVE_CHANGES enumeration member 1537	I_RPT_LINEAR_SUPPORTS enumeration member 1019
I_RBT_D1_BOTTOM enumeration member 1743	I_RPT_LOAD_CASE enumeration member 1019
I_RBT_D1_TOP enumeration member 1743	I_RPT_LOAD_CASE_CMPNT enumeration member 1019
I_RBT_D2_BOTTOM enumeration member 1743	I_RPT_MAX_BUFFER_SIZE enumeration member 1019
I_RBT_D2_TOP enumeration member 1743	I_RPT_MODE enumeration member 1019
I_RCL_LOAD_CASE_ACCIDENTAL enumeration member 1207	I_RPT_MODE_CMB enumeration member 1019
I_RCL_LOAD_CASE DESIGN enumeration member 1207	I_RPT_MULTI_THREADS enumeration member 1019
I_RCL_LOAD_CASE_ACC enumeration member 1207	I_RPT_NODE enumeration member 1019
I_RCL_LOAD_CASE_EXPLOITATION enumeration member 1207	I_RPT_PANEL enumeration member 1019
I_RCL_LOAD_CASE_PERMANENT enumeration member 1207	I_RPT_PANEL_PART enumeration member 1019
I_RCL_LOAD_CASE_SEISMIC enumeration member 1207	I_RPT_REDUCED_CUT_POS enumeration member 1019
I_RCL_LOAD_CASE_SNOW enumeration member 1207	I_RPT_REINFORCE_CALC_METHOD enumeration member 1019
I_RCL_LOAD_CASE_TEMPERATURE enumeration member 1207	I_RPT_RESULT_POINT_COORDINATES enumeration member 1019
I_RCL_LOAD_CASE_TEST enumeration member 1207	I_RPT_SMOOTHING enumeration member 1019
I_RCL_LOAD_CASE_WIND enumeration member 1207	I_RPT_STOREY enumeration member 1019
I_RCM_ANALYTICAL enumeration member 951	I_RPT_THREAD_COUNT enumeration member 1019
I_RCM_NEN enumeration member 951	I_RQRT_DONE enumeration member 1027
I_RCM_WOOD_ARMER enumeration member 951	I_RQRT_MORE_AVAILABLE enumeration member 1027
I_RIT_BLANK_PAGE enumeration member 1510	I_RST_AVAILABLE enumeration member 1027
I_RIT_PAGE_TEMPLATE enumeration member 1510	I_RST_NONE enumeration member 1027
I_RIT_VIEW enumeration member 1510	I_RST_OUT_OF_DATE enumeration member 1027
I_RPT_BAR enumeration member 1019	I_SAAXAT_FREQUENCY enumeration member 313
I_RPT_BAR_DIV_COUNT enumeration member 1019	I_SAAXAT_PERIOD enumeration member 313
I_RPT_BAR_DIV_POINT enumeration member 1019	I_SAAXAT_PULSATION enumeration member 313
I_RPT_BAR_ELEMENT enumeration member 1019	I_SACT_RPA_88_1 enumeration member 262

I\_SACT\_RPA\_88\_2 enumeration member 262  
I\_SACT\_RPA\_88\_3 enumeration member 262  
I\_SACT\_RPA\_88\_4 enumeration member 262  
I\_SACT\_RPA\_88\_5 enumeration member 262  
I\_SACT\_RPA\_88\_6 enumeration member 262  
I\_SACT\_RPA\_88\_7 enumeration member 262  
I\_SACT\_RPA\_88\_8 enumeration member 262  
I\_SAD\_ITALY\_ORDINANZA\_HORIZONTAL enumeration member 276  
I\_SAD\_ITALY\_ORDINANZA\_VERTICAL enumeration member 276  
I\_SADT\_CHINESE\_A enumeration member 256  
I\_SADT\_CHINESE\_B enumeration member 256  
I\_SADT\_HORIZONTAL enumeration member 258  
I\_SADT\_PS\_69\_AVERAGE enumeration member 259  
I\_SADT\_PS\_69\_NORMAL enumeration member 259  
I\_SADT\_PS\_69\_WEAK enumeration member 259  
I\_SADT\_VERTICAL enumeration member 258  
I\_SAET\_CHINESE\_FAR enumeration member 257  
I\_SAET\_CHINESE\_NEAR enumeration member 257  
I\_SAGCT\_EAK\_2000\_ALPHA enumeration member 265  
I\_SAGCT\_EAK\_2000\_BETA enumeration member 265  
I\_SAGCT\_EAK\_2000\_DELTA enumeration member 265  
I\_SAGCT\_EAK\_2000\_GAMMA enumeration member 265  
I\_SAGT\_ARBITRARY enumeration member 1518  
I\_SAGT\_CARTESIAN enumeration member 1518  
I\_SAGT\_CYLINDRICAL enumeration member 1518  
I\_SAICT\_100\_92\_I enumeration member 259  
I\_SAICT\_100\_92\_II enumeration member 259  
I\_SAICT\_100\_92\_III enumeration member 259  
I\_SAICT\_100\_92\_IV enumeration member 259  
I\_SAIFT\_EAK\_2000\_SIGMA1 enumeration member 264  
I\_SAIFT\_EAK\_2000\_SIGMA2 enumeration member 264  
I\_SAIFT\_EAK\_2000\_SIGMA3 enumeration member 264  
I\_SAIFT\_EAK\_2000\_SIGMA4 enumeration member 264  
I\_SAIT\_CHINESE\_6 enumeration member 256  
I\_SAIT\_CHINESE\_7 enumeration member 256  
I\_SAIT\_CHINESE\_8 enumeration member 256  
I\_SAIT\_CHINESE\_9 enumeration member 256  
I\_SALT\_123 enumeration member 1513  
I\_SALT\_ABC enumeration member 1513  
I\_SALT\_DEFINE enumeration member 1513  
I\_SALT\_VALUE enumeration member 1513  
I\_SALT\_VARIOUS enumeration member 1513  
I\_SAMPC\_SQUARE\_ROOT\_OF\_SUM\_SQUARES enumeration member 1571  
I\_SAMPC\_SUM\_ABSOLUTE\_VALUES enumeration member 1571  
I\_SAOYAT\_ACCELERATION enumeration member 314  
I\_SAOYAT\_EXCITATION enumeration member 314  
I\_SAOYAT\_VELOCITY enumeration member 314  
I\_SAPCT\_DM\_16\_1\_96\_1 enumeration member 259  
I\_SAPCT\_DM\_16\_1\_96\_12 enumeration member 259  
I\_SAPCT\_DM\_16\_1\_96\_14 enumeration member 259  
I\_SAS\_ITALY\_ORDINANZA\_DESIGN enumeration member 276  
I\_SAS\_ITALY\_ORDINANZA\_ELASTIC enumeration member 276  
I\_SASC\_RPS\_2000\_I enumeration member 267  
I\_SASC\_RPS\_2000\_II enumeration member 267  
I\_SASCT\_IBC\_2000\_A enumeration member 264  
I\_SASCT\_IBC\_2000\_B enumeration member 264  
I\_SASCT\_IBC\_2000\_C enumeration member 264  
I\_SASCT\_IBC\_2000\_D enumeration member 264  
I\_SASCT\_IBC\_2000\_E enumeration member 264  
I\_SASCT\_IBC\_2000\_F enumeration member 264  
I\_SASCT\_IBC\_2006\_A enumeration member 283  
I\_SASCT\_IBC\_2006\_B enumeration member 283  
I\_SASCT\_IBC\_2006\_C enumeration member 283  
I\_SASCT\_IBC\_2006\_D enumeration member 283  
I\_SASCT\_IBC\_2006\_E enumeration member 283  
I\_SASCT\_IBC\_2006\_F enumeration member 283  
I\_SAST\_AFPS\_90\_A enumeration member 254  
I\_SAST\_AFPS\_90\_B enumeration member 254  
I\_SAST\_AFPS\_90\_C enumeration member 254  
I\_SAST\_AFPS\_90\_S0 enumeration member 254  
I\_SAST\_AFPS\_90\_S1 enumeration member 254  
I\_SAST\_AFPS\_90\_S2 enumeration member 254  
I\_SAST\_AFPS\_90\_S3 enumeration member 254  
I\_SAST\_CHINESE\_BRIDGES enumeration member 255  
I\_SAST\_CHINESE\_BUILDINGS enumeration member 255  
I\_SAST\_CHINESE\_HARBOR\_BUILDINGS enumeration member 255  
I\_SAST\_CHINESE\_I enumeration member 256  
I\_SAST\_CHINESE\_II enumeration member 256  
I\_SAST\_CHINESE\_III enumeration member 256  
I\_SAST\_CHINESE\_IV enumeration member 256

I_SAST_CHINESE_SPECIAL enumeration member 255	I_SAST_TURKISH_23098_Z2 enumeration member 263
I_SAST_CIRSOC_103_A enumeration member 258	I_SAST_TURKISH_23098_Z3 enumeration member 263
I_SAST_CIRSOC_103_A0 enumeration member 258	I_SAST_TURKISH_23098_Z4 enumeration member 263
I_SAST_CIRSOC_103_B enumeration member 258	I_SAST_UBC_97_A enumeration member 264
I_SAST_CIRSOC_103_I enumeration member 257	I_SAST_UBC_97_B enumeration member 264
I_SAST_CIRSOC_103_II enumeration member 257	I_SAST_UBC_97_C enumeration member 264
I_SAST_CIRSOC_103_III enumeration member 257	I_SAST_UBC_97_Sa enumeration member 263
I_SAST_DIMENSIONING enumeration member 255	I_SAST_UBC_97_Sb enumeration member 263
I_SAST_ELASITC enumeration member 255	I_SAST_UBC_97_Sc enumeration member 263
I_SAST_ITALY_ORDINANZA_A enumeration member 275	I_SAST_UBC_97_Sd enumeration member 263
I_SAST_ITALY_ORDINANZA_B enumeration member 275	I_SAST_UBC_97_Se enumeration member 263
I_SAST_ITALY_ORDINANZA_C enumeration member 275	I_SAST_UBC_97_Sf enumeration member 263
I_SAST_ITALY_ORDINANZA_D enumeration member 275	I_SAUT_RPA_2003_1A enumeration member 270
I_SAST_ITALY_ORDINANZA_E enumeration member 275	I_SAUT_RPA_2003_1B enumeration member 270
I_SAST_PS_69_FLEXIBLE enumeration member 260	I_SAUT_RPA_2003_2 enumeration member 270
I_SAST_PS_69_RIGID enumeration member 260	I_SAUT_RPA_2003_3 enumeration member 270
I_SAST_PS_92_2008_B enumeration member 289	I_SAUT_RPA_88_1 enumeration member 261
I_SAST_PS_92_2008_C enumeration member 289	I_SAUT_RPA_88_2 enumeration member 261
I_SAST_PS_92_2008_D enumeration member 289	I_SAUT_RPA_88_3 enumeration member 261
I_SAST_PS_92_2008_ENVELOPE enumeration member 289	I_SAVT_ERRORS_AND_WARNINGS enumeration member 1564
I_SAST_PS_92_2008_S0 enumeration member 289	I_SAVT_ERRORS_ONLY enumeration member 1564
I_SAST_PS_92_2008_S1 enumeration member 289	I_SAVT_NONE enumeration member 1564
I_SAST_PS_92_2008_S2 enumeration member 289	I_SAQT_AFPS_90_IA enumeration member 250
I_SAST_PS_92_2008_S3 enumeration member 289	I_SAQT_AFPS_90_IB enumeration member 250
I_SAST_PS_92_B enumeration member 260	I_SAQT_AFPS_90_II enumeration member 250
I_SAST_PS_92_C enumeration member 260	I_SAQT_AFPS_90_III enumeration member 250
I_SAST_PS_92_D enumeration member 260	I_SAQT_CIRSOC_103_0 enumeration member 251
I_SAST_PS_92_ENVELOPE enumeration member 261	I_SAQT_CIRSOC_103_1 enumeration member 251
I_SAST_PS_92_S0 enumeration member 261	I_SAQT_CIRSOC_103_2 enumeration member 251
I_SAST_PS_92_S1 enumeration member 261	I_SAQT_CIRSOC_103_3 enumeration member 251
I_SAST_PS_92_S2 enumeration member 261	I_SAQT_CIRSOC_103_4 enumeration member 251
I_SAST_PS_92_S3 enumeration member 261	I_SAQT_EAK_2000_I enumeration member 253
I_SAST_RPA_2003_S1 enumeration member 270	I_SAQT_EAK_2000_II enumeration member 253
I_SAST_RPA_2003_S2 enumeration member 270	I_SAQT_EAK_2000_III enumeration member 253
I_SAST_RPA_2003_S3 enumeration member 270	I_SAQT_EAK_2000_IV enumeration member 253
I_SAST_RPA_2003_S4 enumeration member 270	I_SAQT_ITALY_ORDINANZA_1 enumeration member 275
I_SAST_RPA_88_FLEXIBLE enumeration member 262	I_SAQT_ITALY_ORDINANZA_2 enumeration member 275
I_SAST_RPA_88_RIGID enumeration member 262	I_SAQT_ITALY_ORDINANZA_3 enumeration member 275
I_SAST_RPS_2000_S1 enumeration member 267	I_SAQT_ITALY_ORDINANZA_4 enumeration member 275
I_SAST_RPS_2000_S2 enumeration member 267	I_SAQT_P_100_92_A enumeration member 251
I_SAST_RPS_2000_S3 enumeration member 267	I_SAQT_P_100_92_B enumeration member 251
I_SAST_TURKISH_23098_Z1 enumeration member 263	I_SAQT_P_100_92_C enumeration member 251

I_SAQT_P_100_92_D enumeration member 251	I_SOORV_LOCAL enumeration member 51
I_SAQT_P_100_92_E enumeration member 251	I_SOORV_PX enumeration member 51
I_SAQT_P_100_92_F enumeration member 251	I_SOORV_PY enumeration member 51
I_SAQT_PS_92_2008_2 enumeration member 289	I_SOORV_PZ enumeration member 51
I_SAQT_PS_92_2008_3 enumeration member 289	I_SRMDT_LAST_CALC_MODE enumeration member 291
I_SAQT_PS_92_2008_4 enumeration member 289	I_SRMDT_LIMIT_FREQUENCY enumeration member 291
I_SAQT_PS_92_2008_5 enumeration member 289	I_SRPD_AUTOMATIC enumeration member 1576
I_SAQT_PS_92_IA enumeration member 252	I_SRPD_DIR_X enumeration member 1576
I_SAQT_PS_92_IB enumeration member 252	I_SRPD_DIR_Y enumeration member 1576
I_SAQT_PS_92_II enumeration member 252	I_SRPD_DIR_Z enumeration member 1576
I_SAQT_PS_92_III enumeration member 252	I_SRS_SLAB_SEG_ENTIRE enumeration member 1220
I_SAQT_RPA_2003_I enumeration member 270	I_SRS_SLAB_SEG_SINGLE enumeration member 1220
I_SAQT_RPA_2003_Ila enumeration member 270	I_SSLDM_AUTOMATIC enumeration member 450
I_SAQT_RPA_2003_Ilb enumeration member 270	I_SSLDM_BY_STOREY enumeration member 450
I_SAQT_RPA_2003_III enumeration member 270	I_SSLDM_BY_Z_COORDINATE enumeration member 450
I_SAQT_RPA_88_I enumeration member 252	I_SSPT_AS_1170_4_1_100 enumeration member 467
I_SAQT_RPA_88_II enumeration member 252	I_SSPT_AS_1170_4_1_1000 enumeration member 467
I_SAQT_RPA_88_III enumeration member 252	I_SSPT_AS_1170_4_1_1500 enumeration member 467
I_SAQT_RPS_2000_1 enumeration member 267	I_SSPT_AS_1170_4_1_20 enumeration member 467
I_SAQT_RPS_2000_2 enumeration member 267	I_SSPT_AS_1170_4_1_200 enumeration member 467
I_SAQT_RPS_2000_3 enumeration member 267	I_SSPT_AS_1170_4_1_2000 enumeration member 467
I_SAQT_TURKISH_23098_1 enumeration member 253	I_SSPT_AS_1170_4_1_25 enumeration member 467
I_SAQT_TURKISH_23098_2 enumeration member 253	I_SSPT_AS_1170_4_1_250 enumeration member 467
I_SAQT_TURKISH_23098_3 enumeration member 253	I_SSPT_AS_1170_4_1_2500 enumeration member 467
I_SAQT_TURKISH_23098_4 enumeration member 253	I_SSPT_AS_1170_4_1_50 enumeration member 467
I_SAQT_UBC_97_1 enumeration member 252	I_SSPT_AS_1170_4_1_500 enumeration member 467
I_SAQT_UBC_97_2A enumeration member 252	I_SSPT_AS_1170_4_1_800 enumeration member 467
I_SAQT_UBC_97_2B enumeration member 252	I_SSSC_AS_1170_4_A enumeration member 466
I_SAQT_UBC_97_3 enumeration member 252	I_SSSC_AS_1170_4_B enumeration member 466
I_SAQT_UBC_97_4 enumeration member 252	I_SSSC_AS_1170_4_C enumeration member 466
I_SCFT_FIXED enumeration member 522	I_SSSC_AS_1170_4_D enumeration member 466
I_SCFT_PINNED enumeration member 522	I_SSSC_AS_1170_4_E enumeration member 466
I_SCFT_ROLLER enumeration member 522	I_SSSC_EC_8_A enumeration member 462
I_SCUT_COPY_TO_CLIPBOARD enumeration member 1452	I_SSSC_EC_8_B enumeration member 462
I_SCUT_CURRENT_VIEW enumeration member 1452	I_SSSC_EC_8_C enumeration member 462
I_SCUT_UPDATED_UPON_PRINTING enumeration member 1452	I_SSSC_EC_8_D enumeration member 462
I_SCUT_UPDATED_WHOLE_STRUCTURE enumeration member 1452	I_SSSC_EC_8_E enumeration member 462
I_SEET_COLUMN enumeration member 518	I_SSSC_EC_8_ENVELOPE enumeration member 462
I_SEET_WALL enumeration member 518	I_SSSCT_ASCE_7_10_A enumeration member 454
I_SMSM_MDA enumeration member 1565	I_SSSCT_ASCE_7_10_B enumeration member 454
I_SMSM_NDM enumeration member 1565	I_SSSCT_ASCE_7_10_C enumeration member 454
	I_SSSCT_ASCE_7_10_D enumeration member 454

I_SSCT_ASCE_7_10_E enumeration member 454	I_SWCECGT_TYPE_II enumeration member 146
I_SSCT_ASCE_7_10_F enumeration member 454	I_SWCECGT_TYPE_III enumeration member 146
I_SSST_AS_1170_4_BRACED_STEEL_FRAMES enumeration member 467	I_SWCECGT_TYPE_IV enumeration member 146
I_SSST_AS_1170_4_CONCRETE_FRAMES enumeration member 467	I_SWCECST_TYPE_I enumeration member 146
I_SSST_AS_1170_4_OTHER_STRUCTURES enumeration member 467	I_SWCECST_TYPE_II enumeration member 146
I_SSST_AS_1170_4_STEEL_FRAMES enumeration member 467	I_SWCECST_TYPE_III enumeration member 146
I_SSST_ASCE_7_10_BRACED_STEEL_FRAMES enumeration member 458	I_SWCECST_TYPE_IV enumeration member 146
I_SSST_ASCE_7_10_CONCRETE_FRAMES enumeration member 458	I_SWCECST_TYPE_V enumeration member 146
I_SSST_ASCE_7_10_OTHER_STRUCTURE_SYSTEM enumeration member 458	I_SWCECST_TYPE_VI enumeration member 146
I_SSST_ASCE_7_10_STEEL_FRAMES enumeration member 458	I_SWCFRST_ACCIDENTAL enumeration member 129
I_SSST_ASCE_7_10_OTHER_STRUCTURE_SYSTEM enumeration member 458	I_SWCFRST_FOLDED_OR_CORRUGATED enumeration member 129
I_SSST_ASCE_7_10_STEEL_FRAMES enumeration member 458	I_SWCFRST_NORMAL enumeration member 129
I_SSST_ASCE_7_10_STEEL_FRAMES enumeration member 458	I_SWCFRST_NORMAL_AND_ACCIDENTAL enumeration member 129
I_SSST_EC_8_BRACED_STEEL_FRAMES enumeration member 462	I_SWCFRST_RIBBED enumeration member 129
I_SSST_EC_8_CONCRETE_FRAMES enumeration member 462	I_SWCFRST_SMOOTH_OR_CORRUGATED enumeration member 129
I_SSST_EC_8_OTHER_STRUCTURE_SYSTEM enumeration member 462	I_SWCFRWS_EXPOSED enumeration member 128
I_SSST_EC_8_STEEL_FRAMES enumeration member 462	I_SWCFRWS_NORMAL enumeration member 128
I_SSST_EC_8_Type1 enumeration member 463	I_SWCFRWS_OBSCURED enumeration member 128
I_SSST_EC_8_Type2 enumeration member 463	I_SWCFRWT_EXTREME enumeration member 128
I_SST_SURFACE_PLAIN enumeration member 1256	I_SWCFRWT_NORMAL enumeration member 128
I_SST_SURFACE_RIBBED enumeration member 1256	I_SWCPLSZ_I enumeration member 109
I_SSTBM_APPROXIMATE_BY_CODE enumeration member 451	I_SWCPLSZ_II enumeration member 109
I_SSTBM_PRECISE_MODAL enumeration member 451	I_SWCPLSZ_III enumeration member 109
I_SSTBM_PRECISE_MODAL_WITH_MASS enumeration member 451	I_SWCPLSZ_IV enumeration member 109
I_SSTBM_USER_DEFINED enumeration member 451	I_SWCPWPDT_CONSTANT enumeration member 129
I_STR_PT_DOUBLE enumeration member 1273	I_SWCPWPDT_VARIABLE enumeration member 129
I_STR_PT_DOUBLE_3 enumeration member 1273	I_SWCPLWS_A enumeration member 108
I_STR_PT_GROUP enumeration member 1273	I_SWCPLWS_B enumeration member 108
I_STR_PT_INTEGER enumeration member 1273	I_SWCPLWS_C enumeration member 108
I_STR_PT_SECTION enumeration member 1273	I_SWCPLWZ_I enumeration member 108
I_STR_PT_SELECTION enumeration member 1273	I_SWCPLWZ_II enumeration member 108
I_STR_PT_TEXT enumeration member 1273	I_SWCPLWZ_IA enumeration member 108
I_STR_PT_TEXT_LIST enumeration member 1273	I_SWCPLWZ_IB enumeration member 108
I_SWCECCT_TYPE_I enumeration member 147	I_SWCPLWZ_III enumeration member 108
I_SWCECCT_TYPE_II enumeration member 147	I_TCF_BAR_DEFL_DEFLECTIONS enumeration member 1355
I_SWCECCT_TYPE_III enumeration member 147	I_TCF_BAR_DEFL_DISPLACEMENTS enumeration member 1355
I_SWCECGT_TYPE_I enumeration member 146	I_TCF_BAR_DEFL_MAX_DEFLECTIONS enumeration member 1355
	I_TCF_CASE_INFO enumeration member 1355
	I_TCF_COEXISTENT_VALUES enumeration member 1355

I_TCF_COMB_CODE_COMPONENTS enumeration member 1355	I_TDT_CODE_GROUPS enumeration member 1354
I_TCF_COMB_DEFINITIONS enumeration member 1355	I_TDT_DEFAULT enumeration member 1354
I_TCF_COMB_MOVING_LOADS enumeration member 1355	I_TDT_DISPLACEMENTS enumeration member 1354
I_TCF_COMB_TIME_HISTORY_COMPONENTS enumeration member 1355	I_TDT_ENVELOPE enumeration member 1354
I_TCF_DETAILS enumeration member 1355	I_TDT_EXTREMES enumeration member 1354
I_TCF_EXTREME_COMBINATIONS enumeration member 1355	I_TDT_FE enumeration member 1354
I_TCF_FE_DIR_X_AXIS enumeration member 1355	I_TDT_INFO enumeration member 1354
I_TCF_FE_DIR_Y_AXIS enumeration member 1355	I_TDT_MATERIAL enumeration member 1354
I_TCF_FE_DIR_Z_AXIS enumeration member 1355	I_TDT_MEMBERS enumeration member 1354
I_TCF_FE_IN_ELEMENT_CENTERS enumeration member 1355	I_TDT_NODE enumeration member 1354
I_TCF_FE_IN_ELENODE enumeration member 1355	I_TDT_PANEL enumeration member 1354
I_TCF_FE_IN_NODES enumeration member 1355	I_TDT_PARA enumeration member 1354
I_TCF_FE_LAYER_LOWER enumeration member 1355	I_TDT_REDUCED_FORCES enumeration member 1354
I_TCF_FE_LAYER_MAX enumeration member 1355	I_TDT_TOTAL enumeration member 1354
I_TCF_FE_LAYER_MAX_ABSOLUTE enumeration member 1355	I_TDT_VALUES enumeration member 1354
I_TCF_FE_LAYER_MIDDLE enumeration member 1355	I_TDT_VOLUME enumeration member 1354
I_TCF_FE_LAYER_MIN enumeration member 1355	I_THAM_HHT enumeration member 366
I_TCF_FE_LAYER_UPPER enumeration member 1355	I_THAM_MODAL_DECOMPOSITION enumeration member 366
I_TCF_GLOBAL_ANALYSIS_RESULTS_NPOINTS enumeration member 1355	I_THAM_NEWMARK enumeration member 366
I_TCF_GLOBAL_ANALYSIS_RESULTS_RELATIVE enumeration member 1355	I_THAM_NEWMARK_ACCELERATION enumeration member 366
I_TCF_GLOBAL_COORDINATES enumeration member 1355	I_THT_CONSTANT enumeration member 723
I_TCF_LOCAL_COORDINATES enumeration member 1355	I_THT_VARIABLE_ALONG_LINE enumeration member 723
I_TCF_LOCALS enumeration member 1355	I_THT_VARIABLE_ON_PLANE enumeration member 723
I_TCF_ORDER_BY_CASE enumeration member 1355	I_TMV_D_XXXX enumeration member 735
I_TCF_PROP_STD_TAPERED_SECTION enumeration member 1355	I_TMV_D_XXYY enumeration member 735
I_TCF_REAC_APPLIED_FORCES_SUM enumeration member 1355	I_TMV_D_XYXY enumeration member 735
I_TCF_REAC_DDC enumeration member 1355	I_TMV_D_YYYY enumeration member 735
I_TCF_REAC_EQUILIBRIUM_PRECISION enumeration member 1355	I_TMV_H_XX enumeration member 735
I_TCF_REAC_RESIDUUM enumeration member 1355	I_TMV_H_YY enumeration member 735
I_TCF_REAC_VALUES enumeration member 1355	I_TMV_K_XXXX enumeration member 735
I_TCF_REINF_IN_ELEMENT_CENTERS enumeration member 1355	I_TMV_K_XXYY enumeration member 735
I_TCF_REINF_IN_NODES enumeration member 1355	I_TMV_K_XYXY enumeration member 735
I_TDT_BAR enumeration member 1354	I_TMV_K_YYYY enumeration member 735
I_TDT_BRACKET enumeration member 1354	I_TO_COPY enumeration member 1043
I_TDT_CABLE enumeration member 1354	I_TO_COPY_WITH_DRAG enumeration member 1043
	I_TO_MOVE enumeration member 1043
	I_TODT_AUTOMATIC enumeration member 724
	I_TODT_DIR_X enumeration member 724
	I_TODT_DIR_Y enumeration member 724
	I_TODT_DIR_Z enumeration member 724
	I_TODT_VECTOR enumeration member 724

I_TOT_BIDIR_BOX_FLOOR enumeration member 723	I_TT_OFFSETS enumeration member 1352
I_TOT_DOUBLE_SIDED_UNIDIR_RIBS enumeration member 723	I_TT_ORTHOTROPIC enumeration member 717
I_TOT_GRILLAGE enumeration member 723	I_TT_PANELS enumeration member 1352
I_TOT_HOLLOW_CORE_SLAB enumeration member 723	I_TT_PLASTIC_HISTORY enumeration member 1352
I_TOT_MATERIAL enumeration member 723	I_TT_PROPERTIES enumeration member 1352
I_TOT_ONE_SIDED_BIDIR_RIBS enumeration member 723	I_TT_PSEUDOSTATIC enumeration member 1352
I_TOT_ONE_SIDED_UNIDIR_RIBS enumeration member 723	I_TTREACTIONS enumeration member 1352
I_TOT_SLAB_COMPOSED_WITH_TRAPEZOID_PLATE enumeration member 723	I_TTREINFORCEMENT enumeration member 1352
I_TOT_SLAB_ON_TRAPEZOID_PLATE enumeration member 723	I_TTSTEEL_MEMBERS enumeration member 1352
I_TOT_UNIDIR_BOX_FLOOR enumeration member 723	I_TTSTOREYS enumeration member 1352
I_TOT_USER enumeration member 723	I_TTSTRESSES enumeration member 1352
I_TRV_GRADIENT_1 enumeration member 49	I_TTSUPPORTS enumeration member 1352
I_TRV_GRADIENT_2 enumeration member 49	I_TTSUPPORTS_LABEL enumeration member 1352
I_TRV_GRADIENT_3 enumeration member 49	I_TUT_MINUS enumeration member 718
I_TRV_T_1 enumeration member 49	I_TUT_NONE enumeration member 718
I_TRV_T_2 enumeration member 49	I_TUT_PLUS enumeration member 718
I_TRV_T_3 enumeration member 49	I_UD_UY enumeration member 540
I_TSCUT_COPY_TO_CLIPBOARD enumeration member 1372	I_UD_UZ enumeration member 540
I_TSCUT_CURRENT_VIEW enumeration member 1372	I_UMT_FORCE enumeration member 1277
I_TSCUT_UPDATED_UPON_PRINTING enumeration member 1372	I_UMT_LENGTH enumeration member 1277
I_TT_BAR_DEFLECTIONS enumeration member 1352	I_UMT_MASS enumeration member 1277
I_TT_BARS enumeration member 1352	I_URT_UNKNOWN enumeration member 1013
I_TT_BUCKLING enumeration member 1352	I_URT_VALUE enumeration member 1013
I_TT_COMBINATIONS enumeration member 1352	I_URT_VALUE_3D enumeration member 1013
I_TT_COREWALLS enumeration member 1352	I_URV_LOCAL_SYSTEM enumeration member 48
I_TT_DYNAMIC enumeration member 1352	I_URV_PROJECTED enumeration member 48
I_TT_FE_RESULTS enumeration member 1352	I_URV_PX enumeration member 48
I_TTFINITE_ELEMENTS enumeration member 1352	I_URV_PY enumeration member 48
I_TT_FORCES enumeration member 1352	I_URV_PZ enumeration member 48
I_TT_GLOBAL_ANALYSIS_BARS enumeration member 1352	I_URV_RELATIVE enumeration member 48
I_TT_HOMOGENEOUS enumeration member 717	I_US_MINUS enumeration member 541
I_TT_INFLUENCE_LINE enumeration member 1352	I_US_NONE enumeration member 541
I_TT_INTERACTION_FORCES enumeration member 1352	I_US_PLUS enumeration member 541
I_TT_LIARIG_LABEL enumeration member 1352	I_UT_ANGLE_ROTATION_DATA enumeration member 1275
I_TT_LOADS enumeration member 1352	I_UT_ANGLE_ROTATION_RESULT enumeration member 1275
I_TT_MASS enumeration member 1352	I_UT_DIAMETER_RC_BASE enumeration member 1275
I_TT_MEASURE enumeration member 1352	I_UT_DIMENSIONLESS_QUALITY enumeration member 1275
I_TT_NODE_DISPLACEMENTS enumeration member 1352	I_UT_DISPLACEMENT enumeration member 1275
I_TT_NODES enumeration member 1352	I_UT_FORCE enumeration member 1275
	I_UT_MASS enumeration member 1275
	I_UT_MOMENT enumeration member 1275

I_UT_REINFORCEMENT AREAS enumeration member 1275	1411
I_UT_RULER enumeration member 1275	I_VDA_ADVANCED_RC_CONTINOUS_FOOTING enumeration member 1411
I_UT_SECTION_DIMENSION enumeration member 1275	I_VDA_ADVANCED_RC_DEEP_BEAM enumeration member 1411
I_UT_SECTION_PROPERTIES enumeration member 1275	I_VDA_ADVANCED_RC_FOUNDATION enumeration member 1411
I_UT_STEEL_CONECTIONS enumeration member 1275	I_VDA_ADVANCED_RC_WALL enumeration member 1411
I_UT_STRESS enumeration member 1275	I_VDA_ADVANCED_RELEASE_CODES enumeration member 1411
I_UT_STRUCTURE_DIMENSION enumeration member 1275	I_VDA_ADVANCED_RELEASE_SYMBOLS enumeration member 1411
I_UT_TEMPERATURE enumeration member 1275	I_VDA_ADVANCED_RIGID_LINK_SURFACE enumeration member 1411
I_UT_WEIGHT enumeration member 1275	I_VDA_ADVANCED_RIGID_LINKS enumeration member 1411
I_VBMRT DESIGN MEMBER LENGTH enumeration member 1423	I_VDA_ADVANCED_STEEL_CONNECTION NAMES enumeration member 1411
I_VBMRT DESIGN_RATIO enumeration member 1423	I_VDA_ADVANCED_STEEL_CONNECTION_NUMBERS enumeration member 1411
I_VBMRT DESIGN_SLEND_LAY enumeration member 1423	I_VDA_ADVANCED_STEEL_CONNECTION_TYPES enumeration member 1411
I_VBMRT DESIGN_SLEND_LAZ enumeration member 1423	I_VDA_ADVANCED_STEEL_CONNECTIONS enumeration member 1411
I_VBMRT NOTHING enumeration member 1423	I_VDA_ADVANCED_STORIES_COLORS enumeration member 1411
I_VBMRT_NTM_FX enumeration member 1423	I_VDA_ADVANCED_STORIES_RESULTS enumeration member 1411
I_VBMRT_NTM_FY enumeration member 1423	I_VDA_ADVANCED_TENSION_COMPRESSION enumeration member 1411
I_VBMRT_NTM_FZ enumeration member 1423	I_VDA_FE_AUXILIARY_OBJECTS enumeration member 1411
I_VBMRT_NTM_MX enumeration member 1423	I_VDA_FE_CHARACTERISTIC_POINTS enumeration member 1411
I_VBMRT_NTM_MY enumeration member 1423	I_VDA_FE_CLADDING_COLORS enumeration member 1411
I_VBMRT_NTM_MZ enumeration member 1423	I_VDA_FE_CLADDING_INTERIOR enumeration member 1411
I_VBMRT_STRESS_FX_AX enumeration member 1423	I_VDA_FE_COLOR_LEGEND enumeration member 1411
I_VBMRT_STRESS_S_MAX enumeration member 1423	I_VDA_FE_CONTOUR_COMPONENTS enumeration member 1411
I_VBMRT_STRESS_S_MAX_MY enumeration member 1423	I_VDA_FE_EDGE_NUMBERS enumeration member 1411
I_VBMRT_STRESS_S_MAX_MZ enumeration member 1423	I_VDA_FE_EMITTERS enumeration member 1411
I_VBMRT_STRESS_S_MIN enumeration member 1423	I_VDA_FE_FE_INTERIOR enumeration member 1411
I_VBMRT_STRESS_S_MIN_MY enumeration member 1423	I_VDA_FEFINITE_ELEMENT_NUMBERS enumeration member 1411
I_VBMRT_STRESS_S_MIN_MZ enumeration member 1423	I_VDA_FEFINITE_ELEMENTS enumeration member 1411
I_VBMRT_STRESS_SHEAR_TY enumeration member 1423	I_VDA_FELoad_DIRECTION enumeration member 1411
I_VBMRT_STRESS_SHEAR_TZ enumeration member 1423	I_VDA_FEMESH_PREVIEW enumeration member 1411
I_VBMRT_STRESS_TORSION_T enumeration member 1423	I_VDA_FEPANEL_COMPLEX_DESC enumeration member
I_VDA_ADVANCED_CABLES enumeration member 1411	
I_VDA_ADVANCED_COMPATIBLE_NODES enumeration member 1411	
I_VDA_ADVANCED_ELASTIC_FOUNDATION enumeration member 1411	
I_VDA_ADVANCED_GEOIMPERFECTIONS enumeration member 1411	
I_VDA_ADVANCED_GEOIMPERFECTIONS_NAME enumeration member 1411	
I_VDA_ADVANCED_NONLINEAR_HINGES enumeration member 1411	
I_VDA_ADVANCED_OFFSETS enumeration member 1411	
I_VDA_ADVANCED_RC_BEAM enumeration member 1411	
I_VDA_ADVANCED_RC_COLUMN enumeration member	

1411	I_VDA_SECTIONS_COLORS enumeration member 1411
I_VDA_FE_PANEL_CONTOURS enumeration member 1411	I_VDA_SECTIONS_MATERIAL enumeration member 1411
I_VDA_FE_PANEL_INTERIOR enumeration member 1411	I_VDA_SECTIONS_MEMBER_TYPE_LEGEND enumeration member 1411
I_VDA_FE_PANEL_NAMES enumeration member 1411	I_VDA_SECTIONS_MEMBER_TYPE_NAME enumeration member 1411
I_VDA_FE_PANEL_NUMBERS enumeration member 1411	I_VDA_SECTIONS_NAME enumeration member 1411
I_VDA_FE_PANEL_REINFORCEMENT enumeration member 1411	I_VDA_SECTIONS_SHAPE enumeration member 1411
I_VDA_FE_PANEL_THICKNESSES enumeration member 1411	I_VDA_SECTIONS_SURFACE enumeration member 1411
I_VDA_FE_THICKNESS enumeration member 1411	I_VDA_SECTIONS_SYMBOLS enumeration member 1411
I_VDA_LOADS_AUTOMATICALLY enumeration member 1411	I_VDA_STRUCTURE_ATTRIBUTE_LABELS enumeration member 1411
I_VDA_LOADS_DISTRIBUTION_REGIONS enumeration member 1411	I_VDA_STRUCTURE_BAR NAMES enumeration member 1411
I_VDA_LOADS_MOVING_ELEMENTS enumeration member 1411	I_VDA_STRUCTURE_BAR_NUMBERS enumeration member 1411
I_VDA_LOADS_MOVING_ROUTE enumeration member 1411	I_VDA_STRUCTURE_EXPLODE_BARS enumeration member 1411
I_VDA_LOADS_MOVING_VEHICLE enumeration member 1411	I_VDA_STRUCTURE_EXPLODE_FE enumeration member 1411
I_VDA_LOADS_PRESSURE_MAP enumeration member 1411	I_VDA_STRUCTURE_GROUP_COLORS enumeration member 1411
I_VDA_LOADS_SYMBOLS_CONCENTRATED enumeration member 1411	I_VDA_STRUCTURE_LOCAL_SYSTEM_BARS enumeration member 1411
I_VDA_LOADS_SYMBOLS_LINEAR enumeration member 1411	I_VDA_STRUCTURE_LOCAL_SYSTEM_FE enumeration member 1411
I_VDA_LOADS_SYMBOLS_PLANAR enumeration member 1411	I_VDA_STRUCTURE_LOCAL_SYSTEM_PANELS enumeration member 1411
I_VDA_LOADS_SYMBOLS_UNIFORM_SIZE enumeration member 1411	I_VDA_STRUCTURE_NODE_NUMBERS enumeration member 1411
I_VDA_LOADS_VALUES enumeration member 1411	I_VDA_STRUCTURE_ONLY_FOR_SELECTED_OBJECTS enumeration member 1411
I_VDA_OTHER_CALC_ELEM_NUMBERS enumeration member 1411	I_VDA_STRUCTURE_STRUCTURE enumeration member 1411
I_VDA_OTHER_CALC_POINT_NUMBERS enumeration member 1411	I_VDA_STRUCTURE_SUPPORT_CODES enumeration member 1411
I_VDA_OTHER_DIMENSION_LINES enumeration member 1411	I_VDA_STRUCTURE_SUPPORT_DETAILED_SYMBOLS enumeration member 1411
I_VDA_OTHER_GRID enumeration member 1411	I_VDA_STRUCTURE_SUPPORT_DIRECTION enumeration member 1411
I_VDA_OTHER_HIDE_INACTIVE enumeration member 1411	I_VDA_STRUCTURE_SUPPORT_SHAPES enumeration member 1411
I_VDA_OTHER_HIDE_NODES enumeration member 1411	I_VDA_STRUCTURE_SUPPORT_SYMBOLS enumeration member 1411
I_VDA_OTHER_OBJECTS_OUT_OF_PLANE enumeration member 1411	I_VDA_VIEW_COLOR enumeration member 1411
I_VDA_OTHER_RULER enumeration member 1411	I_VDA_VIEW_HIDDEN_LINES enumeration member 1411
I_VDA_OTHER_STRUCTURAL_AXIS enumeration member 1411	I_VDA_VIEW_NONE enumeration member 1411
I_VDA_OTHER_STRUCTURAL_AXIS_DESCRIPTION enumeration member 1411	I_VDA_VIEW_QUICK_SHADING enumeration member 1411
I_VDA_SECTIONS_CODE_GROUPS enumeration member 1411	I_VDA_VIEW_SHADING enumeration member 1411

I_VDA_VIEW_SHADING_EDGES enumeration member 1411	I_VDART_REINFORCE_TOP enumeration member 1426
I_VDA_VIEWOGL_ANTYALIASING enumeration member 1411	I_VDART_REINFORCE_TOP_REAL enumeration member 1426
I_VDA_VIEWOGL_BLACK_EDGES enumeration member 1411	I_VDART_STRESS_AXIAL_FX_AX enumeration member 1426
I_VDA_VIEWOGL_COLOR_FACES enumeration member 1411	I_VDART_STRESS_BENDING_S_MAX_MY enumeration member 1426
I_VDA_VIEWOGL_DETAILS enumeration member 1411	I_VDART_STRESS_BENDING_S_MAX_MZ enumeration member 1426
I_VDA_VIEWOGL_DRAW_OUT_OF_SCREEN enumeration member 1411	I_VDART_STRESS_BENDING_S_MIN_MY enumeration member 1426
I_VDA_VIEWOGL_HIDE_INVISIBLE_LINES enumeration member 1411	I_VDART_STRESS_BENDING_S_MIN_MZ enumeration member 1426
I_VDA_VIEWOGL_HIDE_XY_CUTTING_PLANES enumeration member 1411	I_VDART_STRESS_S_MAX enumeration member 1426
I_VDA_VIEWOGL_HIDE_XZ_CUTTING_PLANES enumeration member 1411	I_VDART_STRESS_S_MIN enumeration member 1426
I_VDA_VIEWOGL_HIDE_YZ_CUTTING_PLANES enumeration member 1411	I_VDART_STRESS_SHEAR_TY enumeration member 1426
I_VDA_VIEWOGL_LIGHT enumeration member 1411	I_VDART_STRESS_SHEAR_TZ enumeration member 1426
I_VDA_VIEWOGL_LIGHT_ON_MAPS enumeration member 1411	I_VDART_STRESS_TORSION_T enumeration member 1426
I_VDA_VIEWOGL_REDRAW enumeration member 1411	I_VDATT_GLOBAL_EXTREMES enumeration member 1409
I_VDA_VIEWOGL_TRANSLUCENT enumeration member 1411	I_VDATT_LOCAL_EXTREMES enumeration member 1409
I_VDART_NTM_FX enumeration member 1426	I_VDATT_VALUES enumeration member 1409
I_VDART_NTM_FY enumeration member 1426	I_VDDT_LABELS enumeration member 1421
I_VDART_NTM_FZ enumeration member 1426	I_VDDT_NONE enumeration member 1421
I_VDART_NTM_MX enumeration member 1426	I_VDDT_TEXT enumeration member 1421
I_VDART_NTM_MY enumeration member 1426	I_VDFT_FENCE enumeration member 1421
I_VDART_NTM_MZ enumeration member 1426	I_VDFT_FILLED enumeration member 1421
I_VDART_NTM_UX enumeration member 1426	I_VDPT_IN_PLANE enumeration member 1441
I_VDART_NTM_UY enumeration member 1426	I_VDPT_NORMAL enumeration member 1441
I_VDART_NTM_UZ enumeration member 1426	I_VDRT_DEFORMATION_DEFORIFICATION enumeration member 1420
I_VDART_REINFORCE_BOTTOM enumeration member 1426	I_VDRT_DEFORMATION_EXACT enumeration member 1420
I_VDART_REINFORCE_BOTTOM_REAL enumeration member 1426	I_VDRT_NTM_FX enumeration member 1420
I_VDART_REINFORCE_NUMBER_OF_LOWER_BARS enumeration member 1426	I_VDRT_NTM_FY enumeration member 1420
I_VDART_REINFORCE_NUMBER_OF_UPPER_BARS enumeration member 1426	I_VDRT_NTM_FZ enumeration member 1420
I_VDART_REINFORCE_RATIO enumeration member 1426	I_VDRT_NTM_KY enumeration member 1420
I_VDART_REINFORCE_RATIO_REAL enumeration member 1426	I_VDRT_NTM_KZ enumeration member 1420
I_VDART_REINFORCE_STIRRUP_SPACING enumeration member 1426	I_VDRT_NTM_MX enumeration member 1420
I_VDART_REINFORCE_STIRRUP_SPACING_REAL enumeration member 1426	I_VDRT_NTM_MY enumeration member 1420
	I_VDRT_NTM_MZ enumeration member 1420
	I_VDRTREACTION_DESC enumeration member 1420
	I_VDRTREACTION_DESC_AVERAGE enumeration member 1420
	I_VDRTREACTION_DESC_INTEGRAL enumeration member 1420
	I_VDRTREACTION_FORCES enumeration member 1420
	I_VDRTREACTION_FX enumeration member 1420

I_VDRTREACTIONFY enumeration member 1420	I_VDSDTUNDIFFERENTIATED enumeration member 1422
I_VDRTREACTIONFZ enumeration member 1420	I_VDVTALL enumeration member 1454
I_VDRTREACTIONLINEARSUPPORTS enumeration member 1420	I_VDVTGLOBALEXTREMES enumeration member 1454
I_VDRTREACTIONMOMENTS enumeration member 1420	I_VDVTLOCALEXTREMES enumeration member 1454
I_VDRTREACTIONMX enumeration member 1420	I_VFMCPTMOMENTS enumeration member 1437
I_VDRTREACTIONMY enumeration member 1420	I_VFMCPTNONE enumeration member 1437
I_VDRTREACTIONMZ enumeration member 1420	I_VFMCPTNORMALFORCES enumeration member 1437
I_VDRTREACTIONPSEUDOSTATICFORCES enumeration member 1420	I_VFMCPTSTRESSES enumeration member 1437
I_VDRTREACTIONPSEUDOSTATICMOMENTS enumeration member 1420	I_VFMLSTAUTOMATIC enumeration member 1437
I_VDRTREACTIONRESIDUALFORCES enumeration member 1420	I_VFMLSTCARTESIANALONGVECTOR enumeration member 1437
I_VDRTREACTIONRESIDUALMOMENTS enumeration member 1420	I_VFMLSTCARTESIANALONGX enumeration member 1437
I_VDRTREINFORCEBOTTOM enumeration member 1420	I_VFMLSTCARTESIANALONGY enumeration member 1437
I_VDRTREINFORCEBOTTOMREAL enumeration member 1420	I_VFMLSTCARTESIANALONGZ enumeration member 1437
I_VDRTREINFORCENUMBEROFLOWERBARS enumeration member 1420	I_VFMLSTPOLARINNODE enumeration member 1437
I_VDRTREINFORCENUMBEROFTUBEBARS enumeration member 1420	I_VFMLSTPOLARINPOINT enumeration member 1437
I_VDRTREINFORCERATIO enumeration member 1420	I_VFMLTABSOLUTEMAXIMUM enumeration member 1438
I_VDRTREINFORCERATIOREAL enumeration member 1420	I_VFMLTARBITRARY enumeration member 1438
I_VDRTREINFORCESTIRRUPSPACING enumeration member 1420	I_VFMLTLOWER enumeration member 1438
I_VDRTREINFORCESTIRRUPSPACINGREAL enumeration member 1420	I_VFMLTMAXIMUM enumeration member 1438
I_VDRTREINFORCETOP enumeration member 1420	I_VFMLTMIDDLE enumeration member 1438
I_VDRTREINFORCETOPREAL enumeration member 1420	I_VFMLTMINIMUM enumeration member 1438
I_VDRTSTRESSAXIALFXAX enumeration member 1420	I_VFMLTUPPER enumeration member 1438
I_VDRTSTRESSBENDINGSMAXMY enumeration member 1420	I_VFMRTCOMPLEXMEMBRANEFORCES enumeration member 1430
I_VDRTSTRESSBENDINGSMAXMZ enumeration member 1420	I_VFMRTCOMPLEXMOMENTS enumeration member 1430
I_VDRTSTRESSBENDINGSMINMY enumeration member 1420	I_VFMRTCOMPLEXREINFORCEBOTTOMMXX enumeration member 1430
I_VDRTSTRESSBENDINGSMINMZ enumeration member 1420	I_VFMRTCOMPLEXREINFORCEBOTTOMMYY enumeration member 1430
I_VDRTSTRESSSMAX enumeration member 1420	I_VFMRTCOMPLEXREINFORCETOPMXX enumeration member 1430
I_VDRTSTRESSSMIN enumeration member 1420	I_VFMRTCOMPLEXREINFORCETOPMYY enumeration member 1430
I_VDRTSTRESSSHEARTY enumeration member 1420	I_VFMRTCOMPLEXSTRESSES enumeration member 1430
I_VDRTSTRESSSHEARTZ enumeration member 1420	I_VFMRTDETAILEDDISPLACEMENTXX enumeration member 1430
I_VDRTSTRESTITSION enumeration member 1420	I_VFMRTDETAILEDDISPLACEMENTYY enumeration member 1430
I_VDSDTDIFFERENTIATED enumeration member 1422	I_VFMRTDETAILEDDISPLACEMENTZ enumeration member 1430

I_VFMRT_DETAILED_MEMBRANE_FORCE_XY enumeration member 1430	I_VFMRT_PRINCIPAL_MOMENT_2 enumeration member 1430
I_VFMRT_DETAILED_MEMBRANE_FORCE_YY enumeration member 1430	I_VFMRT_PRINCIPAL_MOMENT_ANGLE enumeration member 1430
I_VFMRT_DETAILED_MOMENT_XX enumeration member 1430	I_VFMRT_PRINCIPAL_SHEAR_FORCE_1_2 enumeration member 1430
I_VFMRT_DETAILED_MOMENT_XY enumeration member 1430	I_VFMRT_PRINCIPAL_SHEAR_STRESS_1_2 enumeration member 1430
I_VFMRT_DETAILED_MOMENT_YY enumeration member 1430	I_VFMRT_PRINCIPAL_STRESS_1 enumeration member 1430
I_VFMRT_DETAILED_ROTATION_XX enumeration member 1430	I_VFMRT_PRINCIPAL_STRESS_1_2 enumeration member 1430
I_VFMRT_DETAILED_ROTATION_YY enumeration member 1430	I_VFMRT_PRINCIPAL_STRESS_2 enumeration member 1430
I_VFMRT_DETAILED_ROTATION_Z enumeration member 1430	I_VFMRT_PRINCIPAL_STRESS_ANGLE enumeration member 1430
I_VFMRT_DETAILED_SHEAR_FORCE_XX enumeration member 1430	I_VFMRT_TOTAL_DISPLACEMENTS enumeration member 1430
I_VFMRT_DETAILED_SHEAR_FORCE_YY enumeration member 1430	I_VFMST_GLOBAL_SMOOTHING enumeration member 1438
I_VFMRT_DETAILED_SHEAR_STRESS_XX enumeration member 1430	I_VFMST_NO_SMOOTHING enumeration member 1438
I_VFMRT_DETAILED_SHEAR_STRESS_YY enumeration member 1430	I_VFMST_SMOOTHING_ACCORDING_TO_SELECTION enumeration member 1438
I_VFMRT_DETAILED_SOIL_REACTION_Z enumeration member 1430	I_VFMST_SMOOTHING_WITH_PANEL enumeration member 1438
I_VFMRT_DETAILED_STRESS_XX enumeration member 1430	I_VGAPT DESIGN MEMBER_LENGTH enumeration member 1446
I_VFMRT_DETAILED_STRESS_XY enumeration member 1430	I_VGAPT DESIGN_RATIO enumeration member 1446
I_VFMRT_DETAILED_STRESS_YY enumeration member 1430	I_VGAPT DESIGN_SLENDERNESS_LAY enumeration member 1446
I_VFMRT_DETAILED_STRESS_Z enumeration member 1430	I_VGAPT DESIGN_SLENDERNESS_LAZ enumeration member 1446
I_VFMRT_GLOBAL_DISPLACEMENT_X enumeration member 1430	I_VGAPT_FORCE_FX enumeration member 1446
I_VFMRT_GLOBAL_DISPLACEMENT_Y enumeration member 1430	I_VGAPT_FORCE_FY enumeration member 1446
I_VFMRT_GLOBAL_DISPLACEMENT_Z enumeration member 1430	I_VGAPT_FORCE_FZ enumeration member 1446
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_1 enumeration member 1430	I_VGAPT_FORCE_KX enumeration member 1446
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_1_2 enumeration member 1430	I_VGAPT_FORCE_KY enumeration member 1446
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_2 enumeration member 1430	I_VGAPT_FORCE_MX enumeration member 1446
I_VFMRT_PRINCIPAL_MEMBRANE_FORCE_ANGLE enumeration member 1430	I_VGAPT_FORCE_MY enumeration member 1446
I_VFMRT_PRINCIPAL_MOMENT_1 enumeration member 1430	I_VGAPT_FORCE_MZ enumeration member 1446
I_VFMRT_PRINCIPAL_MOMENT_1_2 enumeration member 1430	I_VGAPT_STRESS_AXIAL_SFX enumeration member 1446
	I_VGAPT_STRESS_BENDING_SMY enumeration member 1446
	I_VGAPT_STRESS_BENDING_SMZ enumeration member 1446
	I_VGAPT_STRESS_EXTREME enumeration member 1446
	I_VGAPT_STRESS_NORMAL enumeration member 1446
	I_VGAPT_STRESS_SHEAR enumeration member 1446

I_VGAPT_STRESS_TORSION enumeration member 1446	I_VILRT_DETAILED_ROTATION_XX enumeration member 1386
I_VGART_N_POINTS enumeration member 1446	I_VILRT_DETAILED_ROTATION_YY enumeration member 1386
I_VGART_RELATIVE enumeration member 1446	I_VILRT_DETAILED_ROTATION_Z enumeration member 1386
I_VHLDT_COLOR_SIDES_EDGES enumeration member 1415	I_VILRT_DETAILED_SHEAR_FORCE_XX enumeration member 1386
I_VHLDT_HIDDEN_LINES enumeration member 1415	I_VILRT_DETAILED_SHEAR_FORCE_YY enumeration member 1386
I_VHLDT_NONE enumeration member 1415	I_VILRT_DETAILED_SHEAR_STRESS_XX enumeration member 1386
I_VHLDT_QUICK_SHADING_OF_FACES enumeration member 1415	I_VILRT_DETAILED_SHEAR_STRESS_YY enumeration member 1386
I_VHLDT_SHADING enumeration member 1415	I_VILRT_DETAILED_SOIL_REACTION_Z enumeration member 1386
I_VHLDT_SHADING_EDGES enumeration member 1415	I_VILRT_DETAILED_STRESS_XX enumeration member 1386
I_VILLST_AUTOMATIC enumeration member 1387	I_VILRT_DETAILED_STRESS_XY enumeration member 1386
I_VILLST_CARTESIAN_ALONG_VECTOR enumeration member 1387	I_VILRT_DETAILED_STRESS_YY enumeration member 1386
I_VILLST_CARTESIAN_ALONG_X enumeration member 1387	I_VILRT_DETAILED_STRESS_Z enumeration member 1386
I_VILLST_CARTESIAN_ALONG_Y enumeration member 1387	I_VILRT_EXTREME_MEMBRANE_FORCES_1 enumeration member 1386
I_VILLST_CARTESIAN_ALONG_Z enumeration member 1387	I_VILRT_EXTREME_MEMBRANE_FORCES_1_2 enumeration member 1386
I_VILLST_POLAR_IN_NODE enumeration member 1387	I_VILRT_EXTREME_MEMBRANE_FORCES_2 enumeration member 1386
I_VILLST_POLAR_IN_POINT enumeration member 1387	I_VILRT_EXTREME_MEMBRANE_FORCES_ANGLE enumeration member 1386
I_VILLT_ARBITRARY enumeration member 1388	I_VILRT_EXTREME_MOMENTS_1 enumeration member 1386
I_VILLT_LOWER enumeration member 1388	I_VILRT_EXTREME_MOMENTS_1_2 enumeration member 1386
I_VILLT_MIDDLE enumeration member 1388	I_VILRT_EXTREME_MOMENTS_2 enumeration member 1386
I_VILLT_UPPER enumeration member 1388	I_VILRT_EXTREME_MOMENTS_ANGLE enumeration member 1386
I_VILRT_COMPLEX_MEMBRANE_FORCE enumeration member 1386	I_VILRT_EXTREME_SHEAR_FORCE_1_2 enumeration member 1386
I_VILRT_COMPLEX_MOMENT enumeration member 1386	I_VILRT_EXTREME_SHEAR_STRESS_1_2 enumeration member 1386
I_VILRT_COMPLEX_STRESS enumeration member 1386	I_VILRT_EXTREME_STRESS_1 enumeration member 1386
I_VILRT_DETAILED_DISPLACEMENT_XX enumeration member 1386	I_VILRT_EXTREME_STRESS_1_2 enumeration member 1386
I_VILRT_DETAILED_DISPLACEMENT_YY enumeration member 1386	I_VILRT_EXTREME_STRESS_2 enumeration member 1386
I_VILRT_DETAILED_DISPLACEMENT_Z enumeration member 1386	I_VILRT_EXTREME_STRESS_ANGLE enumeration member 1386
I_VILRT_DETAILED_MEMBRANE_FORCE_XX enumeration member 1386	I_VILRT_NODES_fx enumeration member 1386
I_VILRT_DETAILED_MEMBRANE_FORCE_XY enumeration member 1386	I_VILRT_NODES_fy enumeration member 1386
I_VILRT_DETAILED_MEMBRANE_FORCE_YY enumeration member 1386	
I_VILRT_DETAILED_MOMENTS_XX enumeration member 1386	
I_VILRT_DETAILED_MOMENTS_XY enumeration member 1386	
I_VILRT_DETAILED_MOMENTS_YY enumeration member 1386	

I_VILRT_NODES_FZ enumeration member 1386	I_VPI_COMPANY_NAME enumeration member 1483
I_VILRT_NODES_MX enumeration member 1386	I_VPI_COMPANY_TELEPHONE enumeration member 1483
I_VILRT_NODES_MY enumeration member 1386	I_VPI_DATE enumeration member 1483
I_VILRT_NODES_MZ enumeration member 1386	I_VPI_DESIGNER_ADDRESS1 enumeration member 1483
I_VILRT_NODES_RX enumeration member 1386	I_VPI_DESIGNER_ADDRESS2 enumeration member 1483
I_VILRT_NODES_RY enumeration member 1386	I_VPI_DESIGNER_CONTACT enumeration member 1483
I_VILRT_NODES_RZ enumeration member 1386	I_VPI_DESIGNER_EMAIL enumeration member 1483
I_VILRT_NODES_UX enumeration member 1386	I_VPI_DESIGNER_FAX enumeration member 1483
I_VILRT_NODES_UY enumeration member 1386	I_VPI_DESIGNER_NAME enumeration member 1483
I_VILRT_NODES_UZ enumeration member 1386	I_VPI_DESIGNER_TELEPHONE enumeration member 1483
I_VILRT_NTM_FX enumeration member 1386	I_VPI_INVESTOR_ADDRESS1 enumeration member 1483
I_VILRT_NTM_FY enumeration member 1386	I_VPI_INVESTOR_ADDRESS2 enumeration member 1483
I_VILRT_NTM_FZ enumeration member 1386	I_VPI_INVESTOR_CONTACT enumeration member 1483
I_VILRT_NTM_KY enumeration member 1386	I_VPI_INVESTOR_EMAIL enumeration member 1483
I_VILRT_NTM_KZ enumeration member 1386	I_VPI_INVESTOR_FAX enumeration member 1483
I_VILRT_NTM_MX enumeration member 1386	I_VPI_INVESTOR_NAME enumeration member 1483
I_VILRT_NTM_MY enumeration member 1386	I_VPI_INVESTOR_TELEPHONE enumeration member 1483
I_VILRT_NTM_MZ enumeration member 1386	I_VPI_PROJECT_ADDRESS1 enumeration member 1483
I_VILRT_NTM_UX enumeration member 1386	I_VPI_PROJECT_ADDRESS2 enumeration member 1483
I_VILRT_NTM_UZ enumeration member 1386	I_VPI_PROJECT_COMMENT enumeration member 1483
I_VLT_CONCENTRATED enumeration member 348	I_VPI_PROJECT_CONTENT enumeration member 1483
I_VLT_LINEAR enumeration member 348	I_VPI_PROJECT_CREATED enumeration member 1483
I_VLT_SURFACE enumeration member 348	I_VPI_PROJECT_DIRECTORY enumeration member 1483
I_VP_3DXYZ enumeration member 1398	I_VPI_PROJECT_FILE enumeration member 1483
I_VP_XY enumeration member 1398	I_VPI_PROJECT_MODIFIED enumeration member 1483
I_VP_XY_3D enumeration member 1398	I_VPI_PROJECT_NAME enumeration member 1483
I_VP_XZ enumeration member 1398	I_VPI_PROJECT_REFERENCE enumeration member 1483
I_VP_XZ_3D enumeration member 1398	I_VPI_PROJECT_SIZE enumeration member 1483
I_VP_YZ enumeration member 1398	I_VPI_PROJECT_STATISTICS enumeration member 1483
I_VP_YZ_3D enumeration member 1398	I_VPI_PROJECT_STEP enumeration member 1483
I_VPI_ARCHITECT_ADDRESS1 enumeration member 1483	I_VPI_PROJECT_VERSION enumeration member 1483
I_VPI_ARCHITECT_ADDRESS2 enumeration member 1483	I_VPI_REPORT_PAGE_NUMBER enumeration member 1483
I_VPI_ARCHITECT_CONTACT enumeration member 1483	I_VPI_ROBOT_APP enumeration member 1483
I_VPI_ARCHITECT_EMAIL enumeration member 1483	I_VPI_ROBOT_PROVIDER enumeration member 1483
I_VPI_ARCHITECT_FAX enumeration member 1483	I_VPI_ROBOT_VER enumeration member 1483
I_VPI_ARCHITECT_NAME enumeration member 1483	I_VPI_TIME enumeration member 1483
I_VPI_ARCHITECT_TELEPHONE enumeration member 1483	I_VPI_USER_ADDRESS enumeration member 1483
I_VPI_COMPANY_ADDRESS1 enumeration member 1483	I_VPI_USER_NAME enumeration member 1483
I_VPI_COMPANY_ADDRESS2 enumeration member 1483	I_VPI_VERIF_ADDRESS1 enumeration member 1483
I_VPI_COMPANY_CONTACT enumeration member 1483	I_VPI_VERIF_ADDRESS2 enumeration member 1483
I_VPI_COMPANY_EMAIL enumeration member 1483	I_VPI_VERIF_CONTACT enumeration member 1483
I_VPI_COMPANY_FAX enumeration member 1483	I_VPI_VERIF_EMAIL enumeration member 1483

I_VPI_VERIF_FAX enumeration member 1483	I_VVST_BAR_LOADS enumeration member 1399
I_VPI_VERIF_NAME enumeration member 1483	I_VVST_BAR_MAPS enumeration member 1399
I_VPI_VERIF_TELEPHONE enumeration member 1483	I_VVST_BEST enumeration member 1399
I_VRRT_A_MIN enumeration member 1441	I_VVST_BUCKLING enumeration member 1399
I_VRRT_AX enumeration member 1441	I_VVST_COMPATIBILITIES enumeration member 1399
I_VRRT_AX_BOTTOM enumeration member 1441	I_VVST_CtrT enumeration member 1399
I_VRRT_AX_TOP enumeration member 1441	I_VVST_DEFORMATION enumeration member 1399
I_VRRT_AY enumeration member 1441	I_VVST_Dep enumeration member 1399
I_VRRT_AY_BOTTOM enumeration member 1441	I_VVST_DESCRIPTIONS enumeration member 1399
I_VRRT_AY_TOP enumeration member 1441	I_VVST_DetDi enumeration member 1399
I_VRRT_E_AX_BOTTOM enumeration member 1441	I_VVST_Dia3D enumeration member 1399
I_VRRT_E_AX_TOP enumeration member 1441	I_VVST_DIM_LINES enumeration member 1399
I_VRRT_E_AY_BOTTOM enumeration member 1441	I_VVST_ELE enumeration member 1399
I_VRRT_E_AY_TOP enumeration member 1441	I_VVST_ENVELOPES enumeration member 1399
I_VRRT_F enumeration member 1441	I_VVST_EXPLODE enumeration member 1399
I_VSCR_2048 enumeration member 1455	I_VVST_EXTREMES enumeration member 1399
I_VSCR_3072 enumeration member 1455	I_VVST_FE enumeration member 1399
I_VSCR_4096 enumeration member 1455	I_VVST_FE_CUTS enumeration member 1399
I_VSCR_DEFAULT enumeration member 1455	I_VVST_FE_DIRECTION enumeration member 1399
I_VST_DEFORM enumeration member 1405	I_VVST_FE_LAY enumeration member 1399
I_VST_FX enumeration member 1405	I_VVST_FILL enumeration member 1399
I_VST_FY enumeration member 1405	I_VVST_FORCES enumeration member 1399
I_VST_FZ enumeration member 1405	I_VVST_GloDi enumeration member 1399
I_VST_MX enumeration member 1405	I_VVST_GRID enumeration member 1399
I_VST_MY enumeration member 1405	I_VVST_LOADS enumeration member 1399
I_VST_MZ enumeration member 1405	I_VVST_LOCAL enumeration member 1399
I_VST_SIG enumeration member 1405	I_VVST_MAPS enumeration member 1399
I_VST_TAU enumeration member 1405	I_VVST_MAPS_DEFORATION enumeration member 1399
I_VT_DETAILED_ANALYSIS enumeration member 1388	I_VVST_MAPS_LAYER enumeration member 1399
I_VT_DIAGRAMS enumeration member 1388	I_VVST_MOBILE enumeration member 1399
I_VT_GLOBAL_ANALYSIS enumeration member 1388	I_VVST_MODE enumeration member 1399
I_VT_INFLUENCE_LINES enumeration member 1388	I_VVST_MTC enumeration member 1399
I_VT_MAPS_ON_BARS enumeration member 1388	I_VVST_NODE_LOADS enumeration member 1399
I_VT_MAPS_ONFINITE_ELEMENTS enumeration member 1388	I_VVST_NODES enumeration member 1399
I_VT_PANEL_CUTS enumeration member 1388	I_VVST_NOT_SELECTED enumeration member 1399
I_VT_STANDARD enumeration member 1388	I_VVST_OFFSETS enumeration member 1399
I_VVST_ACTIVE_CASE enumeration member 1399	I_VVST_PLOT enumeration member 1399
I_VVST_ACTIVE_MODE enumeration member 1399	I_VVSTREACTIONS enumeration member 1399
I_VVST_ACTIVE_PHASE enumeration member 1399	I_VVSTREINF_CROSSES enumeration member 1399
I_VVST_ACTIVE_QCMB enumeration member 1399	I_VVSTREINF_DIR enumeration member 1399
I_VVST_ANALYSIS enumeration member 1399	I_VVSTRELEASES enumeration member 1399
I_VVST_ANIMATION enumeration member 1399	I_VVSTRIGID_LINKS enumeration member 1399

I\_VVST\_RULER enumeration member 1399  
I\_VVST\_SCALE enumeration member 1399  
I\_VVST\_SECTION\_DRAW enumeration member 1399  
I\_VVST\_SECTION\_NAMES enumeration member 1399  
I\_VVST\_SECTION\_SURF enumeration member 1399  
I\_VVST\_SECTION\_SYMB enumeration member 1399  
I\_VVST\_SECTIONS enumeration member 1399  
I\_VVST\_STRESSES enumeration member 1399  
I\_VVST\_STRESSES\_GLOBAL enumeration member 1399  
I\_VVST\_STRUCTURE enumeration member 1399  
I\_VVST\_STRUCTURE\_3D enumeration member 1399  
I\_VVST\_SUPERPOS enumeration member 1399  
I\_VVST\_SUPPORTS enumeration member 1399  
I\_VVST\_THICKNESS enumeration member 1399  
I\_VVST\_TITLE enumeration member 1399  
I\_VVSV\_ANALYSIS\_DET enumeration member 1400  
I\_VVSV\_ANALYSIS\_GLO enumeration member 1400  
I\_VVSV\_ANALYSIS\_INF enumeration member 1400  
I\_VVSV\_BAR\_MAPS DESIGN MEMB LENGTH enumeration member 1400  
I\_VVSV\_BAR\_MAPS DESIGN RATIO enumeration member 1400  
I\_VVSV\_BAR\_MAPS DESIGN SLEND LAY enumeration member 1400  
I\_VVSV\_BAR\_MAPS DESIGN SLEND LAZ enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_FX enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_FY enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_FZ enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_MX enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_MY enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_MZ enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_SHEAR\_STRESS\_T enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_SHEAR\_STRESS TY enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_SHEAR\_STRESS\_TZ enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_FX\_AX enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MAX enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MAX\_MY enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MAX\_MZ enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MIN enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MIN\_MY enumeration member 1400  
I\_VVSV\_BAR\_MAPS\_STRESS\_S\_MIN\_MZ enumeration member 1400  
I\_VVSV\_DEFORMATION\_EXACT enumeration member 1400  
I\_VVSV\_DEFORMATION\_STD enumeration member 1400  
I\_VVSV\_DESCRIPTIONS\_HORIZONTAL enumeration member 1400  
I\_VVSV\_DESCRIPTIONS\_NONE enumeration member 1400  
I\_VVSV\_DESCRIPTIONS\_VERTICAL enumeration member 1400  
I\_VVSV\_ELE\_BAR\_NUM enumeration member 1400  
I\_VVSV\_ELE\_CAL\_NUM enumeration member 1400  
I\_VVSV\_ELE\_FE\_NUM enumeration member 1400  
I\_VVSV\_ELE\_FE\_VIS enumeration member 1400  
I\_VVSV\_ENVELOPES\_CMP enumeration member 1400  
I\_VVSV\_ENVELOPES\_CUT enumeration member 1400  
I\_VVSV\_ENVELOPES\_HST enumeration member 1400  
I\_VVSV\_ENVELOPES\_MAX enumeration member 1400  
I\_VVSV\_ENVELOPES\_MIN enumeration member 1400  
I\_VVSV\_ENVELOPES\_ONE enumeration member 1400  
I\_VVSV\_ENVELOPES\_SET enumeration member 1400  
I\_VVSV\_ENVELOPES\_SEU enumeration member 1400  
I\_VVSV\_FE\_ADV enumeration member 1400  
I\_VVSV\_FE\_COE enumeration member 1400  
I\_VVSV\_FE\_COI enumeration member 1400  
I\_VVSV\_FE\_COL enumeration member 1400  
I\_VVSV\_FE\_CON enumeration member 1400  
I\_VVSV\_FE\_DSC enumeration member 1400  
I\_VVSV\_FE\_DSO enumeration member 1400  
I\_VVSV\_FE\_EDG enumeration member 1400  
I\_VVSV\_FE\_EMI enumeration member 1400  
I\_VVSV\_FE\_MES enumeration member 1400  
I\_VVSV\_FE\_RES enumeration member 1400  
I\_VVSV\_FILL\_AVG enumeration member 1400  
I\_VVSV\_FILL\_BOR enumeration member 1400  
I\_VVSV\_FILL\_DEF enumeration member 1400  
I\_VVSV\_FILL\_FE enumeration member 1400  
I\_VVSV\_FILL\_FILL enumeration member 1400  
I\_VVSV\_FILL\_ISO enumeration member 1400  
I\_VVSV\_FILL\_SMOOTH\_CAR enumeration member 1400  
I\_VVSV\_FILL\_SMOOTH\_GLO enumeration member 1400

I_VVSV_FILL_SMOOTH_LOC enumeration member 1400	enumeration member 1400
I_VVSV_FILL_SMOOTH_NON enumeration member 1400	I_VVSV_MAPS_DETAILED_MEMB_FORCES_XX enumeration member 1400
I_VVSV_FILL_SMOOTH_SEL enumeration member 1400	I_VVSV_MAPS_DETAILED_MEMB_FORCES_XY enumeration member 1400
I_VVSV_FILL_TAN enumeration member 1400	I_VVSV_MAPS_DETAILED_MEMB_FORCES_YY enumeration member 1400
I_VVSV_FILL_VAL enumeration member 1400	I_VVSV_MAPS_DETAILED_MOMENTS_XX enumeration member 1400
I_VVSV_FORCES_BAR.REACT_KY enumeration member 1400	I_VVSV_MAPS_DETAILED_MOMENTS_XY enumeration member 1400
I_VVSV_FORCES_BAR.REACT_KZ enumeration member 1400	I_VVSV_MAPS_DETAILED_ROTATION_XX enumeration member 1400
I_VVSV_FORCES_DFL enumeration member 1400	I_VVSV_MAPS_DETAILED_ROTATION_YY enumeration member 1400
I_VVSV_FORCES_FX enumeration member 1400	I_VVSV_MAPS_DETAILED_ROTATION_Z enumeration member 1400
I_VVSV_FORCES_FXC enumeration member 1400	I_VVSV_MAPS_DETAILED_SHEAR_FORCES_XX enumeration member 1400
I_VVSV_FORCES_FY enumeration member 1400	I_VVSV_MAPS_DETAILED_SHEAR_FORCES_YY enumeration member 1400
I_VVSV_FORCES_FZ enumeration member 1400	I_VVSV_MAPS_DETAILED_SOIL_REACTIONS enumeration member 1400
I_VVSV_FORCES_MX enumeration member 1400	I_VVSV_MAPS_DETAILED_STRESSES_XX enumeration member 1400
I_VVSV_FORCES_MY enumeration member 1400	I_VVSV_MAPS_DETAILED_STRESSES_YY enumeration member 1400
I_VVSV_FORCES_MZ enumeration member 1400	I_VVSV_MAPS_DETAILED_STRESSES_Z enumeration member 1400
I_VVSV_FORCES_UX enumeration member 1400	I_VVSV_MAPS_ISOLINES enumeration member 1400
I_VVSV_FORCES_UY enumeration member 1400	I_VVSV_MAPS_LAYER_ABS_MINIMUM enumeration member 1400
I_VVSV_FORCES_UZ enumeration member 1400	I_VVSV_MAPS_LAYER_ARBITRARY enumeration member 1400
I_VVSV_LOADS_SYMBOL enumeration member 1400	I_VVSV_MAPS_LAYER_LOWER enumeration member 1400
I_VVSV_LOADS_VALUE enumeration member 1400	I_VVSV_MAPS_LAYER_MAXIMUM enumeration member 1400
I_VVSV_MAPS_COMPLEX_MEMB_FORCES enumeration member 1400	I_VVSV_MAPS_LAYER_MIDDLE enumeration member 1400
I_VVSV_MAPS_COMPLEX_MOMENTS enumeration member 1400	I_VVSV_MAPS_LAYER_MINIMUM enumeration member 1400
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_BOTTOM_ MXX enumeration member 1400	I_VVSV_MAPS_LAYER_UPPER enumeration member 1400
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_BOTTOM_ MYY enumeration member 1400	I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_X enumeration member 1400
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_TOP_MXX enumeration member 1400	
I_VVSV_MAPS_COMPLEX_REINFORCEMENT_TOP_MYY enumeration member 1400	
I_VVSV_MAPS_COMPLEX_STRESSES enumeration member 1400	
I_VVSV_MAPS_CROSS_M enumeration member 1400	
I_VVSV_MAPS_CROSS_N enumeration member 1400	
I_VVSV_MAPS_CROSS_S enumeration member 1400	
I_VVSV_MAPS_DESCRIPTION enumeration member 1400	
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_XX enumeration member 1400	
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_YY enumeration member 1400	
I_VVSV_MAPS_DETAILED_DISPLACEMENTS_Z	

I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_Y enumeration member 1400	I_VVSVREACTIONS_DIV enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_GLOBAL_DISP_Z enumeration member 1400	I_VVSVREACTIONS_F enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_1 enumeration member 1400	I_VVSVREACTIONS_FX enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_1_2 enumeration member 1400	I_VVSVREACTIONS_FY enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_2 enumeration member 1400	I_VVSVREACTIONS_FZ enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MEMB_FORCES_ANGLE enumeration member 1400	I_VVSVREACTIONS_M enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MOMENTS_1 enumeration member 1400	I_VVSVREACTIONS_MX enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MOMENTS_1_2 enumeration member 1400	I_VVSVREACTIONS_MY enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MOMENTS_2 enumeration member 1400	I_VVSVREACTIONS_MZ enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_MOMENTS_ANGLE enumeration member 1400	I_VVSVREACTIONS_PSEUDO_F enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_SHEAR_FORCESENumeration member 1400	I_VVSVREACTIONS_PSEUDO_M enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_SHEAR_STRESSES enumeration member 1400	I_VVSVREACTIONS_RESID_F enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_STRESSES_1 enumeration member 1400	I_VVSVREACTIONS_RESID_M enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_STRESSES_1_2 enumeration member 1400	I_VVSVREACTIONS_ROT enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_STRESSES_2 enumeration member 1400	I_VVSVREACTIONS_VAL enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_STRESSES_ANGLE enumeration member 1400	I_VVSVREINFORCEMENT_A1 enumeration member 1400
I_VVSV_MAPS_PRINCIPAL_TOTAL_DISP enumeration member 1400	I_VVSVREINFORCEMENT_A1BARS enumeration member 1400
I_VVSV_MAPS_SMOOTH_CAR enumeration member 1400	I_VVSVREINFORCEMENT_A1R enumeration member 1400
I_VVSV_MAPS_SMOOTH_GLO enumeration member 1400	I_VVSVREINFORCEMENT_A2 enumeration member 1400
I_VVSV_MAPS_SMOOTH_LOC enumeration member 1400	I_VVSVREINFORCEMENT_A2BARS enumeration member 1400
I_VVSV_MAPS_SMOOTH_NON enumeration member 1400	I_VVSVREINFORCEMENT_A2R enumeration member 1400
I_VVSV_MAPS_SMOOTH_SEL enumeration member 1400	I_VVSVREINFORCEMENT_DISTRIB enumeration member 1400
I_VVSV_MOBILE_CAR enumeration member 1400	I_VVSVREINFORCEMENT_DISTRIBR enumeration member 1400
I_VVSV_MOBILE_ELE enumeration member 1400	I_VVSVREINFORCEMENT_PRC enumeration member 1400
I_VVSV_MOBILE_LOADS enumeration member 1400	I_VVSVREINFORCEMENT_PRCR enumeration member 1400
I_VVSV_MOBILE_ROUTE enumeration member 1400	I_VVSVRIGIDLINKS_DET enumeration member 1400
I_VVSV_MTC_CABLE enumeration member 1400	I_VVSVRIGIDLINKS_RIG enumeration member 1400
I_VVSV_MTC_COMPRESSION enumeration member 1400	I_VVSVRIGIDLINKS_SUR enumeration member 1400
I_VVSV_MTC_FEM enumeration member 1400	I_VVSVRIGIDLINKS_SYM enumeration member 1400
I_VVSV_MTC_MATERIAL enumeration member 1400	I_VVSVSECTION_DRAW_COLOR enumeration member 1400
I_VVSV_MTC_TENSION enumeration member 1400	I_VVSVSECTION_DRAW_GROUP enumeration member 1400
	I_VVSVSECTION_DRAW_SHAPE enumeration member 1400
	I_VVSVSECTION_DRAW_TOP_BOTTOM enumeration member 1400
	I_VVSVSECTION_SYMB_ELA enumeration member 1400
	I_VVSVSECTION_SYMB_SYMB enumeration member 1400

I_VVSV_STRESSES_AXIAL enumeration member 1400	I_WS_NORMAL enumeration member 1644
I_VVSV_STRESSES_FLEXMAX_MY enumeration member 1400	IBST_BEAM_SUPPORT_FIXED enumeration member 1162
I_VVSV_STRESSES_FLEXMAX_MZ enumeration member 1400	IBST_BEAM_SUPPORT_PINNED enumeration member 1162
I_VVSV_STRESSES_FLEXMIN_MY enumeration member 1400	IBST_BEAM_SUPPORT_SWAY enumeration member 1162
I_VVSV_STRESSES_FLEXMIN_MZ enumeration member 1400	Influence lines 1378
I_VVSV_STRESSES_GLOBAL_MISES_MAX enumeration member 1400	IRBestBendType 1689
I_VVSV_STRESSES_GLOBAL_MISES_MIN enumeration member 1400	IRBestCalcErrors 1677
I_VVSV_STRESSES_GLOBAL_NORMAL_MAX enumeration member 1400	IRBestCalcParamsData 1668
I_VVSV_STRESSES_GLOBAL_NORMAL_MIN enumeration member 1400	about IRBestCalcParamsData 1668
I_VVSV_STRESSES_GLOBAL_TAU_MAX enumeration member 1400	IRBestCalcParamsData fields 1669
I_VVSV_STRESSES_GLOBAL_TAU_MIN enumeration member 1400	IRBestCalcParamsData members 1668
I_VVSV_STRESSES_GLOBAL_USER_MAX enumeration member 1400	ModularityList 1669
I_VVSV_STRESSES_GLOBAL_USER_MIN enumeration member 1400	IRBestCalcParamsDataDoubleValue 1669
I_VVSV_STRESSES_NORMALMAX enumeration member 1400	IRBestCalcParamsDataIntegerValue 1670
I_VVSV_STRESSES_NORMALMIN enumeration member 1400	IRBestCalcParamsDataList 1671
I_VVSV_STRESSES_SHEAR_Y enumeration member 1400	about IRBestCalcParamsDataList 1671
I_VVSV_STRESSES_SHEAR_Z enumeration member 1400	Create 1673
I_VVSV_STRESSES_TORSION enumeration member 1400	Delete 1673
I_VVSV_STRUCTURE_COR enumeration member 1400	Get 1673
I_VVSV_STRUCTURE_HDF enumeration member 1400	IRBestCalcParamsDataList fields 1672
I_VVSV_STRUCTURE_HDL enumeration member 1400	IRBestCalcParamsDataList members 1671
I_VVSV_STRUCTURE_HDR enumeration member 1400	IRBestCalcParamsDataList methods 1673
I_VVSV_STRUCTURE_HDY enumeration member 1400	LabNames 1672
I_VVSV_STRUCTURE_HID enumeration member 1400	Selected 1672
I_VVSV_STRUCTURE_ON enumeration member 1400	Store 1674
I_VVSV_STRUCTURE_POI enumeration member 1400	IRBestCalcParamsDataStringValue 1671
I_VVSV_STRUCTURE_SHD enumeration member 1400	IRBestCalcParamsDlg 1660
I_VVSV_STRUCTURE_SPD enumeration member 1400	about IRBestCalcParamsDlg 1660
I_VVSV_SUPPORTS_COD enumeration member 1400	DoModal 1660
I_VVSV_SUPPORTS_LBL enumeration member 1400	IRBestCalcParamsDlg members 1660
I_VVSV_SUPPORTS_SYMB enumeration member 1400	IRBestCalcParamsDlg methods 1660
I_WS_MAXIMIZE enumeration member 1644	SetStandard 1661
I_WS_MINIMIZE enumeration member 1644	IRBestCalculationType 1688
	IRBestCalcWarnings 1678
	IRBestCodeCalcEngine 1682
	about IRBestCodeCalcEngine 1682
	Calculate 1683
	GetResults 1683
	IRBestCodeCalcEngine members 1682
	IRBestCodeCalcEngine methods 1683
	SetForces 1684
	SetForcesArray 1684

SetForcesSlender	1684	IRBestMemberDataIntegerValue	1675
SetGeometry	1684	IRBestMemberDataStringValue	1685
SetParams	1685	IRBestMemberDlg	1661
IRBestCodeCalculationType	1687	about IRBestMemberDlg	1661
IRBestCodeService	1658	DoModal	1662
about IRBestCodeService	1658	IRBestMemberDlg members	1661
CalcEngine	1659	IRBestMemberDlg methods	1661
CalcParamsDlg	1659	SetStandard	1662
IRBestCodeService fields	1659	IRBestMemberType	1685
IRBestCodeService members	1658	IRBestParamSet	1662
MemberDlg	1659	about IRBestParamSet	1662
IRBestCodeService2	1691	Clear	1664
about IRBestCodeService2	1691	ClearDouble	1664
IRBestCodeService2 fields	1692	ClearInteger	1665
IRBestCodeService2 members	1691	ClearString	1665
RobotVersion	1692	GetDouble	1665
ShowBucklingButton	1692	GetInteger	1665
IRBestCodeServiceExt	1687	GetString	1666
about IRBestCodeServiceExt	1687	IRBestParamSet members	1663
GetCalcParamsDlg	1688	IRBestParamSet methods	1663
IRBestCodeServiceExt members	1687	IsValidDouble	1666
IRBestCodeServiceExt methods	1688	IsValidInteger	1666
IsServed	1688	IsValidString	1666
IRBestCoordSystem	1690	SetDouble	1667
IRBestDimParams	1681	SetInteger	1667
about IRBestDimParams	1681	SetString	1667
IRBestDimParamsDoubleValue	1689	IRBestPlateCalcParamsDlg	1686
IRBestDimParamsIntegerValue	1681	about IRBestPlateCalcParamsDlg	1686
IRBestDirection	1689	DoModal	1686
IRBestForceData	1679	IRBestPlateCalcParamsDlg members	1686
about IRBestForceData	1679	IRBestPlateCalcParamsDlg methods	1686
IRBestForceData fields	1680	SetStandard	1687
IRBestForceData members	1679	IRBestResults	1675
LimitState	1680	about IRBestResults	1675
Values	1681	IRBestResultsDoubleValue	1676
IRBestForceDataDoubleValue	1690	IRBestResultsIntegerValue	1677
IRBestForceDataIntegerValue	1690	IRBestResultsStringValue	1678
IRBestForceDataSLSCombType	1691	IRConcr_ACI318_BarDownDim	1742
IRBestLevel	1689	IRConcr_ACI318_ConcreteParams	1741
IRBestMemberData	1674	about IRConcr_ACI318_ConcreteParams	1741
about IRBestMemberData	1674	AsInStructure	1742
IRBestMemberDataDoubleValue	1674	IRConcr_ACI318_ConcreteParams fields	1741

IRConcr_ACI318_ConcreteParams members	1741	CrackingTop	1727
IRConcr_ACI318_LoadActionPeriodType	1740	CrackWaterLevel	1727
IRConcr_ACI318_MetricBarDim	1743	CrackWaterLevelBot	1727
IRConcr_ACI318_ReinforceData	1733	CrackWaterLevelTop	1728
about IRConcr_ACI318_ReinforceData	1733	IRConcr_BAEL_ReinforceData	fields 1723
GetBarDim	1738	IRConcr_BAEL_ReinforceData	members 1722
GetMetricBarDim	1739	Main	1728
IRConcr_ACI318_ReinforceData	fields 1734	SLS_ConcreteStressesBot	1728
IRConcr_ACI318_ReinforceData	members 1733	SLS_ConcreteStressesTop	1728
IRConcr_ACI318_ReinforceData	methods 1738	SLS_Deflection	1729
IsMetric	1734	SLS_DeflectionReinfCorrect	1729
Main	1735	SLS_MaxDeflection	1729
SetBarDim	1739	SLS_SteelStressesBot	1730
SetMetricBarDim	1739	SLS_SteelStressesTop	1730
SLS_Cracking	1735	SteelSymbol	1730
SLS_CreepingCoef	1735	IRConcr_BAEL_SteelGrades	1732
SLS_CreepingCoefValue	1735	IRConcr_BAEL_WaterLevel	1731
SLS_Deflection	1736	IRConcr_BS8110_ConcreteAge	1750
SLS_DeflectionReinfCorrection	1736	IRConcr_BS8110_ConcreteGrades	1751
SLS_Factor	1736	IRConcr_BS8110_ExposureRatings	1750
SLS_FactorValue	1737	IRConcr_BS8110_PartialSafetyFactors	1751
SLS_LoadActionPeriod	1737	IRConcr_BS8110_ReinforceData	1744
SLS_LoadRatio	1737	about IRConcr_BS8110_ReinforceData	1744
SLS_MaxCracking	1738	IRConcr_BS8110_ReinforceData	fields 1745
SLS_MaxDeflection	1738	IRConcr_BS8110_ReinforceData	members 1744
IRConcr_ACI318_SteelGrades	1740	Main	1745
IRConcr_BAEL_ConcreteGrades	1731	PartialSafetyFactors	1745
IRConcr_BAEL_CrackingType	1730	SLS_ConcreteAge	1746
IRConcr_BAEL_EnvironmentType	1731	SLS_Cracking	1746
IRConcr_BAEL_ReinforceData	1721	SLS_CrackingReinfCorection	1746
about IRConcr_BAEL_ReinforceData	1721	SLS_CreepingCoef	1747
CrackAlpha	1724	SLS_CreepingCoefValue	1747
CrackAlphaBot	1724	SLS_Deflection	1747
CrackAlphaTop	1725	SLS_DeflectionReinfCorection	1748
CrackEnv	1725	SLS_EnvHumidity	1748
CrackEnvBot	1725	SLS_Exposure	1748
CrackEnvTop	1725	SLS_LoadRatio	1749
CrackExtraParams	1726	SLS_MaxCracking	1749
CrackExtraParamsBot	1726	SLS_MaxDeflection	1749
CrackExtraParamsTop	1726	IRConcr_EC2_ConcreteGrades	1766
Cracking	1726	IRConcr_EC2_ExposureRatings	1766
CrackingBot	1727	IRConcr_EC2_ITALIAN_SteelGrades	1766

IRConcr_EC2_NAD 1771	IRConcr_PN84_ReinforceData 1705
IRConcr_EC2_ReinforceData 1766	about IRConcr_PN84_ReinforceData 1705
about IRConcr_EC2_ReinforceData 1766	CheckConstructionStages 1707
IRConcr_EC2_ReinforceData fields 1768	IRConcr_PN84_ReinforceData fields 1707
IRConcr_EC2_ReinforceData members 1767	IRConcr_PN84_ReinforceData members 1706
NAD 1768	LongActionAbove100 1708
SLS_ConcreteAge 1769	Main 1708
SLS_Cracking 1769	SingleShortLoad 1708
SLS_CrackingReinfCorrect 1769	SLS_ConcreteAge 1708
SLS_CreepingCoef 1769	SLS_Cracking 1709
SLS_CreepingCoefValue 1769	SLS_CrackingReinfCorrect 1709
SLS_Deflection 1770	SLS_CreepingCoef 1709
SLS_DeflectionReinfCorrect 1770	SLS_Deflection 1709
SLS_EnvHumidityVal 1770	SLS_DeflectionReinfCorrect 1710
SLS_Exposure 1770	SLS_EnvHumidity 1710
SLS_MaxCracking 1770	SLS_Exposure 1710
SLS_MaxDeflection 1771	SLS_LoadsRatio 1711
IRConcr_IS2000_EnvironmentType 1771	SLS_MaxCracking 1711
IRConcr_IS2000_ReinforceData 1772	SLS_MaxDeflection 1711
about IRConcr_IS2000_ReinforceData 1772	IRConcr_PN99_ConcreteGrades 1721
IRConcr_IS2000_ReinforceData fields 1773	IRConcr_PN99_ExposureRatings 1713
IRConcr_IS2000_ReinforceData members 1772	IRConcr_PN99_ReinforceData 1713
Main 1774	about IRConcr_PN99_ReinforceData 1713
SLS_ConcreteAge 1774	IRConcr_PN99_ReinforceData fields 1715
SLS_Cracking 1775	IRConcr_PN99_ReinforceData members 1714
SLS_CrackingReinfCorrect 1775	IsSpecialStructure 1716
SLS_CreepingCoeff 1775	Main 1716
SLS_CreepingCoeffValue 1775	SLS_Concrete 1716
SLS_Deflection 1776	SLS_ConcreteAge 1716
SLS_DeflectionReinfCorrect 1776	SLS_Cracking 1717
SLS_ExposureBot 1776	SLS_CrackingReinfCorrect 1717
SLS_ExposureTop 1776	SLS_CreepingCoef 1717
SLS_LoadRatio 1777	SLS_CreepingCoefValue 1717
SLS_MaxCrackingBot 1777	SLS_Deflection 1718
SLS_MaxCrackingBotEnabled 1777	SLS_DeflectionReinfCorrect 1718
SLS_MaxCrackingTop 1777	SLS_EnvHumidityVal 1718
SLS_MaxCrackingTopEnabled 1778	SLS_ExposureBot 1718
SLS_MaxDeflection 1778	SLS_ExposureTop 1719
IRConcr_PN_SteelGrades 1711	SLS_LoadRatio 1719
IRConcr_PN84_ConcreteGrades 1712	SLS_MaxCrackingBot 1719
IRConcr_PN84_ExposureRatings 1712	SLS_MaxCrackingBotEnabled 1720
IRConcr_PN84_HumidityType 1732	SLS_MaxCrackingTop 1720

SLS_MaxCrackingTopEnabled 1720	IRConcrBarSectionData fields 1787
SLS_MaxDeflection 1720	IRConcrBarSectionData members 1786
IRConcr_SNIP_ConcreteGrades 1763	N 1789
IRConcr_SNIP_ConcreteParams 1763	Per 1789
about IRConcr_SNIP_ConcreteParams 1763	Type 1789
ConcretingInLayers 1765	IRConcrBarSectionGeometryType 1790
CuringMethod 1765	IRConcrBarSubtype 1250
HighHumidity 1765	IRConcrBarType 1250
IRConcr_SNIP_ConcreteParams fields 1764	IRConcrBeam 1124
IRConcr_SNIP_ConcreteParams members 1764	about IRConcrBeam 1124
Type 1765	Activate 1131
IRConcr_SNIP_ConcreteTypes 1763	Calculate 1132
IRConcr_SNIP_CuringMethods 1763	CalculationOptions 1126
IRConcr_SNIP_Exposure 1765	Concrete 1126
IRConcr_SNIP_ReinforceData 1761	CreateCalculationNoteRtf 1132
about IRConcr_SNIP_ReinforceData 1761	CreateFromBars 1132
IRConcr_SNIP_ReinforceData fields 1762	GenerateSpliceBars 1132
IRConcr_SNIP_ReinforceData members 1761	Geometry 1127
SLS_Cracking 1762	ImportOptions 1127
SLS_CrackingReinfCorrection 1762	IRConcrBeam fields 1125
SLS_Exposure 1762	IRConcrBeam members 1124
IRConcr_SNIP_SteelGrades 1763	IRConcrBeam methods 1131
IRConcrBarDiameters 1257	IsActive 1127
about IRConcrBarDiameters 1257	IsSelected 1127
Add 1258	LinearLoad 1128
Count 1258	LinearLoadsCount 1128
IRConcrBarDiameters fields 1258	Name 1128
IRConcrBarDiameters members 1257	NumberOfElements 1128
IRConcrBarDiameters methods 1258	PatternOptions 1129
Item 1258	PointLoad 1129
RemoveAll 1259	PointLoadsCount 1129
IRConcrBarSectionData 1786	Reinforcement 1129
A 1787	Save 1133
about IRConcrBarSectionData 1786	Steel 1130
Ac 1787	StoryOptions 1130
Ap 1787	StructureUserNo 1130
B 1788	StructureUserNoCount 1130
Bf 1788	Uniqueld 1131
Bfb 1788	Verify 1133
H 1788	VerifySpan 1133
Hf 1788	IRConcrBeamCalcOptions 1133
Hfb 1789	about IRConcrBeamCalcOptions 1133

CalcSpanLengthInAxis	1134	Spans	1144
CoverBottom	1135	Type	1145
CoverBottomFixed	1135	Value1	1145
CoverSide	1135	Value2	1145
CoverSideFixed	1135	Value3	1145
CoverTop	1136	X1	1146
CoverTopFixed	1136	X2	1146
IRConcrBeamCalcOptions	fields 1134	X3	1146
IRConcrBeamCalcOptions	members 1134	X4	1146
IRConcrBeamCalcPointDefinitionType	1785	IRConcrBeamLinearLoadType	1159
IRConcrBeamCrackingType	1162	IRConcrBeamLoadNatureType	1159
IRConcrBeamGeometry	1136	IRConcrBeamPatternOptions	1147
about IRConcrBeamGeometry	1136	about IRConcrBeamPatternOptions	1147
AutoNameSpans	1137	IRConcrBeamPatternOptions	fields 1147
AutoNameSupports	1138	IRConcrBeamPatternOptions	members 1147
ConcreteVolume	1138	SpanBySpan	1147
IRConcrBeamGeometry	fields 1137	IRConcrBeamPointLoad	1148
IRConcrBeamGeometry	members 1137	about IRConcrBeamPointLoad	1148
IRConcrBeamGeometry	methods 1139	Case	1148
LeftCantilever	1138	IRConcrBeamPointLoad	fields 1148
RightCantilever	1138	IRConcrBeamPointLoad	members 1148
ShutteringArea	1139	Nature	1149
Span	1139	RelativeCoordinates	1149
SpanNumber	1139	Spans	1149
UpdateInternalGeometryData	1140	Type	1149
IRConcrBeamImportOptions	1140	Value	1150
about IRConcrBeamImportOptions	1140	X1	1150
GroupByGeometry	1141	IRConcrBeamPointLoadType	1160
GroupByLevel	1141	IRConcrBeamSectionDimType	1161
ImportCombinations	1141	IRConcrBeamSectionType	1160
IRConcrBeamImportOptions	fields 1141	IRConcrBeamSegment	1150
IRConcrBeamImportOptions	members 1140	about IRConcrBeamSegment	1150
LiveLongCoeff	1142	Dim	1151
RunCalcAuto	1142	IRConcrBeamSegment	fields 1151
ShowDialog	1142	IRConcrBeamSegment	members 1151
IRConcrBeamLinearLoad	1142	SectionType	1151
about IRConcrBeamLinearLoad	1142	IRConcrBeamSpan	1151
Case	1143	about IRConcrBeamSpan	1151
IRConcrBeamLinearLoad	fields 1143	CalculationLength	1152
IRConcrBeamLinearLoad	members 1143	HasOpenings	1153
Nature	1144	IRConcrBeamSpan	fields 1152
RelativeCoordinates	1144	IRConcrBeamSpan	members 1152

LeftSupport 1153	IRConcrCodeBeamCommand 1798
Length 1153	IRConcrCodeColumn 1793
Name 1153	about IRConcrCodeColumn 1793
RightSupport 1154	IRConcrCodeColumn members 1794
Segment 1154	IRConcrCodeColumn methods 1794
SegmentNumber 1154	IsCommandEnabled 1794
IRConcrBeamSpanNumbers 1154	Verification 1794
about IRConcrBeamSpanNumbers 1154	IRConcrCodeColumnCommand 1795
Add 1156	IRConcrCodeReport 1795
Count 1155	about IRConcrCodeReport 1795
IRConcrBeamSpanNumbers fields 1155	AddError 1796
IRConcrBeamSpanNumbers members 1155	AddMessage 1796
IRConcrBeamSpanNumbers methods 1156	AddWarning 1796
Item 1155	IRConcrCodeReport members 1795
RemoveAll 1156	IRConcrCodeReport methods 1795
IRConcrBeamStoryOptions 1156	IRConcrCodeService 1792
about IRConcrBeamStoryOptions 1156	about IRConcrCodeService 1792
Cracking 1157	Beam 1792
IRConcrBeamStoryOptions fields 1157	Column 1793
IRConcrBeamStoryOptions members 1157	IRConcrCodeService fields 1792
IRConcrBeamSupport 1157	IRConcrCodeService members 1792
about IRConcrBeamSupport 1157	IRConcrCodeService methods 1793
IRConcrBeamSupport fields 1158	IsConcrComponentServed 1793
IRConcrBeamSupport members 1157	IRConcrColumn 1179
MaterialType 1158	about IRConcrColumn 1179
Name 1158	Activate 1187
Type 1158	BucklingModel 1181
Width 1159	Calculate 1187
IRConcrBeamSupportMaterialType 1162	CalculationOptions 1182
IRConcrBeamSupportType 1162	Concrete 1182
IRConcrCalcEngine 1790	CreateCalculationNoteRtf 1187
about IRConcrCalcEngine 1790	CreateFromBars 1187
IRConcrCalcEngine fields 1790	Geometry 1182
IRConcrCalcEngine members 1790	HasUpperColumn 1182
MemberRequiredReinf 1791	ImportOptions 1183
SlabRequiredReinf 1791	IRConcrColumn fields 1181
IRConcrCodeBeam 1796	IRConcrColumn members 1180
about IRConcrCodeBeam 1796	IRConcrColumn methods 1186
IRConcrCodeBeam members 1797	IsActive 1183
IRConcrCodeBeam methods 1797	IsSelected 1183
IsCommandEnabled 1797	Loads 1183
Verification 1797	Name 1184

NumberOfElements 1184	ShutteringArea 1191
PatternOptions 1184	IRConcrColumnImportOptions 1191
Reinforcement 1184	about IRConcrColumnImportOptions 1191
Save 1188	GroupByGeometry 1192
Steel 1185	GroupByLevel 1193
StructureUserNo 1185	ImportCombinations 1193
StructureUserNoCount 1185	IRConcrColumnImportOptions fields 1192
UniqueId 1185	IRConcrColumnImportOptions members 1192
UpperColumn 1186	RunCalcAuto 1193
UpperColumnDY 1186	ShowDialog 1193
UpperColumnDZ 1186	IRConcrColumnLoad 1202
Verify 1188	about IRConcrColumnLoad 1202
IRConcrColumnBucklingModel 1196	CaseName 1203
about IRConcrColumnBucklingModel 1196	CaseType 1203
IRConcrColumnBucklingModel fields 1196	Fy 1203
IRConcrColumnBucklingModel members 1196	Fz 1204
IsDirectionYOff 1197	Gamma 1204
IsDirectionZOff 1197	IRConcrColumnLoad fields 1202
IsSwayY 1197	IRConcrColumnLoad members 1202
IsSwayZ 1198	MnsY 1204
IsTotalStructureHeight 1198	MnsZ 1204
ky 1198	MyA 1205
kz 1198	MyB 1205
Ly 1199	MyC 1205
Lz 1199	Mza 1205
NumberOfStories 1199	Mzb 1206
TotalStructureHeight 1199	Mzc 1206
IRConcrColumnCalcOptions 1188	N 1206
about IRConcrColumnCalcOptions 1188	NdN 1206
Cover 1189	IRConcrColumnLoadCaseType 1207
IRConcrColumnCalcOptions fields 1189	IRConcrColumnLoads 1200
IRConcrColumnCalcOptions members 1188	about IRConcrColumnLoads 1200
IRConcrColumnDimensionType 1195	Add 1201
IRConcrColumnGeometry 1189	Count 1200
about IRConcrColumnGeometry 1189	IRConcrColumnLoads fields 1200
ConcreteVolume 1190	IRConcrColumnLoads members 1200
Dim 1190	IRConcrColumnLoads methods 1201
DimIsFixed 1190	Item 1201
IRConcrColumnGeometry fields 1190	RemoveAll 1201
IRConcrColumnGeometry members 1189	IRConcrColumnPatternOptions 1194
Section 1191	about IRConcrColumnPatternOptions 1194
SectionName 1191	IRConcrColumnPatternOptions fields 1194

IRConcrColumnPatternOptions members 1194	IRConcrContinuousFootingGeometry 1170
TiesToBeam 1194	about IRConcrContinuousFootingGeometry 1170
IRConcrColumnType 1195	AutoNameSpans 1171
IRConcrConcrete 1255	AutoNameSupports 1171
about IRConcrConcrete 1255	ConcreteVolume 1172
CharacteristicStrength 1256	IRConcrContinuousFootingGeometry fields 1171
IRConcrConcrete fields 1256	IRConcrContinuousFootingGeometry members 1170
IRConcrConcrete members 1256	IRConcrContinuousFootingGeometry methods 1173
IRConcrConcreteParams 1700	LeftCantilever 1172
about IRConcrConcreteParams 1700	RightCantilever 1172
Ec 1701	ShutteringArea 1172
Fc 1701	Span 1173
Fc_calc 1702	SpanNumber 1173
Fcj 1702	UpdateInternalGeometryData 1173
Ft 1702	IRConcrContinuousFootingSectionDimType 1178
Grade 1702	IRConcrContinuousFootingSectionType 1177
IRConcrConcreteParams fields 1700	IRConcrContinuousFootingSegment 1173
IRConcrConcreteParams members 1700	about IRConcrContinuousFootingSegment 1173
IRConcrContinuousFooting 1163	Dim 1174
about IRConcrContinuousFooting 1163	IRConcrContinuousFootingSegment fields 1174
Activate 1169	IRConcrContinuousFootingSegment members 1174
Calculate 1169	SectionType 1174
CalculationOptions 1165	IRConcrContinuousFootingSpan 1175
Concrete 1165	about IRConcrContinuousFootingSpan 1175
CreateCalculationNoteRtf 1169	CalculationLength 1176
Geometry 1165	IRConcrContinuousFootingSpan fields 1175
ImportOptions 1166	IRConcrContinuousFootingSpan members 1175
IRConcrContinuousFooting fields 1164	LeftSupport 1176
IRConcrContinuousFooting members 1164	Length 1176
IRConcrContinuousFooting methods 1168	Name 1176
IsActive 1166	RightSupport 1177
IsSelected 1166	Segment 1177
Name 1166	SegmentNumber 1177
NumberOfElements 1167	IRConcrContinuousFootingSupportMaterialType 1178
PatternOptions 1167	IRConcrDeepBeam 1221
Reinforcement 1167	about IRConcrDeepBeam 1221
Save 1169	Activate 1224
Steel 1167	Concrete 1222
StructureUserNo 1168	CreateCalculationNoteRtf 1224
StructureUserNoCount 1168	CreateFromObjects 1225
UniqueId 1168	Geometry 1222
Verify 1170	IRConcrDeepBeam fields 1222

IRConcrDeepBeam members 1221	Name 1105
IRConcrDeepBeam methods 1224	NumberOfElements 1106
IsActive 1222	Pattern 1106
IsSelected 1223	Reinforcement 1106
Name 1223	Results 1106
Reinforcement 1223	Save 1110
Save 1225	Steel 1107
Steel 1223	StructureUserNo 1107
UniqueId 1224	StructureUserNoCount 1107
IRConcrDeepBeamGeometry 1225	UniqueId 1107
about IRConcrDeepBeamGeometry 1225	Verify 1110
IRConcrDrawing 1259	IRConcrFootingCalculationOptions 1118
about IRConcrDrawing 1259	about IRConcrFootingCalculationOptions 1118
Group 1260	Cover 1119
GroupCount 1260	IRConcrFootingCalculationOptions fields 1119
IRConcrDrawing fields 1260	IRConcrFootingCalculationOptions members 1118
IRConcrDrawing members 1259	NominalCover 1119
IRConcrDrawingBarGroup 1260	NominalPierCover 1119
about IRConcrDrawingBarGroup 1260	PierCover 1120
Bar 1261	IRConcrFootingDimType 1113
BarCount 1261	IRConcrFootingGeometry 1110
IRConcrDrawingBarGroup fields 1261	about IRConcrFootingGeometry 1110
IRConcrDrawingBarGroup members 1261	ConcreteVolume 1111
IRConcrFooting 1101	GetDim 1113
about IRConcrFooting 1101	IRConcrFootingGeometry fields 1111
Activate 1108	IRConcrFootingGeometry members 1110
Calculate 1108	IRConcrFootingGeometry methods 1112
CalculationOptions 1103	PierType 1111
Concrete 1103	SetDim 1113
CreateCalculationNoteRtf 1109	Shape 1112
CreateFromNodes 1109	ShutteringArea 1112
CreateReinforcement 1109	Type 1112
DisplayDrawing 1109	IRConcrFootingGround 1115
FootRotation 1104	about IRConcrFootingGround 1115
Geometry 1104	ColumnPierLevel 1116
Ground 1104	IRConcrFootingGround fields 1115
IRConcrFooting fields 1103	IRConcrFootingGround members 1115
IRConcrFooting members 1102	SoilLevel 1116
IRConcrFooting methods 1108	IRConcrFootingLoads 1116
IsActive 1105	about IRConcrFootingLoads 1116
IsSelected 1105	IRConcrFootingPattern 1120
Loads 1105	about IRConcrFootingPattern 1120

Diameter1	1121	about IRConcrPlateCodeService2	1778
Diameter2	1121	IRConcrPlateCodeService2 fields	1779
IRConcrFootingPattern	fields 1121	IRConcrPlateCodeService2 members	1779
IRConcrFootingPattern	members 1120	RobotVersion	1779
Spacing1Max	1121	ShowLongTermCracking	1779
Spacing1Min	1122	IRConcrReinforceBarType	1743
Spacing2Max	1122	IRConcrReinforceCalcType	1757
Spacing2Min	1122	IRConcrReinforceData	1694
IRConcrFootingPierType	1123	about IRConcrReinforceData	1694
IRConcrFootingResults	1117	BarDim_D1_Bot	1696
about IRConcrFootingResults	1117	BarDim_D1_Up	1696
Ax	1117	BarDim_D2_Bot	1696
Ay	1118	BarDim_D2_Up	1697
IRConcrFootingResults	fields 1117	CodeName	1697
IRConcrFootingResults	members 1117	Concrete	1697
Val	1118	Cover_Bot	1697
IRConcrFootingShapeType	1114	Cover_Up	1698
IRConcrFootingType	1114	GetMainDirection	1698
IRConcrFootResultType	1122	IRConcrReinforceData	fields 1695
IRConcrMemberRequiredReinfCalcParams	1782	IRConcrReinforceData	members 1695
about IRConcrMemberRequiredReinfCalcParams	1782	IRConcrReinforceData	methods 1698
BeamPointsType	1783	ReinforcingSteel	1698
BeamPointsValue	1783	SetMainDirection	1699
CasesALS	1784	IRConcrReinforceData2	1757
CasesSLS	1784	about IRConcrReinforceData2	1757
CasesULS	1784	IRConcrReinforceData2	fields 1758
CombALS	1784	IRConcrReinforceData2	members 1757
CombSLS	1785	Main	1758
CombULS	1785	IRConcrReinforceDataMain	1758
IRConcrMemberRequiredReinfCalcParams	fields 1782	about IRConcrReinforceDataMain	1758
IRConcrMemberRequiredReinfCalcParams	members 1782	IRConcrReinforceDataMain	fields 1759
Members	1785	IRConcrReinforceDataMain	members 1759
MembraneReinfInOneLayer	1760	MinimumReinf	1760
UnidirReinf	1760	ReinfCalcType	1760
IRConcrMemberRequiredReinfEngine	1780	UnidirReinf	1760
about IRConcrMemberRequiredReinfEngine	1780	IRConcrReinforceDirection	1699
Calculate	1781	IRConcrReinforcement	1238
GetCalculatedMembers	1781	about IRConcrReinforcement	1238
IRConcrMemberRequiredReinfEngine	fields 1780	Bars	1239
IRConcrMemberRequiredReinfEngine	members 1780	Comment	1239
IRConcrMemberRequiredReinfEngine	methods 1781	Freeze	1241
Params	1781		
IRConcrMinimumReinforcementType	1771		
IRConcrPlateCodeService2	1778		

GroupGlobally 1240	CalculationOptions 1209
GroupingMethod 1240	Concrete 1209
IRConcrReinforcement fields 1239	CreateCalculationNoteRtf 1212
IRConcrReinforcement members 1238	CreateFromObjects 1212
IRConcrReinforcement methods 1240	Geometry 1209
IsFrozen 1240	IRConcrSlab fields 1209
LockRefresh 1241	IRConcrSlab members 1208
RemoveAll 1241	IRConcrSlab methods 1211
RemoveParametric 1242	IsSelected 1210
IRConcrReinforcingBar 1242	Name 1210
about IRConcrReinforcingBar 1242	PatternOptions 1210
CreateCopy 1247	Reinforcement 1210
DirectionX 1243	Save 1212
DirectionY 1244	StartWizard 1213
DirectionZ 1244	Steel 1211
GetROBarEditor 1247	UniqueId 1211
GroupNumber 1244	IRConcrSlabCalculationOptions 1213
IRConcrReinforcingBar fields 1243	about IRConcrSlabCalculationOptions 1213
IRConcrReinforcingBar members 1242	Cover 1213
IRConcrReinforcingBar methods 1246	CoverLateral 1214
IsParametric 1244	IRConcrSlabCalculationOptions fields 1213
Position 1245	IRConcrSlabCalculationOptions members 1213
ROBarEditorId 1245	IRConcrSlabGeometry 1214
SetBarShape 1247	about IRConcrSlabGeometry 1214
SetDirections 1247	ConcreteVolume 1215
SetROBarEditor 1248	IRConcrSlabGeometry fields 1214
Span 1245	IRConcrSlabGeometry members 1214
SteelType 1245	ShutteringArea 1215
Subtype 1246	IRConcrSlabPatternOptions 1215
Type 1246	about IRConcrSlabPatternOptions 1215
Weight 1246	IRConcrSlabPatternOptions fields 1216
IRConcrReinforcingBars 1248	IRConcrSlabPatternOptions members 1215
about IRConcrReinforcingBars 1248	ReinforcementSegment 1216
Add 1249	ReinforcementType 1216
Count 1249	IRConcrSlabRequiredReinfCalcParams 1752
IRConcrReinforcingBars fields 1248	about IRConcrSlabRequiredReinfCalcParams 1752
IRConcrReinforcingBars members 1248	CasesACC 1753
IRConcrReinforcingBars methods 1249	CasesSLS 1753
Item 1249	CasesULS 1753
IRConcrSlab 1208	DisplayErrors 1754
about IRConcrSlab 1208	ForcesReduction 1754
Activate 1211	GloballyAvgDesginForces 1754

IRConcrSlabRequiredReinfCalcParams fields 1752	IRConcrSteelNames members 1251
IRConcrSlabRequiredReinfCalcParams members 1752	Item 1252
Method 1754	IRConcrSteelParams 1703
Panels 1755	about IRConcrSteelParams 1703
IRConcrSlabRequiredReinfEngine 1755	Ey 1704
about IRConcrSlabRequiredReinfEngine 1755	Ft 1704
Calculate 1756	Ft_calc 1704
GetCalculatedPanels 1757	Fy 1704
IRConcrSlabRequiredReinfEngine fields 1756	Fy_calc 1705
IRConcrSlabRequiredReinfEngine members 1755	Grade 1705
IRConcrSlabRequiredReinfEngine methods 1756	IRConcrSteelParams fields 1703
Params 1756	IRConcrSteelParams members 1703
IRConcrSlabRnfSegmentType 1220	IRConcrSteelStrengths 1257
IRConcrSlabRnfType 1219	about IRConcrSteelStrengths 1257
IRConcrSlabSupportType 1220	IRConcrSteelSurfaceType 1256
IRConcrSlabWizard 1216	IRConcrSteelType 1252
about IRConcrSlabWizard 1216	IRConcrWall 1226
AddColumnSupport 1218	about IRConcrWall 1226
AddOpening 1218	Activate 1229
AddOpeningNode 1218	Calculate 1229
AddSlabNode 1219	Concrete 1227
AddWallSupport 1219	CreateCalculationNoteRtf 1229
Finish 1219	CreateFromObjects 1230
IRConcrSlabWizard fields 1217	Geometry 1227
IRConcrSlabWizard members 1217	IRConcrWall fields 1227
IRConcrSlabWizard methods 1217	IRConcrWall members 1226
Thickness 1217	IRConcrWall methods 1229
IRConcrSteel 1253	IsSelected 1227
about IRConcrSteel 1253	Name 1228
AvailableNames 1254	Reinforcement 1228
AvailableStrengths 1254	Save 1230
BarDiameters 1254	Steel 1228
BarsDatabasePath 1255	Uniquelid 1228
CharacteristicStrength 1255	Verify 1230
IRConcrSteel fields 1253	IRConcrWallGeometry 1230
IRConcrSteel members 1253	about IRConcrWallGeometry 1230
SelectedName 1255	Height 1231
SurfaceType 1255	IRConcrWallGeometry fields 1231
IRConcrSteelNames 1251	IRConcrWallGeometry members 1231
about IRConcrSteelNames 1251	LeftBoundaryLength 1232
Count 1252	LeftBoundaryThickness 1232
IRConcrSteelNames fields 1252	Length 1232

Name 1233	IRDimClient methods 1925
Openings 1233	IRDimCodeResCB71 1855
RightBoundaryLength 1233	about IRDimCodeResCB71 1855
RightBoundaryThickness 1233	BadWidthBfUnderFire 1860
Thickness 1234	BuckCoefKMax 1861
IRConcrWallOpening 1234	BuckCoefKy 1861
about IRConcrWallOpening 1234	BuckCoefKz 1861
IRConcrWallOpening fields 1234	BuckLengthComposBeamLey 1861
IRConcrWallOpening members 1234	BuckLengthComposBeamLez 1862
Lx 1235	BuckLengthLfY 1862
Lz 1235	BuckLengthLfZ 1862
Name 1235	BuckSlendComposBeamLaym 1862
PosX 1235	BuckSlendComposBeamLazm 1863
PosZ 1236	BuckSLendLay 1863
IRConcrWallOpenings 1236	BuckSlendLaz 1863
about IRConcrWallOpenings 1236	CoefDependOnAngle 1863
Add 1237	CoefDependOnHeight 1864
Count 1237	CoefDependOnHumAndBend 1864
IRConcrWallOpenings fields 1236	CoefDependOnHumAndComp 1864
IRConcrWallOpenings members 1236	CoefDependOnWidth 1864
IRConcrWallOpenings methods 1237	EfficiencyRatio 1864
Item 1237	FireHeightInMidSpanH 1865
IRDimBuckDiagramCB71 1878	FireInerMomInMidSpanly 1865
IRDimBuckDiagramEC3 1810	FireInerMomInMidSpanlz 1865
IRDimCalcState 1907	FireInerMomIx 1865
about IRDimCalcState 1907	FireInerMomly 1866
GetParam 1908	FireInerMomIz 1866
GetValue 1908	FireLftEdgeDistVpy 1866
IRDimCalcState members 1908	FireLowEdgeDistVpz 1866
IRDimCalcState methods 1908	FireProtectCoefK2 1867
IsFlagSet 1909	FireProtection 1867
SetFlag 1909	FireProtectionPosition 1867
SetParam 1909	FireRedCoefK1b 1867
SetValue 1909	FireRedCoefK1h 1868
IRDimCalcStateFlagType 1906	FireResistance 1868
IRDimCalcStateParamType 1907	FireRgtEdgeDistVy 1868
IRDimCalcStateParamValue 1907	FireSecAreaInMidSpanS 1868
IRDimCalcStateValueType 1907	FireSecAreaS 1869
IRDimClient 1925	FireSeModulWy 1869
about IRDimClient 1925	FireSeModulWZ 1869
GetRDimCodeService 1925	FireShearAreaSY 1869
IRDimClient members 1925	FireShearAreaSZ 1870

FireUprEdgeDistVz 1870	BendParamBetMy 1820
FireWidthAtMembEndB 1870	BuckCoeffMinXi 1820
FireWidthAtMembEndH 1870	BuckCoeffXy 1820
FireWidthB 1871	BuckCoeffXz 1820
FireWidthH 1871	BuckCurveCoeffAlfy 1821
FireWidthInMidSpanB 1871	BuckCurveCoeffAlfz 1821
IRDimCodeResCB71 fields 1858	BuckCurveNumbY 1821
IRDimCodeResCB71 members 1855	BuckCurveNumbZ 1821
IsBuckY 1871	BuckParamFiy 1822
IsBuckZ 1872	BuckParamFiz 1822
LatBuckCoefKinst 1872	BuckParamKy 1822
LatBuckLengthCoef 1872	BuckParamKz 1822
LatBuckLengthLd 1872	BuckRelSlendLaby 1823
LatBuckRelSlendLam 1873	BuckRelSlendLabz 1823
LatBuckStrsCrit 1873	BuckSLendLamy 1823
MatAxCompResist 1873	BuckStrenNbyrd 1823
MatAxTensResist 1873	BuckStrenNbzrd 1824
MatBendResist 1874	BucSlendLamz 1824
MatShearResist 1874	ClassOfSect 1824
MatTrCompResist 1874	ClassOfSectElem1 1824
MatTrTensResist 1874	ClassOfSectElem2 1825
StrsBend 1875	ClassOfSectElem3 1825
StrsBendInCompEdgeMY 1875	ClassOfSectElem4 1825
StrsBendInCompEdgeMZ 1875	CompParamBetaA 1825
StrsBendInCurveBeam 1875	CritMomenMcr 1826
StrsComp 1876	EffectiveMomenMeff 1826
StrsFinal 1876	EffSectAreaSeff 1826
StrsLftEdgeMZ 1876	EffSectModulWyeff 1826
StrsLowEdgeMY 1876	EffSectModulWzeff 1827
StrsRgtEdgeMZ 1877	ElastMomStrenMelyrd 1827
StrsShearY 1877	ElastMomStrenMelzrd 1827
StrsShearZ 1877	ElastSectModulWyel 1827
StrsTens 1877	ElastSectModulWzel 1828
StrsUprEdgeMY 1878	FlangeAreaAf 1828
TimberType 1878	FlangeAreaAw 1828
IRDimCodeResEC3 1810	InteractParamAlfa 1828
about IRDimCodeResEC3 1810	InteractParamBeta 1829
AxForceExcentrEny 1818	InteractParamMilt 1829
AxForcExcentrEnz 1819	InteractParamMiy 1829
BendParamBetaMlty 1819	InteractParamMiz 1829
BendParamBetaMltz 1819	IRDimCodeResEC3 fields 1815
BendParamBetaMz 1819	IRDimCodeResEC3 members 1811

IsBuckY 1830	ShearStrenTplzrd 1840
IsBuckZ 1830	StrsComp 1840
LaodLevel 1830	StrsLftEdgeMZ 1841
LatBuckCoeffXlt 1830	StrsLowEdgeMY 1841
LatBuckLengthLd 1831	StrsRgtEdgeMZ 1841
LatBuckMomStrenmbrd 1831	StrsShearY 1841
LatBuckParamBetaW 1831	StrsShearZ 1842
LatBuckParamC1 1831	StrsTens 1842
LatBuckParamC2 1832	StrsUprEdgeMY 1842
LatBuckParamFilt 1832	TensStrenNurd 1842
LatBuckParamKlt 1832	TorsMomInertIlt 1843
LatBuckParamLd 1832	UppFlanSlend 1843
LatBuckSlendLamlt 1833	WarpingConstantlw 1843
LowFlanSlend1 1833	WebSlend 1843
LowFlanSlend2 1833	WebSlend1 1844
MaEffRatio 1833	IRDimCodeService 1923
MaterCoeffGamma0 1834	about IRDimCodeService 1923
MaterCoeffGamma1 1834	EditMembDef 1924
MaterCoeffGamma2 1834	GetDefaultMembDef 1924
MaterialCapacityFy 1834	GetMembCalc 1925
MaxBuckSlend 1835	IRDimCodeService members 1924
MomStrengthMpzEuro 1835	IRDimCodeService methods 1924
OverallBuckStrenNbrd 1835	IRDimEffDef 1890
PartEffRatio1 1835	about IRDimEffDef 1890
PartEffRatio2 1836	Clear 1894
PartEffRatio3 1836	IRDimEffDef fields 1891
PartEffRatio4 1836	IRDimEffDef members 1890
PartEffRatio5 1836	IRDimEffDef methods 1893
PlastAxForcStrenNplrd 1837	MX 1892
PlastCompStrenNcrd 1837	MY 1892
PlastMomStrenMplyrd 1837	MZ 1892
PlastMomStrenMplzrd 1837	N 1892
PlastSectModulWypl 1838	QY 1893
PlastSectModulWzpl 1838	QZ 1893
PlastTensStrenNtrd 1838	Read_IntPsEff_M_1P4L 1894
ReducMomStrenMcryrd 1838	Read_IntPsEff_M_3P4L 1894
ReducMomStrenMczrd 1839	Read_IntPsEff_M_MID 1895
ReducMomStrenMnyrd 1839	Read_IntPsEff_M1 1895
ReducMomStrenMnzrd 1839	Read_IntPsEff_M12 1895
ReducMomStrenMvyrd 1839	Read_IntPsEff_M2 1895
ReducMomStrenMvzrd 1840	Read_IntPsEff_MN_MAX 1895
ShearStrenTplyrd 1840	Read_IntPsEff_MP_MAX 1896

Read_M_1P4L 1896	IRDimMatDefDblExValType 1900
Read_M_3P4L 1896	IRDimMatDefLongExValType 1900
Read_M_MID 1896	IRDimMatDefType 1899
Read_M1 1896	IRDimMatDefValType 1899
Read_M12 1897	IRDimMembCalc 1919
Read_M2 1897	about IRDimMembCalc 1919
Read_MN_MAX 1897	CalculBuckling 1922
Read_MP_MAX 1897	CalculMember 1922
ReadParam 1897	GetRatio 1922
WriteForces 1898	GetResultsInterface 1922
WriteIntPsEffSet1 1898	IRDimMembCalc fields 1920
WriteIntPsEffSet2 1898	IRDimMembCalc members 1920
WriteParam 1898	IRDimMembCalc methods 1921
WriteValuesSet1 1899	IsExtraMomentsSet1 1920
WriteValuesSet2 1899	IsExtraMomentsSet2 1921
IRDimEffDefDirType 1890	IsIntPointsMomentsSet1 1921
IRDimEffDefIntPsType 1890	IsIntPointsMomentsSet2 1921
IRDimEffDefParamType 1889	SetCalcState 1922
IRDimFireMemberPositionsCB71 1880	SetEffDef 1923
IRDimFireProtectionTimesCB71 1880	SetMatDef 1923
IRDimLatBuckCoeffDiagramEC3 1809	SetMembDef 1923
IRDimLaterBuckTypeCB71 1879	SetProfDef 1923
IRDimLaterBuckTypeEC3 1809	IRDimMembCalcBuckType 1911
IRDimLoadLevelCB71 1879	IRDimMembCalcRetValue 1910
IRDimLoadLevelEC3 1809	IRDimMembDef 1885
IRDimLoadTypeCB71 1879	about IRDimMembDef 1885
IRDimLoadTypeEC3 1808	ClientID 1886
IRDimMatDef 1900	DeflYZRelLimit 1888
about IRDimMatDef 1900	DispIXYRelLimit 1888
IRDimMatDef fields 1901	IRDimMembDef fields 1886
IRDimMatDef members 1900	IRDimMembDef members 1885
IRDimMatDef methods 1902	IRDimMembDef methods 1887
Name 1901	IsBuckCoefConst 1888
ReadDoubleExtraValue 1902	IsDeflectionYZ 1888
ReadLongExtraValue 1902	IsDisplacementXY 1888
ReadValue 1902	Length 1886
SecondName 1901	LengthYZUV 1889
SetNames 1903	MatType 1887
Type 1901	Name 1887
WriteDoubleExtraValue 1903	Retrieve 1889
WriteLongExtraValue 1903	Store 1889
WriteValue 1903	Type 1887

IRDimMembDefBucklingDataType 1885	BucklingDiagramY 1803
IRDimMembDefDeflDataType 1885	BucklingDiagramZ 1803
IRDimMembDefDispDataType 1885	HotRolledPipes 1803
IRDimMembDefLengthDataType 1884	IRDimMembParamsEC3 fields 1801
IRDimMembDefMatType 1884	IRDimMembParamsEC3 members 1800
IRDimMembDefType 1884	LatBuckType 1804
IRDimMembParamsCB71 1845	LatCoeffLowerFlange 1804
about IRDimMembParamsCB71 1845	LatCoeffLowerFlangeValue 1804
ArcBeamCheck 1847	LatCoeffUpperFlange 1804
ArcRadius 1848	LatCoeffUpperFlangeValue 1805
BuckComposBeamCoefCey 1848	LoadLevel 1805
BuckComposBeamCoefCez 1848	LoadLevelValue 1805
BuckIsComposedY 1848	LoadTypeY 1805
BuckIsComposedZ 1849	LoadTypeZ 1806
BuckLengthCoeffZ 1849	MaterCoeffGamma0 1806
BucklingDiagramY 1849	MaterCoeffGamma1 1806
BucklingDiagramZ 1849	RelLimitDeflUy 1806
BuckLengthCoeffY 1850	RelLimitdeflUz 1807
FireContinProtForComposBeam 1850	TensAreaNetGros 1807
FireLftSideProt 1850	TubeControl 1807
FireLowSideProt 1850	YieldStrengthType 1808
FireMembPosition 1851	YieldStrengthValue 1808
FireProtectionTime 1851	IRDimMembRes 1911
FireRequiredResist 1851	about IRDimMembRes 1911
FireRgtSideProt 1851	BlockCount 1915
FireUprSideProt 1852	CreateResWnd 1916
ForceDistFromBeginDistX 1852	EffRatio 1913
HumChangeDeltaH 1852	GetEffDefAccess 1916
InflDeflCoefTheta 1853	GetLineComponent 1916
IRDimMembParamsCB71 fields 1846	GetLineType 1916
IRDimMembParamsCB71 members 1845	GetMatDefAccess 1916
LatBuckType 1853	GetMaxLineNo 1917
LatCoeffLowerFlangeValue 1853	GetMembDefAccess 1917
LatCoeffUpperFlangeValue 1853	GetProfDefAccess 1917
LoadLevel 1854	IRDimMembRes fields 1913
LoadTypeY 1854	IRDimMembRes members 1912
TensAreaNetGros 1854	IRDimMembRes methods 1915
TubeControl 1854	IsLineActive 1917
IRDimMembParamsEC3 1800	IsStatement 1917
about IRDimMembParamsEC3 1800	Language 1913
BuckLengthCoeffY 1802	Ratio 1913
BuckLengthCoeffZ 1803	RecognizedPQ 1918

RefreshUnits 1918	ReadLong 1882
ReplaceMark 1918	ReadText 1883
ResOfCalc 1914	SeekSet 1883
Retrieve 1918	Size 1883
RtfFileName 1914	WriteDouble 1883
SetEffDef 1918	WriteLong 1883
SetMatDef 1919	WriteText 1884
SetMembDef 1919	IRDimStreamType 1881
SetProfDef 1919	IRDimUnits 1926
SlendY 1914	about IRDimUnits 1926
SlendZ 1914	AreRobotUnits 1927
Units 1914	Format 1927
IRDimMembResTableComp 1911	IRDimUnits fields 1926
IRDimMembResTableLineType 1911	IRDimUnits members 1926
IRDimMembSrv 1909	IRDimUnits methods 1927
about IRDimMembSrv 1909	ReadToUserCoef 1927
CheckLabelName 1910	ReadUserName 1928
IRDimMembSrv members 1910	Refresh 1928
IRDimMembSrv methods 1910	IRDimUnitType 1926
Save 1910	IRDimYieldStrengthTypeEC3 1844
IRDimProfDef 1904	IRJointAnchor 1994
about IRDimProfDef 1904	about IRJointAnchor 1994
Clear 1906	AnchorPlate 1995
IRDimProfDef fields 1905	Bolts 1995
IRDimProfDef members 1904	IRJointAnchor fields 1994
IRDimProfDef methods 1905	IRJointAnchor members 1994
IsVar 1905	Tige 1995
Name 1905	Type 1995
ReadValue 1906	IRJointAnchorBolt 1991
Type 1905	about IRJointAnchorBolt 1991
WriteValue 1906	IRJointAnchorBolt fields 1991
IRDimProfDefItemType 1904	IRJointAnchorBolt members 1991
IRDimProfDefType 1903	Length1 1991
IRDimProfDefValType 1904	Length2 1992
IRDimServer 1928	Length3 1992
about IRDimServer 1928	Length4 1992
IRDimStream 1881	IRJointAnchorPlate 1992
about IRDimStream 1881	about IRJointAnchorPlate 1992
Clear 1882	IRJointAnchorPlate fields 1993
IRDimStream members 1881	IRJointAnchorPlate members 1993
IRDimStream methods 1881	Type 1993
ReadDouble 1882	IRJointAnchorPlateType 2009

IRJointAnchorType 1993	BeamLeft 2130
IRJointAngle 2023	BeamRight 2130
about IRJointAngle 2023	DistGirderFlangeToBeamFlangeLeft 2130
DistFromUpperBeamEdge 2024	DistGirderFlangeToBeamFlangeRight 2131
IRJointAngle fields 2023	DistGirderToBeamLeft 2131
IRJointAngle members 2023	DistGirderToBeamRight 2131
Length 2024	ElementType 2131
Profile 2024	Girder 2132
ProfilePosition 2025	IRJointBeamGirder fields 2129
Type 2025	IRJointBeamGirder members 2128
IRJointAngleBolts 2020	PlateLeft 2132
about IRJointAngleBolts 2020	PlateRight 2132
Area 2021	SeatLeftDown 2133
ClassName 2021	SeatLeftUp 2133
Diameter 2021	SeatRightDown 2133
DiameterName 2021	SeatRightUp 2133
DistFromUpperEdge 2022	StiffenerLeft 2134
DistFromWeb 2022	StiffenerRight 2134
Friction 2022	IRJointBeamGirderAngle 2141
IRJointAngleBolts fields 2020	about IRJointBeamGirderAngle 2141
IRJointAngleBolts members 2020	BoltsBeam 2142
Rows 2022	BoltsGirder 2142
Spacing 2023	Element 2142
IRJointAngleLoad 2030	IRJointBeamGirderAngle fields 2142
about IRJointAngleLoad 2030	IRJointBeamGirderAngle members 2141
Fz 2031	IRJointBeamGirderBeam 2143
IRJointAngleLoad fields 2031	about IRJointBeamGirderBeam 2143
IRJointAngleLoad members 2030	CutEnd 2143
IRJointAngleProfilePosition 2030	IRJointBeamGirderBeam fields 2143
IRJointAngleType 2025	IRJointBeamGirderBeam members 2143
IRJointAngleUnit 2167	Profile 2144
IRJointBeamCut 2025	IRJointBeamGirderBolts 2112
about IRJointBeamCut 2025	about IRJointBeamGirderBolts 2112
HeightLower 2026	Area 2113
HeightUpper 2026	ClassName 2113
IRJointBeamCut fields 2026	Cols 2114
IRJointBeamCut members 2026	Diameter 2114
Length 2027	DiameterName 2114
IRJointBeamGirder 2127	DistFromUpperBeamEdge 2115
about IRJointBeamGirder 2127	DistFromVertBeamEdge 2115
AngleLeft 2129	Friction 2115
AngleRight 2130	IRJointBeamGirderBolts fields 2113

IRJointBeamGirderBolts members 2112	Cols 2138
Rows 2115	Diameter 2139
SpacingH 2116	DiameterName 2139
SpacingV 2116	DistFromPerpendicularArmEdge 2139
IRJointBeamGirderLoad 2134	DistFromVertEdge 2139
about IRJointBeamGirderLoad 2134	Friction 2140
IRJointBeamGirderLoad fields 2135	IRJointBeamGirderSeatBolts fields 2138
IRJointBeamGirderLoad members 2134	IRJointBeamGirderSeatBolts members 2137
LFx 2135	IRJointBeamGirderStiffener 2116
LFz 2135	about IRJointBeamGirderStiffener 2116
LMy 2136	BoltsBeam 2118
RFx 2136	ConnectorsType 2118
RFz 2136	CutHeightDown 2118
RMy 2137	CutHeightUp 2118
IRJointBeamGirderPlate 2120	CutLength 2119
about IRJointBeamGirderPlate 2120	IRJointBeamGirderStiffener fields 2117
BoltsBeam 2121	IRJointBeamGirderStiffener members 2117
ConnectorsType 2122	Length 2119
DistFromUpperBeamEdge 2122	Material 2119
IRJointBeamGirderPlate fields 2121	Thick 2120
IRJointBeamGirderPlate members 2121	WeldsBeam 2120
Length 2122	IRJointBearingPlate 2003
Material 2123	about IRJointBearingPlate 2003
Thick 2123	IRJointBearingPlate fields 2004
WeldsBeam 2123	IRJointBearingPlate members 2004
WeldsGirder 2124	ThickBearingBar 2004
Width 2124	IRJointBolts 2153
IRJointBeamGirderSeat 2124	about IRJointBolts 2153
about IRJointBeamGirderSeat 2124	Area 2154
BoltsBeam 2125	ClassName 2155
BoltsGirder 2126	Cols 2155
ConnectorsToWebType 2126	Diameter 2155
IRJointBeamGirderSeat fields 2125	DiameterName 2155
IRJointBeamGirderSeat members 2125	Friction 2156
Length 2126	Height1 2156
Material 2126	IRJointBolts fields 2154
Section 2127	IRJointBolts members 2154
WeldsGirder 2127	Rows 2156
IRJointBeamGirderSeatBolts 2137	IRJointBoltType 2157
about IRJointBeamGirderSeatBolts 2137	IRJointColumnBasePlateCalcModel 1988
Area 2138	IRJointColumnBracket 2006
ClassName 2138	about IRJointColumnBracket 2006

Exist 2009  
Height 2007  
IRJointColumnBracket fields 2007  
IRJointColumnBracket members 2007  
IRJointColumnBracket methods 2009  
Length 2008  
Thick 2008  
VThick 2008  
Width 2008  
IRJointColumnSquare 2014  
about IRJointColumnSquare 2014  
Diameter 2014  
IRJointColumnSquare fields 2014  
IRJointColumnSquare members 2014  
Length 2015  
Profile 2015  
IRJointComponentType 2168  
IRJointConcreteColumn 1981  
about IRJointConcreteColumn 1981  
Base 1982  
CalcModel 1982  
Depth 1982  
Foundation 1982  
IRJointConcreteColumn fields 1981  
IRJointConcreteColumn members 1981  
Materials 1983  
Profile 1983  
SpreadFootingType 1983  
IRJointConcreteColumnFoundation 1984  
about IRJointConcreteColumnFoundation 1984  
IRJointConcreteColumnFoundation fields 1984  
IRJointConcreteColumnFoundation members 1984  
par\_a1 1985  
par\_b 1985  
par\_b1 1985  
par\_b2 1986  
par\_h 1986  
par\_h1 1986  
par\_h2 1986  
par\_l 1987  
par\_l1 1987  
par\_l2 1987  
par\_o1 1987  
par\_o2 1988  
IRJointConcreteColumnLoad 1988  
about IRJointConcreteColumnLoad 1988  
Fx 1989  
FxAssemb 1989  
Fy 1990  
Fz 1990  
IRJointConcreteColumnLoad fields 1989  
IRJointConcreteColumnLoad members 1989  
My 1990  
Mz 1990  
IRJointConcreteMaterials 1979  
about IRJointConcreteMaterials 1979  
CoeffBA 1980  
Dosage 1980  
IRJointConcreteMaterials fields 1980  
IRJointConcreteMaterials members 1979  
Sigma 1980  
IRJointConnection 2147  
about IRJointConnection 2147  
GetFromRobot 2149  
IRJointConnection fields 2148  
IRJointConnection members 2148  
IRJointConnection methods 2149  
SetToRobot 2149  
Type 2148  
WFRelPos 2148  
IRJointConnectionDefType 2144  
IRJointConnectionInfo 2145  
about IRJointConnectionInfo 2145  
DefType 2146  
Elements 2146  
IRJointConnectionInfo fields 2145  
IRJointConnectionInfo members 2145  
Node 2146  
Number 2146  
Type 2147  
Uniqueld 2147  
IRJointConnectionServer 2157  
about IRJointConnectionServer 2157  
Calculate 2159

CalculateNote 2159	Stiff 1978
Count 2158	Washer 1978
Create 2160	Wedge 1978
CreateInfo 2160	IRJointFixedLoad 1970
Delete 2160	about IRJointFixedLoad 1970
Exist 2160	Fx 1971
Find 2161	Fy 1971
FindWithId 2161	Fz 1971
Get 2161	IRJointFixedLoad fields 1971
GetAllNumbers 2162	IRJointFixedLoad members 1970
GetInfo 2162	My 1972
IRJointConnectionServer fields 2158	Mz 1972
IRJointConnectionServer members 2157	IRJointFootBolts 1996
IRJointConnectionServer methods 2158	about IRJointFootBolts 1996
IRJointConnectionType 2144	Area 1997
IRJointConnectorsType 2140	ClassName 1997
IRJointElementType 2140	Diameter 1997
IRJointExtType 2167	DiameterName 1997
IRJointFixedColumnBase 1972	Distance 1998
about IRJointFixedColumnBase 1972	Friction 1998
Anchor 1973	IRJointFootBolts fields 1996
Base 1974	IRJointFootBolts members 1996
BasePlateMaterial 1974	Rows 1998
Bolts 1974	SpacingH 1998
ComplexStiff 1974	SpacingV 1999
FootPlate 1975	IRJointFootMaterials 2012
IRJointFixedColumnBase fields 1973	about IRJointFootMaterials 2012
IRJointFixedColumnBase members 1972	CementAmount 2012
Materials 1975	ConcrClass 2013
NodeNumber 1975	ConcrSteelCoeff 2013
Profile 1975	IRJointFootMaterials fields 2012
SimpleStiff 1976	IRJointFootMaterials members 2012
StiffType 1976	PlateSigma 2013
Washer 1976	PlateYoung 2013
Wedge 1976	IRJointFootPlate 1999
Welds 1977	about IRJointFootPlate 1999
IRJointFixedColumnBaseStiffType 1979	Diameter 2000
IRJointFixedFootWelds 1977	IRJointFootPlate fields 2000
about IRJointFixedFootWelds 1977	IRJointFootPlate members 2000
FootPlate 1978	Type 2000
IRJointFixedFootWelds fields 1977	IRJointFootPlateType 1999
IRJointFixedFootWelds members 1977	IRJointFootStiffenerComplex 2017

about IRJointFootStiffenerComplex 2017	Diameter 2051
Height 2018	DistanceH1 2051
IRJointFootStiffenerComplex fields 2018	Friction 2051
IRJointFootStiffenerComplex members 2017	IRJointGussetBoltsDiag fields 2050
Length 2018	IRJointGussetBoltsDiag members 2050
ThickPlateHor 2018	Rows 2052
ThickStiff 2019	Spacing 2052
IRJointFootStiffenerHoriz 2011	IRJointGussetCornersType 2067
about IRJointFootStiffenerHoriz 2011	IRJointGussetCross 2084
IRJointFootStiffenerHoriz fields 2011	about IRJointGussetCross 2084
IRJointFootStiffenerHoriz members 2011	BoltsClassVector 2086
Thick 2011	BoltsDimensionNamesVector 2086
IRJointFootStiffenerSimple 2015	BoltsDimensionsVector 2087
about IRJointFootStiffenerSimple 2015	DiagLeftLower 2087
Height 2016	DiagLeftUpper 2087
IRJointFootStiffenerSimple fields 2016	DiagRightLower 2087
IRJointFootStiffenerSimple members 2015	DiagRightUpper 2088
Length 2016	FixType_LeftL_RightU 2088
Thick 2016	FixType_LeftLower 2088
Type 2017	FixType_LeftU_RightL 2088
Width 2017	FixType_LeftUpper 2089
IRJointFootStiffenerVert 2009	FixType_RightLower 2089
about IRJointFootStiffenerVert 2009	FixType_RightUpper 2089
IRJointFootStiffenerVert fields 2010	GussetPlate 2090
IRJointFootStiffenerVert members 2010	IRJointGussetCross fields 2085
Length 2010	IRJointGussetCross members 2084
Thick 2010	ProfileCutting 2090
WidthSpacing 2011	IRJointGussetCrossLoad 2107
IRJointFootStiffType 2019	about IRJointGussetCrossLoad 2107
IRJointFootWelds 2004	Fx_LeftLower 2107
about IRJointFootWelds 2004	Fx_LeftUpper 2108
Bearing 2005	Fx_RightLower 2108
FootPlate 2005	Fx_RightUpper 2108
IRJointFootWelds fields 2005	IRJointGussetCrossLoad fields 2107
IRJointFootWelds members 2005	IRJointGussetCrossLoad members 2107
Stiff 2006	IRJointGussetCrossPlate 2076
Washer 2006	about IRJointGussetCrossPlate 2076
Wedge 2006	B1 2077
IRJointGussetBoltsDiag 2049	B2 2077
about IRJointGussetBoltsDiag 2049	B3 2078
BoltAxisShift 2050	B4 2078
ClassName 2050	CornersType 2078

CutH1 2078	DiagonalUp 2103
CutH2 2079	FixType_DiagonalLeft 2103
CutH3 2079	FixType_DiagonalRight 2104
CutH4 2079	FixType_DiagonalUp 2104
CutV1 2080	FixType_FlangeLeft 2104
CutV2 2080	FixType_FlangeRight 2104
CutV3 2080	FlangeLeft 2105
CutV4 2081	FlangeRight 2105
H1 2081	GussetPlate 2105
H2 2081	IRJointGussetFlange fields 2101
H3 2082	IRJointGussetFlange members 2100
H4 2082	ProfileCutting 2105
Height 2082	IRJointGussetFlangeLoad 2108
HOffset 2082	about IRJointGussetFlangeLoad 2108
HorizontalOffset 2083	Fx_FlangeLeft 2109
IRJointGussetCrossPlate fields 2077	Fx_FlangeRight 2110
IRJointGussetCrossPlate members 2076	Fx_LeftUpper 2110
Length 2083	Fx_RightUpper 2110
Material 2083	Fx_Upper 2110
Thick 2084	IRJointGussetFlangeLoad fields 2109
VOffset 2084	IRJointGussetFlangeLoad members 2109
IRJointGussetCrossProfileCutting 2075	IRJointGussetFlangePlate 2090
IRJointGussetDiagonale 2054	about IRJointGussetFlangePlate 2090
about IRJointGussetDiagonale 2054	B1 2092
BarNumber 2055	B2 2092
BoltsDiag 2055	B3 2093
DistanceEC 2055	B4 2093
Exist 2056	CornersType 2093
IRJointGussetDiagonale fields 2054	CutH1 2093
IRJointGussetDiagonale members 2054	CutH2 2094
Position 2056	CutH3 2094
Profile 2056	CutH4 2094
WeldsDiag 2056	CutV1 2095
IRJointGussetDiagonalePositionType 2111	CutV2 2095
IRJointGussetFixType 2057	CutV3 2095
IRJointGussetFlange 2100	CutV4 2096
about IRJointGussetFlange 2100	EH 2096
BoltsClassVector 2102	EV 2096
BoltsDimensionNamesVector 2102	Gheight 2097
BoltsDimensionsVector 2102	H1 2097
DiagonalLeft 2103	H2 2097
DiagonalRight 2103	H3 2097

H4 2098	about IRJointGussetSimpleAttachBoltsVertical 2072
HOffset 2098	DistanceEB 2073
IRJointGussetFlangePlate fields 2091	DistanceH1 2073
IRJointGussetFlangePlate members 2091	IRJointGussetSimpleAttachBoltsVertical fields 2073
Length 2098	IRJointGussetSimpleAttachBoltsVertical members 2073
Material 2099	Rows 2074
NewPlateDefinition 2099	Spacing 2074
Thick 2099	IRJointGussetSimpleAttachment 2067
Type 2099	about IRJointGussetSimpleAttachment 2067
VOffset 2100	BoltsAttach 2068
IRJointGussetFlangePlateRegularType 2111	FixType 2068
IRJointGussetFlangeProfileCutting 2090	IRJointGussetSimpleAttachment fields 2067
IRJointGussetSimple 2057	IRJointGussetSimpleAttachment members 2067
about IRJointGussetSimple 2057	WeldsAttach 2068
Attachment 2058	WeldType 2068
BoltsClassVector 2059	IRJointGussetSimpleAttachWelds 2074
BoltsDimensionNamesVector 2059	about IRJointGussetSimpleAttachWelds 2074
BoltsDimensionsVector 2059	IRJointGussetSimpleAttachWelds fields 2075
Diagonale 2060	IRJointGussetSimpleAttachWelds members 2074
FixType 2060	ThickEdgeA 2075
GussetPlate 2060	ThickEdgeB 2075
IRJointGussetSimple fields 2058	IRJointGussetSimpleLoad 2106
IRJointGussetSimple members 2057	about IRJointGussetSimpleLoad 2106
ProfilePosition 2060	Fx 2106
IRJointGussetSimpleAttachBolts 2069	IRJointGussetSimpleLoad fields 2106
about IRJointGussetSimpleAttachBolts 2069	IRJointGussetSimpleLoad members 2106
ClassName 2069	IRJointGussetSimplePlate 2061
Diameter 2070	about IRJointGussetSimplePlate 2061
Friction 2070	CornersType 2062
Horizontal 2070	DistanceD 2062
IRJointGussetSimpleAttachBolts fields 2069	DistanceEH 2062
IRJointGussetSimpleAttachBolts members 2069	DistanceEV 2063
Vertical 2070	H1 2063
IRJointGussetSimpleAttachBoltsHorizontal 2071	H2 2063
about IRJointGussetSimpleAttachBoltsHorizontal 2071	H3 2064
DistanceEB 2071	H4 2064
DistanceH1 2072	Height 2064
IRJointGussetSimpleAttachBoltsHorizontal fields 2071	IRJointGussetSimplePlate fields 2061
IRJointGussetSimpleAttachBoltsHorizontal members 2071	IRJointGussetSimplePlate members 2061
Rows 2072	Material 2064
Spacing 2072	Thick 2065
IRJointGussetSimpleAttachBoltsVertical 2072	V1 2065

V2 2065	TensionPlateUp 1952
V3 2066	VStiffLow 1952
V4 2066	VStiffUp 1952
Width 2066	WebBeamStiffenerVLower 1952
IRJointGussetSimpleProfilePosition 2057	WebBeamStiffenerVUpper 1953
IRJointGussetWeldsDiag 2052	WebColumnStiffenerHLower 1953
about IRJointGussetWeldsDiag 2052	WebColumnStiffenerHUpper 1953
IRJointGussetWeldsDiag fields 2053	WebPlate 1954
IRJointGussetWeldsDiag members 2052	WebStiffType 1954
Length1 2053	WeldBracketDownFlange 1954
Length2 2053	WeldBracketUpFlange 1954
ThickCorner 2053	WeldFlange 1955
ThickFlange 2054	WeldStiff 1955
IRJointKnee 1942	WeldWeb 1955
about IRJointKnee 1942	IRJointKneeBolts 1930
Beam 1945	about IRJointKneeBolts 1930
Bolts 1945	ClassName 1931
BracketLow 1945	Cols 1932
BracketUp 1946	DiameterName 1932
Column 1946	EqualSpac 1932
DefComponentMaterial 1946	Height1 1933
FixType 1946	IRJointKneeBolts fields 1931
FlanPlateLower 1947	IRJointKneeBolts members 1930
FlanPlatePosLower 1947	Rows 1933
FlanPlatePosUpper 1947	SpacingH 1933
FlanPlateUpper 1948	SpacingV 1933
IRJointKnee fields 1943	Symmetry 1934
IRJointKnee members 1942	IRJointKneeBracket 1934
IsBolted 1948	about IRJointKneeBracket 1934
IsDefComponentMaterialSet 1948	Angle 1935
KneeType 1948	AngleExt 1937
Material 1949	Exist 1935
MaterPlates 1949	Height 1936
Plate 1949	IRJointKneeBracket fields 1935
PlatePosition 1949	IRJointKneeBracket members 1934
PlatePositionLow 1950	IRJointKneeBracket methods 1937
StiffDiag 1950	Length 1936
StiffLow 1950	Material 1936
StiffTypeLow 1951	ThickFlange 1936
StiffTypeUp 1951	ThickWeb 1937
StiffUp 1951	Width 1937
TensionPlateLow 1951	IRJointKneeDiagonalStiff 1956

about IRJointKneeDiagonalStiff	1956	Type 2166	
Exist	1956	IRJointLoadType 2167	
IRJointKneeDiagonalStiff	fields 1956	IRJointPinnedColumnBase 1962	
IRJointKneeDiagonalStiff	members 1956	about IRJointPinnedColumnBase 1962	
Material	1957	Anchor 1963	
Thick	1957	Base 1963	
Type	1957	BasePlateMaterial 1964	
IRJointKneeDiagonalStiffType	1955	Bearing 1964	
IRJointKneeFixType	1941	BearingPlate 1964	
IRJointKneeLoad	1958	Bolts 1964	
about IRJointKneeLoad	1958	FootPlate 1965	
IRJointKneeLoad	fields 1959	IRJointPinnedColumnBase	fields 1963
IRJointKneeLoad	members 1959	IRJointPinnedColumnBase	members 1962
M	1959	Materials 1965	
N	1959	NodeNumber 1965	
Q	1959	Profile 1965	
IRJointKneeMaterials	1960	Square 1966	
IRJointKneeReinfType	1941	StiffHoriz 1966	
IRJointKneeStiffColumn	1957	StiffType 1966	
about IRJointKneeStiffColumn	1957	StiffVert 1966	
IRJointKneeStiffColumn	fields 1958	Washer 1967	
IRJointKneeStiffColumn	members 1958	Wedge 1967	
Thick	1958	Welds 1967	
IRJointKneeType	1941	IRJointPinnedColumnBaseStiffType 1961	
IRJointKneeWebPlate	1938	IRJointPinnedLoad 1967	
about IRJointKneeWebPlate	1938	about IRJointPinnedLoad 1967	
Bolts	1939	IRJointPinnedLoad	fields 1968
Height	1939	IRJointPinnedLoad	members 1968
IRJointKneeWebPlate	fields 1938	Nc 1968	
IRJointKneeWebPlate	members 1938	Nt 1968	
Material	1939	NTy 1969	
Thick	1939	NTz 1969	
Type	1940	Ty 1969	
WeldLong	1940	Tz 1969	
WeldTran	1940	IRJointPlate 2151	
Width	1940	about IRJointPlate 2151	
IRJointKneeWebStiffType	1955	AngleExt 2153	
IRJointLoad	2165	Exist 2152	
about IRJointLoad	2165	IRJointPlate	fields 2151
Cases	2166	IRJointPlate	members 2151
IRJointLoad	fields 2166	IRJointPlate	methods 2153
IRJointLoad	members 2166	Length 2152	

Material 2152	Profile 2042
Thick 2152	Section 2042
Width 2153	IRJointTubeFlangeProfile 2036
IRJointProfile 2162	about IRJointTubeFlangeProfile 2036
about IRJointProfile 2162	BarNumber 2037
Angle 2163	Excentr 2037
AngleExt 2164	Exist 2038
BarNumber 2163	IRJointTubeFlangeProfile fields 2037
Edit 2165	IRJointTubeFlangeProfile members 2037
EditComponent 2165	Material 2038
IRJointProfile fields 2163	Profile 2038
IRJointProfile members 2162	Section 2039
IRJointProfile methods 2164	IRJointTubeLoad 2045
Material 2164	about IRJointTubeLoad 2045
Section 2164	DiagLeftLowerM 2046
IRJointSpreadFootingType 1983	DiagLeftLowerN 2046
IRJointTube 2031	DiagLeftUpperM 2046
about IRJointTube 2031	DiagLeftUpperN 2046
DiagLeftLower 2033	DiagRightUpperM 2047
DiagLeftUpper 2033	DiagRightUpperN 2047
DiagRightUpper 2033	FlangeM 2047
Flange 2033	FlangeN 2048
IRJointTube fields 2032	FlangeQ 2048
IRJointTube members 2032	IRJointTubeLoad fields 2045
PostUpper 2034	IRJointTubeLoad members 2045
StiffHBracketMaterial 2034	PostUpperN 2048
StiffHoriz 2034	IRJointTubePostProfile 2042
StiffLateral 2035	about IRJointTubePostProfile 2042
TubeType 2035	BarNumber 2043
WeldDiag 2035	Exist 2043
WeldStiff 2035	IRJointTubePostProfile fields 2043
IRJointTubeDiagProfile 2039	IRJointTubePostProfile members 2042
about IRJointTubeDiagProfile 2039	Length 2043
Angle 2040	Material 2044
BarNumber 2040	Profile 2044
Exist 2040	Section 2044
Gap 2041	IRJointTubeProfileType 2036
IRJointTubeDiagProfile fields 2039	IRJointTubeType 2036
IRJointTubeDiagProfile members 2039	IRJointWebFlangeRelativePos 2147
Length 2041	IRJointWebType 2157
Material 2041	IRJointWedge 2001
Overlap 2041	about IRJointWedge 2001

IRJointWedge fields 2001	about IRobotAddInRegistrar 1549
IRJointWedge members 2001	Guid 1550
Length 2001	InstallMenu 1551
Profile 2002	IRobotAddInRegistrar fields 1549
Thick 2002	IRobotAddInRegistrar members 1549
Type 2002	IRobotAddInRegistrar methods 1551
Width 2002	ProductName 1550
XTypeMaterial 2003	ProgId 1550
IRJointWedgeType 2003	ProviderName 1551
IRJointWeld 2149	Register 1552
about IRJointWeld 2149	Unregister 1552
IRJointWeld fields 2150	IRobotAdvancedResultServer 919
IRJointWeld members 2150	about IRobotAdvancedResultServer 919
Strength 2150	Eigenvalues 920
Thick 2150	Eigenvectors 920
IRJointWithAngles 2027	FootfallValue 922
about IRJointWithAngles 2027	FRF 920
Angle 2028	IRobotAdvancedResultServer fields 920
BeamBolts 2028	IRobotAdvancedResultServer members 919
BeamCut 2029	IRobotAdvancedResultServer methods 921
BeamProfile 2029	MassSum 921
ColumnBolts 2029	SpectralCoeffs 921
ColumnProfile 2029	TimeHistory 921
Distance 2030	IRobotAdvancedSupportData 513
IRJointWithAngles fields 2028	about IRobotAdvancedSupportData 513
IRJointWithAngles members 2027	B 514
IRRobotActiveModelType 1351	D 514
IRRobotAddIn 1541	EquivalentElasticity 515
about IRobotAddIn 1541	H 515
Connect 1542	IRobotAdvancedSupportData fields 514
Disconnect 1543	IRobotAdvancedSupportData members 513
DoCommand 1543	IsEquivalentElasticity 515
GetExpectedVersion 1543	Type 515
InstallCommands 1544	IRobotAdvancedSupportType 513
IRRobotAddIn members 1541	IRobotApplication 1528
IRRobotAddIn methods 1542	about IRobotApplication 1528
IRRobotAddInMngr 1544	CmpntFactory 1529
about IRobotAddInMngr 1544	Interactive 1530
InstallCommand 1545	IRobotApplication fields 1529
IRRobotAddInMngr members 1544	IRobotApplication members 1528
IRRobotAddInMngr methods 1544	IRobotApplication methods 1532
IRRobotAddInRegistrar 1549	Is360 1530

Kernel 1530	IRobotBackgroundLayers members 1319
LicenseCheckEntitlement 1533	IRobotBackgroundLayers methods 1320
Preferences 1530	IsImported 1321
ProgramVersion 1531	SetImported 1321
Project 1531	IRobotBackgroundServer 1331
Quit 1533	about IRobotBackgroundServer 1331
UserControl 1531	Create 1332
Version 1532	Get 1332
Visible 1532	GetAllNumbers 1332
Window 1532	IRobotBackgroundServer members 1331
IRobotBackground 1325	IRobotBackgroundServer methods 1331
about IRobotBackground 1325	Remove 1333
Color 1326	RemoveAll 1333
FilePath 1327	IRobotBackgroundVisibilityRange 1329
InsertParams 1327	about IRobotBackgroundVisibilityRange 1329
IRobotBackground fields 1326	FromPos 1330
IRobotBackground members 1326	IRobotBackgroundVisibilityRange fields 1330
IRobotBackground methods 1328	IRobotBackgroundVisibilityRange members 1330
IsOn 1327	ToPos 1330
Name 1327	Type 1330
Number 1328	IRobotBackgroundVisibilityRangeType 1329
Save 1328	IRobotBar 618
VisibilityRange 1328	about IRobotBar 618
IRobotBackgroundInsertParams 1321	AnalyzeTTMethod 621
about IRobotBackgroundInsertParams 1321	CalcGamma 628
InsertionPoint 1323	ChangeOrientation 628
IRobotBackgroundInsertParams fields 1322	Elements 621
IRobotBackgroundInsertParams members 1322	End 622
Layers 1323	EndNode 622
Mirror 1323	Gamma 622
Plane 1324	GetElemsData 628
PositionOnNormalAxis 1324	GetLCS 629
ReferencePoint 1324	GetSimpleBars 629
RotationAngle 1324	InactiveBar 623
ScalingFactor 1325	IRobotBar fields 620
Units 1325	IRobotBar members 619
IRobotBackgroundLayers 1319	IRobotBar methods 627
about IRobotBackgroundLayers 1319	IsSuperBar 623
Count 1320	Length 623
FindName 1320	Name 623
Get 1321	NameTemplate 624
IRobotBackgroundLayers fields 1320	ReversedOffset 624

ReversedRelease 624	PosUX 891
ReversedSection 625	PosUY 891
SetOffset 629	PosUZ 891
SetOrientation 630	UX 891
SetSection 630	UY 891
ShearForces 625	UZ 892
Start 625	IRobotBarDeflectionServer 870
StartNode 626	about IRobotBarDeflectionServer 870
StructuralType 626	DynComb.MaxValue 872
TensionCompression 626	DynCombValue 872
TrussBar 627	Dyn.MaxValue 872
Uniqueld 627	DynValue 872
IRobotBarBucklingData 892	IRobotBarDeflectionServer members 871
about IRobotBarBucklingData 892	IRobotBarDeflectionServer methods 871
BuckLengthY 893	MaxValue 873
BuckLengthZ 893	MaxValueEx 873
CriticalCoef 893	Value 873
CriticalForce 894	ValueEx 874
IRobotBarBucklingData fields 892	IRobotBarDilatationRecordValues 40
IRobotBarBucklingData members 892	IRobotBarDisplacementServer 877
SlendY 894	about IRobotBarDisplacementServer 877
SlendZ 894	DynCombValue 878
IRobotBarBucklingServer 876	DynValue 878
about IRobotBarBucklingServer 876	IRobotBarDisplacementServer members 878
CriticalCoef 877	IRobotBarDisplacementServer methods 878
EigenValue 877	Value 879
IRobotBarBucklingServer members 876	IRobotBarElasticGroundData 538
IRobotBarBucklingServer methods 876	about IRobotBarElasticGroundData 538
IRobotBarCableAssemblingParamType 537	GetOneDir 540
IRobotBarCableData 535	HX 538
about IRobotBarCableData 535	IRobotBarElasticGroundData fields 538
AssemblingParam 535	IRobotBarElasticGroundData members 538
AssemblingParamValue 536	IRobotBarElasticGroundData methods 539
IRobotBarCableData fields 535	KY 539
IRobotBarCableData members 535	KZ 539
MaterialName 536	SetOneDir 540
SectionAX 536	IRobotBarElement 646
IRobotBarDeadRecordValues 43	about IRobotBarElement 646
IRobotBarDeflectionData 890	EndNode 647
about IRobotBarDeflectionData 890	Inactive 647
IRobotBarDeflectionData fields 890	IRobotBarElement fields 647
IRobotBarDeflectionData members 890	IRobotBarElement members 647

IsCalc 648	IsValueRelative 617
Number 648	SetValue 617
StartNode 648	Type 616
IRobotBarElementData 649	IRobotBarEndBracketDataValue 615
about IRobotBarElementData 649	IRobotBarEndBracketType 615
Bar 649	IRobotBarEndOffsetData 613
GetEndNode 650	about IRobotBarEndOffsetData 613
GetStartNode 651	IRobotBarEndOffsetData fields 614
IRobotBarElementData fields 649	IRobotBarEndOffsetData members 614
IRobotBarElementData members 649	MemberLength 614
IRobotBarElementData methods 650	UX 614
Number 650	UY 615
IRobotBarElementDataSet 651	UZ 615
about IRobotBarElementDataSet 651	IRobotBarEndReleaseData 577
BarCount 652	about IRobotBarEndReleaseData 577
ElemCount 652	AX 578
GetBarNumber 653	AY 579
GetElem 653	AZ 579
GetElemCountForBar 653	BX 579
GetElemForBar 654	BY 580
IRobotBarElementDataSet fields 652	BZ 580
IRobotBarElementDataSet members 651	HX 580
IRobotBarElementDataSet methods 652	HY 580
IRobotBarEnd 642	HZ 581
about IRobotBarEnd 642	IRobotBarEndReleaseData fields 578
GetLabel 644	IRobotBarEndReleaseData members 577
GetLabelName 644	KX 581
GetLabels 644	KY 581
GetOffsetValue 645	KZ 581
HasLabel 645	NonlinearModel 582
IRobotBarEnd fields 643	RX 582
IRobotBarEnd members 643	RY 582
IRobotBarEnd methods 644	RZ 582
Node 643	UX 583
RemoveLabel 645	UY 583
SetLabel 646	UZ 583
IRobotBarEndBracketData 615	IRobotBarEndReleaseValue 575
about IRobotBarEndBracketData 615	IRobotBarForceConcentrateRecordValues 41
GetValue 617	IRobotBarForceData 884
IRobotBarEndBracketData fields 616	about IRobotBarForceData 884
IRobotBarEndBracketData members 616	FX 885
IRobotBarEndBracketData methods 617	FY 885

FZ 885	IRobotBarOffsetData fields 612
IRobotBarForceData fields 884	IRobotBarOffsetData members 611
IRobotBarForceData members 884	ObjectNumber 613
KY 885	Position 613
KYAvailable 885	Start 613
KZ 886	IRobotBarOffsetMemberLength 618
KZAvailable 886	IRobotBarReleaseData 576
MX 886	about IRobotBarReleaseData 576
MY 886	EndNode 576
MZ 887	IRobotBarReleaseData fields 576
IRobotBarForceServer 867	IRobotBarReleaseData members 576
about IRobotBarForceServer 867	StartNode 577
DynCombValue 868	IRobotBarResultServer 864
DynCombValueByNPoints 868	about IRobotBarResultServer 864
DynValue 869	Buckling 865
DynValueByNPoints 869	Deflections 865
IRobotBarForceServer members 867	Displacements 866
IRobotBarForceServer methods 868	Forces 866
Value 869	Geolimperfections 867
ValueByNPoints 870	IRobotBarResultServer fields 865
ValueByNPointsEx 870	IRobotBarResultServer members 864
ValueEx 870	IRobotBarResultServer methods 866
IRobotBarGeolimperfectionsAxis 587	Stresses 866
IRobotBarGeolimperfectionsData 583	IRobotBarSectionComplexData 568
about IRobotBarGeolimperfectionsData 583	about IRobotBarSectionComplexData 568
GetBucklingCoeff 585	B 569
GetUser 585	Count 569
IRobotBarGeolimperfectionsData members 584	D 569
IRobotBarGeolimperfectionsData methods 584	Get 570
IsAutomatic 585	GetShape 570
IsMinus 586	GetValue 570
SetAutomatic 586	IRobotBarSectionComplexData fields 568
SetMinus 586	IRobotBarSectionComplexData members 568
SetUser 587	IRobotBarSectionComplexData methods 569
IRobotBarIntersectRelationship 2209	Set 571
IRobotBarMomentDistributedRecordValues 41	IRobotBarSectionComponentShape 571
IRobotBarOffsetAutoPosition 618	IRobotBarSectionConcreteCutsPosition 547
IRobotBarOffsetData 611	IRobotBarSectionConcreteData 560
about IRobotBarOffsetData 611	about IRobotBarSectionConcreteData 560
AxisOffset 612	BeamCutsPosition 561
CoordinateSystem 612	CalcGeometry 562
End 612	GetReduction 563

GetTapered 563	IRobotBarSectionDataValue 544
GetValue 563	IRobotBarSectionElasticParams 565
IRobotBarSectionConcreteData fields 561	about IRobotBarSectionElasticParams 565
IRobotBarSectionConcreteData members 560	Active 566
IRobotBarSectionConcreteData methods 562	IRobotBarSectionElasticParams fields 565
IsBeam 562	IRobotBarSectionElasticParams members 565
IsColumn 562	L1 566
SetReduction 564	L2 566
SetTapered 564	MaterialModel 566
SetValue 564	N 567
IRobotBarSectionConcreteDataValue 546	N1 567
IRobotBarSectionData 548	N2 567
about IRobotBarSectionData 548	IRobotBarSectionNonstdData 559
AliasCount 550	about IRobotBarSectionNonstdData 559
CalcNonstdGeometry 555	GetValue 560
Concrete 550	IRobotBarSectionNonstdData fields 559
CreateNonstd 555	IRobotBarSectionNonstdData members 559
DrawSymbol 555	IRobotBarSectionNonstdData methods 559
ElasticParams 551	Position 559
FindAlias 556	SetValue 560
FindNonstd 556	IRobotBarSectionNonstdDataValue 546
GetAlias 556	IRobotBarSectionQuantitySurvey 1077
GetAliasEx 556	about IRobotBarSectionQuantitySurvey 1077
GetNonstd 557	Count 1078
GetValue 557	GetLength 1078
IRobotBarSectionData fields 549	GetName 1078
IRobotBarSectionData members 548	GetPaintArea 1079
IRobotBarSectionData methods 554	GetUnitWeight 1079
IsConcrete 551	GetVolume 1079
IsJoist 551	GetWeight 1080
IsSpecial 551	IRobotBarSectionQuantitySurvey fields 1077
LoadFromDBase 557	IRobotBarSectionQuantitySurvey members 1077
LoadFromDBase2 558	IRobotBarSectionQuantitySurvey methods 1078
MaterialName 552	IRobotBarSectionShapeType 542
Members 552	IRobotBarSectionSpecialData 572
Name 552	about IRobotBarSectionSpecialData 572
NonstdCount 553	DbFullName 572
RemoveNonstd 558	DbName 573
SetValue 558	GetValue 574
ShapeType 553	IRobotBarSectionSpecialData fields 572
Special 553	IRobotBarSectionSpecialData members 572
Type 554	IRobotBarSectionSpecialData methods 573

Section 573	IRobotBarStressData members 887
SetValue 574	ShearY 888
IRobotBarSectionSpecialDataValue 574	ShearZ 888
IRobotBarSectionType 545	Smax 888
IRobotBarServer 630	SmaxMY 889
about IRobotBarServer 630	SmaxMZ 889
BeginMultiOperation 634	Smin 889
CalcGamma 635	SminMY 889
ChangeOrientation 635	SminMZ 889
Create 635	Torsion 889
CreateSuperBar 636	IRobotBarStressServer 874
EndMultiOperation 636	about IRobotBarStressServer 874
FindWithId 636	DynCombValue 875
FreeNumber 632	DynValue 875
GetAnalyzeTTMethodEnabled 636	IRobotBarStressServer members 874
GetElemsData 637	IRobotBarStressServer methods 874
GetLCS 637	Value 875
GetName 637	ValueEx 876
GetNameTemplate 638	IRobotBarTensionCompression 646
GetStructuralType 638	IRobotBarThermalRecordValues 43
GetUniqueId 638	IRobotBarTrapezoidalRecordValues 42
IRobotBarServer fields 632	IRobotBarUniformRecordValues 41
IRobotBarServer members 630	IRobotBucklingAnalysisMethod 360
IRobotBarServer methods 633	IRobotBucklingAnalysisParams 356
IsInactive 639	about IRobotBucklingAnalysisParams 356
IsTrussBar 639	Increment 357
NonlinearHingeModels 633	IRobotBucklingAnalysisParams fields 357
NonlinearHinges 633	IRobotBucklingAnalysisParams members 356
SetAnalyzeTTMethod 639	IsNonlinear 358
SetInactive 640	IterationsCount 358
SetLabelExt 640	Method 358
SetNameTemplate 640	ModesCount 358
SetOrientation 641	NonlinearParams 359
SetShearForces 641	Shift 359
SetStructuralType 641	SturmVerification 359
SetTensionCompression 642	Tolerance 360
SetTrussBar 642	IRobotBucklingDeformationParams 1581
Update 642	about IRobotBucklingDeformationParams 1581
IRobotBarStressData 887	CaseNumber 1582
about IRobotBarStressData 887	GetModeCoeff 1583
FXSX 888	IRobotBucklingDeformationParams fields 1582
IRobotBarStressData fields 887	IRobotBucklingDeformationParams members 1581

IRobotBucklingDeformationParams methods 1583	IRobotCalculationResume methods 1030
MaxDisplacement 1582	IRobotCalculationStatus 1586
OmitCaseForDeformations 1582	IRobotCalculationsType 1540
SetModeCoeff 1583	IRobotCase 404
IRobotCalcEngine 1553	about IRobotCase 404
about IRobotCalcEngine 1553	AnalizeType 405
AnalysisParams 1555	IRobotCase fields 405
AutoFreezeResults 1555	IRobotCase members 404
AutoGenerateModel 1555	Name 405
BucklingDeformation 1556	Nature 405
Calculate 1558	Number 406
CalculateEx 1558	Type 406
DAM 1556	IRobotCaseAnalizeType 407
GenerateModel 1559	IRobotCaseAnalysisModesFilter 432
GenerationParams 1556	about IRobotCaseAnalysisModesFilter 432
IRobotCalcEngine fields 1554	IRobotCaseAnalysisModesFilter fields 432
IRobotCalcEngine members 1553	IRobotCaseAnalysisModesFilter members 432
IRobotCalcEngine methods 1558	MassPercentage 432
SaveResultsInExternalFile 1556	Modes 433
SeismicResultsSaveParams 1557	Type 433
StatusWindowParent 1557	IRobotCaseAnalysisModesFilterType 433
StopCalculation 1559	IRobotCaseCollection 431
UseStatusWindow 1557	about IRobotCaseCollection 431
IRobotCalcEngineEvents 1584	IRobotCaseCombination 422
about IRobotCalcEngineEvents 1584	about IRobotCaseCombination 422
CalcMessage 1584	CaseFactors 423
IRobotCalcEngineEvents members 1584	CombinationType 423
IRobotCalcEngineEvents methods 1584	GetAnalysisParams 426
IRobotCalcMessageSeverityLevel 1585	IRobotCaseCombination fields 423
IRobotCalcMessageSource 1585	IRobotCaseCombination members 422
IRobotCalculationMode 1586	IRobotCaseCombination methods 426
IRobotCalculationModelCoherence 1318	IsAuxiliary 424
IRobotCalculationResume 1028	Label 424
about IRobotCalculationResume 1028	NatureName 424
DiagonalStiffnessMatrixMax 1029	Quadratic 425
DiagonalStiffnessMatrixMin 1029	SeismicType 425
DiagonalStiffnessMatrixPrecision 1029	SetAnalysisParams 426
EquationSolvingMethodUsed 1030	SetNatureExt 427
GetEnergy 1030	Signed 425
GetEnergyPrecision 1031	UniqueID 425
IRobotCalculationResume fields 1029	IRobotCaseFactor 430
IRobotCalculationResume members 1028	about IRobotCaseFactor 430

CaseNumber	430	SnowWindEngine	410
Factor	431	TimeHistoryFunctions	410
IRobotCaseFactor	fields 430	WindLoadsSimulationEngine	411
IRobotCaseFactor	members 430		
IRobotCaseFactorMngr	427	IRobotCaseType	429
about IRobotCaseFactorMngr	427	IRobotCladdingData	609
Count	428	about IRobotCladdingData	609
Delete	428	IRobotCladdingData	fields 609
Get	428	IRobotCladdingData	members 609
IRobotCaseFactorMngr	fields 428	Method	610
IRobotCaseFactorMngr	members 427	Type	610
IRobotCaseFactorMngr	methods 428	IRobotCladdingMethod	611
New	429	IRobotCladdingType	610
IRobotCaseNature	406	IRobotCmdInfo	1547
IRobotCasePredefinedNumber	474	about IRobotCmdInfo	1547
IRobotCaseRelatedValueType	442	Id	1547
IRobotCaseServer	407	IRobotCmdInfo	fields 1547
about IRobotCaseServer	407	IRobotCmdInfo	members 1547
BeginMultiOperation	412	MenuChecked	1548
CodeCmbEngine	409	MenuEnabled	1548
CreateCombination	412	Name	1548
CreateMobile	412	IRobotCmdList	1545
CreateSimple	413	about IRobotCmdList	1545
Delete	413	Count	1546
DeleteMany	413	Get	1546
EndMultiOperation	414	IRobotCmdList	fields 1545
Exist	414	IRobotCmdList	members 1545
FindWithId	414	IRobotCmdList	methods 1546
FreeNumber	409	New	1546
Get	414	IRobotCodeCmbActionCoeffType	162
GetAll	415	IRobotCodeCmbActionServer	158
GetCmpntCount	415	about IRobotCodeCmbActionServer	158
GetMany	415	Count	159
GetRelatedValue	416	GetCoeff	160
GetUniqueId	416	GetName	160
IRobotCaseServer	fields 409	GetNature	160
IRobotCaseServer	members 408	IRobotCodeCmbActionServer	fields 159
IRobotCaseServer	methods 411	IRobotCodeCmbActionServer	members 158
IsAuxiliary	416	IRobotCodeCmbActionServer	methods 159
QCmbTau	410	New	161
SELFSeismicEngine	410	Remove	161
SetAuxiliary	417	SetCoeff	161
		SetName	161

SetNature 162	IRobotCodeCmbFactor members 153
IRobotCodeCmbActiveCaseInfo 184	IRobotCodeCmbFlag 155
about IRobotCodeCmbActiveCaseInfo 184	IRobotCodeCmbGenerationParams 166
CaseNature 185	about IRobotCodeCmbGenerationParams 166
CaseNumber 185	ActiveCases 168
Coefficient 186	AllBars 168
GroupNumber 186	AllNodes 168
IRobotCodeCmbActiveCaseInfo fields 185	BarSel 168
IRobotCodeCmbActiveCaseInfo members 185	ExtremalSnowFactor 169
IsSelected 186	GenType 169
IRobotCodeCmbCombPartType 165	Groups 169
IRobotCodeCmbCombs 163	IRobotCodeCmbGenerationParams fields 167
about IRobotCodeCmbCombs 163	IRobotCodeCmbGenerationParams members 166
Count 163	IRobotCodeCmbGenerationParams methods 170
Get 164	IsCombinationSelected 171
IRobotCodeCmbCombs fields 163	IsCombinationTypeSelected 171
IRobotCodeCmbCombs members 163	IsDecidingValueSelected 171
IRobotCodeCmbCombs methods 164	NodeSel 169
New 164	PointsOnBar 170
Remove 165	Regulations 170
Set 165	Relations 170
IRobotCodeCmbCombType 166	SelectCombination 172
IRobotCodeCmbComponent 154	SelectCombinationType 172
about IRobotCodeCmbComponent 154	SelectDecidingValue 172
Count 155	IRobotCodeCmbGenerationType 186
Get 155	IRobotCodeCmbGroup 173
IRobotCodeCmbComponent fields 154	about IRobotCodeCmbGroup 173
IRobotCodeCmbComponent members 154	Add 175
IRobotCodeCmbComponent methods 155	AddAll 176
IRobotCodeCmbComponentMngr 152	CaseCount 174
about IRobotCodeCmbComponentMngr 152	IRobotCodeCmbGroup fields 174
Count 152	IRobotCodeCmbGroup members 173
Get 153	IRobotCodeCmbGroup methods 175
IRobotCodeCmbComponentMngr fields 152	IsFull 176
IRobotCodeCmbComponentMngr members 152	Nature 174
IRobotCodeCmbComponentMngr methods 153	Operator 175
IRobotCodeCmbDecidingValueType 173	IRobotCodeCmbGroupRelation 179
IRobotCodeCmbFactor 153	about IRobotCodeCmbGroupRelation 179
about IRobotCodeCmbFactor 153	AddGroup 181
CaseNumber 154	Clear 181
Factor 154	GetGroup 181
IRobotCodeCmbFactor fields 153	GetGroupCount 182

GetOperator 182	IRobotCodeCombination members 150
IRobotCodeCmbGroupRelation fields 179	IRobotCodeCombination methods 151
IRobotCodeCmbGroupRelation members 179	Uniqueld 151
IRobotCodeCmbGroupRelation methods 180	IRobotCodeCombinationEngine 148
Nature 180	about IRobotCodeCombinationEngine 148
NewRow 182	Generate 149
RemoveRow 182	IRobotCodeCombinationEngine fields 148
RowCount 180	IRobotCodeCombinationEngine members 148
SetGroup 183	IRobotCodeCombinationEngine methods 149
SetOperator 183	Params 149
IRobotCodeCmbGroupRelationServer 183	IRobotCodeRegistrar 1637
about IRobotCodeCmbGroupRelationServer 183	about IRobotCodeRegistrar 1637
Get 184	CodeName 1639
IRobotCodeCmbGroupRelationServer members 183	CodeType 1639
IRobotCodeCmbGroupRelationServer methods 184	Country 1639
Set 184	Guid 1640
IRobotCodeCmbGroupServer 176	IRobotCodeRegistrar fields 1638
about IRobotCodeCmbGroupServer 176	IRobotCodeRegistrar members 1638
FindByCase 177	IRobotCodeRegistrar methods 1640
Get 178	Manufacturer 1640
GetCount 178	ProgId 1640
IRobotCodeCmbGroupServer members 177	Register 1641
IRobotCodeCmbGroupServer methods 177	Unregister 1641
New 178	IRobotCodeType 1303
IRobotCodeCmbOperator 176	IRobotCollection 9
IRobotCodeCmbRegulations 156	about IRobotCollection 9
Actions 157	Count 10
CodeName 157	Get 10
Combinations 157	IRobotCollection fields 10
IRobotCodeCmbRegulations fields 156	IRobotCollection members 9
IRobotCodeCmbRegulations members 156	IRobotCollection methods 10
IsEuroCode 157	IRobotCombinationType 429
MaterialType 158	IRobotComponentFactory 1645
Version 158	about IRobotComponentFactory 1645
IRobotCodeCombination 149	Create 1646
about IRobotCodeCombination 149	CreateExt 1646
CombinationType 150	IRobotComponentFactory members 1646
Components 150	IRobotComponentFactory methods 1646
FindByFlag 151	IRobotComponentType 1647
Flag 151	IRobotDAMAnalysisType 1589
IRobotCodeCombination fields 150	IRobotDAMCalcModule 1587
	about IRobotDAMCalcModule 1587

DeleteModel 1589	IRobotDatabaseType 1317
IRobotDAMCalcModule fields 1588	IRobotDataObject 29
IRobotDAMCalcModule members 1588	about IRobotDataObject 29
IRobotDAMCalcModule methods 1589	GetLabel 31
IsActive 1588	GetLabelName 31
Params 1588	GetLabels 31
Run 1589	HasLabel 32
IRobotDAMLateralLoadCombType 1594	IRobotDataObject fields 30
IRobotDAMNotionalLoads 1590	IRobotDataObject members 30
about IRobotDAMNotionalLoads 1590	IRobotDataObject methods 30
ActiveXN 1591	Number 30
ActiveXP 1591	RemoveLabel 32
ActiveYN 1592	SetLabel 32
ActiveYP 1592	IRobotDataObjectServer 33
Coefficient 1592	about IRobotDataObjectServer 33
GravityLoadCombEnabled 1593	Delete 34
IRobotDAMNotionalLoads fields 1591	DeleteMany 34
IRobotDAMNotionalLoads members 1590	Exist 34
LateralLoadCombEnabled 1593	Get 34
LateralLoadCombType 1593	GetAll 35
IRobotDAMParams 1597	GetMany 35
about IRobotDAMParams 1597	IRobotDataObjectServer members 33
Analysis 1598	IRobotDataObjectServer methods 33
GetNLDPParams 1600	RemoveLabel 35
IRobotDAMParams fields 1598	SetLabel 36
IRobotDAMParams members 1598	IRobotDeadRecordValues 51
IRobotDAMParams methods 1599	IRobotDegreeOfFreedom 1648
NotionalLoads 1599	IRobotDialogId 1469
ReducedStiffness 1599	IRobotDirectory 1536
SetNLDPParams 1600	IRobotDirectoryExtension 1538
Update 1600	IRobotDisplacementData 882
IRobotDAMReducedStiffness 1594	about IRobotDisplacementData 882
about IRobotDAMReducedStiffness 1594	IRobotDisplacementData fields 882
IRobotDAMReducedStiffness fields 1595	IRobotDisplacementData members 882
IRobotDAMReducedStiffness members 1594	RX 882
RCBeamsValue 1595	RY 883
RCColumnsAndWallsValue 1595	RZ 883
RCSlabsValue 1596	UX 883
SteelMembersReductionType 1596	UY 883
SteelMembersTauBValue 1596	UZ 883
SteelMembersValue 1597	IRobotDynamicAnalysisDamping 439
IRobotDAMSteelMembersReductionType 1597	about IRobotDynamicAnalysisDamping 439

ConstValue 440	Period 897
GetModeDamping 441	Precision 897
GetModes 441	Pulsation 897
IRobotDynamicAnalysisDamping fields 440	IRobotEigenvaluesServer 922
IRobotDynamicAnalysisDamping members 439	about IRobotEigenvaluesServer 922
IRobotDynamicAnalysisDamping methods 440	CombValue 923
RemoveModeDamping 441	IRobotEigenvaluesServer members 922
SetModeDamping 442	IRobotEigenvaluesServer methods 922
Type 440	Value 923
IRobotDynamicAnalysisDampingType 442	IRobotEigenvectorsServer 927
IRobotDynamicAnalysisExcitationDirection 434	about IRobotEigenvectorsServer 927
about IRobotDynamicAnalysisExcitationDirection 434	CombValue 928
CombType 435	IRobotEigenvectorsServer members 927
Group1 435	IRobotEigenvectorsServer methods 927
Group2 436	Value 928
Group3 436	IRobotEmitter 511
IRobotDynamicAnalysisExcitationDirection fields 435	about IRobotEmitter 511
IRobotDynamicAnalysisExcitationDirection members 434	EstimatedElemNumber 511
Lambda 436	H0 512
Mi 436	IRobotEmitter fields 511
NormalizedX 436	IRobotEmitter members 511
NormalizedY 437	R1 512
NormalizedZ 437	R2 512
QuadraticActive 437	VariableMeshDensityIncrement 512
QuadraticSigned 437	IRobotEquationSolvingMethod 1559
ResolutionActive 437	IRobotEurocodeFactors 1316
Rx 438	about IRobotEurocodeFactors 1316
Ry 438	IRobotEurocodeFactors fields 1317
Rz 438	IRobotEurocodeFactors members 1316
UseNormalized 438	SteelConnections 1317
X 438	SteelDesign 1317
Y 439	IRobotEurocodeSteelConnectionFactors 1311
Z 439	about IRobotEurocodeSteelConnectionFactors 1311
IRobotEigenvalues 894	Gamma0 1312
about IRobotEigenvalues 894	Gamma1 1312
AvPartCoeff 895	Gamma2 1313
Damping 896	Gamma3 1313
EigenValue 896	Gamma3Ser 1313
Energy 896	Gamma4 1314
Frequence 896	Gamma5 1314
IRobotEigenvalues fields 895	Gamma6 1314
IRobotEigenvalues members 895	Gamma7 1315

GammaC 1315	Position 908
IRobotEurocodeSteelConnectionFactors fields 1312	Value 908
IRobotEurocodeSteelConnectionFactors members 1311	IRobotExtremeValueType 910
IRobotEurocodeSteelConnectionFactors methods 1315	IRobotFeExtremeParams 981
LoadFromCode 1315	about IRobotFeExtremeParams 981
LoadFromCodeNumber 1316	CaseCmpnt 983
IRobotEurocodeSteelDesignFactors 1308	CaseSel 983
about IRobotEurocodeSteelDesignFactors 1308	ElementSel 984
Gamma0 1309	GetDirX 987
Gamma1 1309	IRobotFeExtremeParams fields 983
Gamma2 1309	IRobotFeExtremeParams members 982
GammaFire 1310	IRobotFeExtremeParams methods 986
IRobotEurocodeSteelDesignFactors fields 1308	Layer 984
IRobotEurocodeSteelDesignFactors members 1308	LayerArbitraryValue 984
IRobotEurocodeSteelDesignFactors methods 1310	ModeCmb 985
LoadFromCode 1310	NodeSel 985
LoadFromCodeNumber 1310	PanelSel 985
IRobotExternalFileFormat 1266	ReducedCutPos 986
IRobotExternalPreviewFormat 1510	Reset 987
IRobotExtremeParams 908	ResultId 986
about IRobotExtremeParams 908	SetDirX 987
BarDivision 909	Smoothing 986
IRobotExtremeParams fields 909	IRobotFeExtremeValue 988
IRobotExtremeParams members 909	about IRobotFeExtremeValue 988
Selection 909	Case 989
ValueType 910	CaseCmpnt 990
IRobotExtremeResultServer 1012	Element 990
about IRobotExtremeResultServer 1012	GetDirX 993
IRobotExtremeResultServer members 1012	IRobotFeExtremeValue fields 989
IRobotExtremeResultServer methods 1012	IRobotFeExtremeValue members 988
.MaxValue 1012	IRobotFeExtremeValue methods 993
.MinValue 1013	IsAvailable 990
IRobotExtremeValue 905	Layer 990
about IRobotExtremeValue 905	LayerArbitraryValue 991
Bar 906	ModeCmb 991
Case 906	Node 991
CaseCmpnt 906	Panel 991
IRobotExtremeValue fields 905	ReducedCutPos 992
IRobotExtremeValue members 905	ResultId 992
IsAvailable 907	Smoothing 992
ModeCmb 907	Value 992
Node 907	IRobotFeLayerType 932

IRobotFeMultiExtremeValue 995	RXX 942
about IRobotFeMultiExtremeValue 995	RYY 943
Count 996	SXX 943
Get 996	SXY 943
GetByResType 997	SYY 943
IRobotFeMultiExtremeValue fields 996	TXX 944
IRobotFeMultiExtremeValue members 996	TYY 944
IRobotFeMultiExtremeValue methods 996	UXX 944
IRobotFeMultiResultType 993	UYY 945
about IRobotFeMultiResultType 993	WNorm 945
Add 994	IRobotFeResultKind 981
Count 994	IRobotFeResultParams 932
Get 995	about IRobotFeResultParams 932
IRobotFeMultiResultType fields 994	CalcMethod 934
IRobotFeMultiResultType members 994	Case 934
IRobotFeMultiResultType methods 994	CaseCmpnt 934
Remove 995	Element 935
IRobotFeResultComplex 958	GetDirX 937
about IRobotFeResultComplex 958	IRobotFeResultParams fields 933
IRobotFeResultComplex fields 959	IRobotFeResultParams members 933
IRobotFeResultComplex members 958	IRobotFeResultParams methods 937
M_MISES 959	Layer 935
MXX_BOTTOM 959	LayerArbitraryValue 935
MXX_TOP 960	ModeCmb 936
MYY_BOTTOM 960	Node 936
MYY_TOP 960	Panel 936
N_MISES 961	SetDirX 937
S_MISES 961	IRobotFeResultPrincipal 951
IRobotFeResultDetailed 938	about IRobotFeResultPrincipal 951
about IRobotFeResultDetailed 938	IRobotFeResultPrincipal fields 952
IRobotFeResultDetailed fields 939	IRobotFeResultPrincipal members 952
IRobotFeResultDetailed members 938	M1 953
MXX 939	M1_2 953
MXY 940	M2 953
MYY 940	MAL 953
NX 940	N1 954
NY 940	N1_2 954
NY 941	N2 954
PNorm 941	NAL 955
QXX 941	Q1_2 955
QYY 942	S1 955
RNorm 942	S1_2 956

S2 956	IRobotFeResultReinforcement fields 962
SAL 956	IRobotFeResultReinforcement members 961
T1_2 956	IRobotFeResultServer 945
U 957	about IRobotFeResultServer 945
UGX 957	Complex 947
UGY 957	Detailed 948
UGZ 958	IRobotFeResultServer fields 946
IRobotFeResultReduced 966	IRobotFeResultServer members 946
about IRobotFeResultReduced 966	IRobotFeResultServer methods 947
CutPos 968	MaxValue 948
Height 968	MinValue 948
IRobotFeResultReduced fields 967	MultiMaxValue 949
IRobotFeResultReduced members 967	MultiMinValue 949
Length 968	PanelCuts 946
MY 969	Principal 949
MZ 969	Reduced 950
NodeLeftBottom 969	ReducedEx 950
NodeLeftTop 969	Reinforcement 950
NodeRightBottom 970	IRobotFeResultSmoothing 980
NodeRightTop 970	IRobotFeResultType 979
NX 970	IRobotFileInsertParams 1346
SE 970	about IRobotFileInsertParams 1346
SO 971	AsObject 1347
T 971	GetInsertionPoint 1348
TY 971	GetRotation 1348
TZ 972	IRobotFileInsertParams fields 1347
IRobotFeResultReducedCutPosition 972	IRobotFileInsertParams members 1346
IRobotFeResultReinforcement 961	IRobotFileInsertParams methods 1348
A_MIN 962	ReferenceNode 1347
about IRobotFeResultReinforcement 961	ScaleFactor 1347
AX 963	SetInsertionPoint 1348
AX_BOTTOM 963	SetRotation 1349
AX_TOP 963	IRobotFiniteElement 828
AY 964	about IRobotFiniteElement 828
AY_BOTTOM 964	CalcArea 830
AY_TOP 964	FeType 829
CalcError 965	IRobotFiniteElement fields 829
E_AX_BOTTOM 965	IRobotFiniteElement members 828
E_AX_TOP 965	IRobotFiniteElement methods 830
E_AY_BOTTOM 965	Nodes 829
E_AY_TOP 966	ObjectNumber 829
F 966	ObjectPartIdx 830

Uniquid 830	IRobotFiniteElementType 831
IRobotFiniteElementData 837	IRobotFootfallAnalysisExcitationForces 395
about IRobotFiniteElementData 837	IRobotFootfallAnalysisExcitationMethod 395
GetNode 839	IRobotFootfallAnalysisModalParams 398
IRobotFiniteElementData fields 838	about IRobotFootfallAnalysisModalParams 398
IRobotFiniteElementData members 838	FrequencyLimit 398
IRobotFiniteElementData methods 839	IgnoreDensity 399
NodeCount 838	IncludeMassForDirX 399
Number 839	IncludeMassForDirY 399
Object 839	IncludeMassForDirZ 400
IRobotFiniteElementDataSet 840	IRobotFootfallAnalysisModalParams fields 398
about IRobotFiniteElementDataSet 840	IRobotFootfallAnalysisModalParams members 398
ElemCount 841	IRobotFootfallAnalysisNodeSelection 396
GetElem 842	about IRobotFootfallAnalysisNodeSelection 396
GetElemCountForObject 842	IRobotFootfallAnalysisNodeSelection fields 396
GetElemForObject 842	IRobotFootfallAnalysisNodeSelection members 396
GetObjectNumber 843	SelectedNodes 396
IRobotFiniteElementDataSet fields 840	SelectedPanels 397
IRobotFiniteElementDataSet members 840	Type 397
IRobotFiniteElementDataSet methods 841	IRobotFootfallAnalysisNodeSelectionType 397
ObjectCount 841	IRobotFootfallAnalysisParams 400
IRobotFiniteElementNodes 831	about IRobotFootfallAnalysisParams 400
about IRobotFiniteElementNodes 831	Damping 401
Count 832	ExcitationForces 401
Get 833	ExcitationMethod 402
GetAll 833	ExcitationNodes 402
IRobotFiniteElementNodes fields 832	FootstepsNumber 402
IRobotFiniteElementNodes members 832	IRobotFootfallAnalysisParams fields 401
IRobotFiniteElementNodes methods 833	IRobotFootfallAnalysisParams members 400
Set 833	MaxWalkingFrequency 402
SetAll 834	MinWalkingFrequency 403
IRobotFiniteElementServer 834	ModalParams 403
about IRobotFiniteElementServer 834	ResponseNodes 403
CalcArea 836	WalkersWeight 404
Create 836	IRobotFootfallResults 915
FreeNumber 835	A 916
IRobotFiniteElementServer fields 835	about IRobotFootfallResults 915
IRobotFiniteElementServer members 834	ExcitationNode 916
IRobotFiniteElementServer methods 836	Frequency 917
MeshConcentrate 836	IRobotFootfallResults fields 916
MeshConsolidate 837	IRobotFootfallResults members 915
Update 837	RF_Overall 917

RF_Resonant	917	P3	1621
RF_Transient	918	IRobotGeoArcDefinitionMethod	1626
VRMQ	918	IRobotGeoCircle	1621
VRMS	918	about IRobotGeoCircle	1621
IRobotForcesData	897	IRobotGeoCircle fields	1622
		IRobotGeoCircle members	1622
FX	898	P1	1622
FY	898	P2	1622
FZ	899	P3	1623
IRobotForcesData	fields 898	PC	1623
IRobotForcesData	members 898	IRobotGeoContour	1618
MX	899	about IRobotGeoContour	1618
MY	899	Add	1619
MZ	899	Clear	1619
IRobotFRFAnalysisParams	392	IRobotGeoContour fields	1619
		IRobotGeoContour members	1618
Damping	393	IRobotGeoContour methods	1619
FinalFrequency	393	Segments	1619
FrequencyDivision	394	IRobotGeoCoordinateAxis	1606
IncludeEigenfrequencies	394	IRobotGeoCoordinateAxisSense	1626
InitialFrequency	394	IRobotGeoCoordinateSystem	1607
IRobotFRFAnalysisParams	fields 393	IRobotGeoCurveDiv	1608
IRobotFRFAnalysisParams	members 393	about IRobotGeoCurveDiv	1608
IRobotFRFResults	913	IRobotGeoCurveDiv fields	1609
		IRobotGeoCurveDiv members	1608
Frequency	914	IsAnalytical	1609
IRobotFRFResults	fields 914	Length	1609
IRobotFRFResults	members 913	Mode	1609
VX	914	N	1610
VY	914	IRobotGeoLayer	1624
VZ	915	about IRobotGeoLayer	1624
IRobotFRFResultServer	929	IRobotGeoLayer fields	1624
		IRobotGeoLayer members	1624
IRobotFRFResultServer	members 929	IsThickDefined	1624
IRobotFRFResultServer	methods 930	P1	1625
Value	930	P2	1625
IRobotGeoArc	1620	P3	1625
		PDir	1625
IRobotGeoArc	fields 1620	Thickness	1626
IRobotGeoArc	members 1620	IRobotGeoObject	1615
P1	1621	about IRobotGeoObject	1615
P2	1621	Initialize	1616

IRobotGeoObject fields 1615	about IRobotGeoSegmentArc 1613
IRobotGeoObject members 1615	Div 1614
IRobotGeoObject methods 1616	IRobotGeoSegmentArc fields 1613
Type 1615	IRobotGeoSegmentArc members 1613
IRobotGeoObjectType 1608	IRobotGeoSegmentArc methods 1614
IRobotGeoPlane 1627	P2 1614
IRobotGeoPoint2D 1602	Set 1614
about IRobotGeoPoint2D 1602	IRobotGeoSegmentCollection 1618
Get 1604	about IRobotGeoSegmentCollection 1618
IRobotGeoPoint2D fields 1603	IRobotGeoSegmentLine 1611
IRobotGeoPoint2D members 1602	about IRobotGeoSegmentLine 1611
IRobotGeoPoint2D methods 1603	Div 1612
Set 1604	IRobotGeoSegmentLine fields 1612
X 1603	IRobotGeoSegmentLine members 1611
Y 1603	NDiv 1612
IRobotGeoPoint3D 1604	Radius 1612
about IRobotGeoPoint3D 1604	Round 1613
Get 1606	IRobotGeoSegmentType 1607
IRobotGeoPoint3D fields 1605	IRobotGroup 1036
IRobotGeoPoint3D members 1604	about IRobotGroup 1036
IRobotGeoPoint3D methods 1606	Color 1037
Set 1606	CreateCollection 1038
X 1605	CreateSelection 1039
Y 1605	IRobotGroup fields 1037
Z 1605	IRobotGroup members 1036
IRobotGeoPoint3DCollection 1623	IRobotGroup methods 1038
about IRobotGeoPoint3DCollection 1623	Name 1037
IRobotGeoPolyline 1616	ObjectType 1038
about IRobotGeoPolyline 1616	Read 1039
Add 1617	SellList 1038
Clear 1617	Store 1039
IRobotGeoPolyline fields 1617	IRobotGroupObjectServer 843
IRobotGeoPolyline members 1616	about IRobotGroupObjectServer 843
IRobotGeoPolyline methods 1617	Explode 844
Segments 1617	FindFirst 844
IRobotGeoSegment 1610	FindNext 845
about IRobotGeoSegment 1610	GetContents 845
IRobotGeoSegment fields 1610	GroupGiven 845
IRobotGeoSegment members 1610	GroupSelected 846
P1 1611	IRobotGroupObjectServer members 843
Type 1611	IRobotGroupObjectServer methods 844
IRobotGeoSegmentArc 1613	IRobotGroupServer 1040

about IRobotGroupServer 1040	CalcEngine 2197
Create 1041	CmpntFactory 2197
Delete 1041	ConcrReinfEngine 2197
Find 1041	GetExtension 2201
Get 1042	IRobotKernel fields 2196
GetCount 1042	IRobotKernel members 2195
IRobotGroupServer members 1040	IRobotKernel methods 2200
IRobotGroupServer methods 1040	Preferences 2197
IRobotHarmonicAnalysisParams 387	ProgramName 2198
about IRobotHarmonicAnalysisParams 387	ProgramPath 2198
Excitation 387	ProgramVersion 2198
IRobotHarmonicAnalysisParams fields 387	ProjectActiveModel 2199
IRobotHarmonicAnalysisParams members 387	ProjectNew 2201
MassMatrix 388	ProjectNewFromTemplate 2201
IRobotHtmlView 1460	ProjectOpen 2202
about IRobotHtmlView 1460	ProjectPreferences 2199
IRobotHtmlView fields 1461	ProjectSave 2202
IRobotHtmlView members 1461	ProjectSaveAs 2202
IRobotHtmlView methods 1461	ProjectUniqueld 2199
LoadFromFile 1462	Structure 2200
Printable 1461	Version 2200
SaveToFile 1462	IRobotKernelPreferences 2203
IRobotIn3PointsRecordValues 44	about IRobotKernelPreferences 2203
IRobotInContourRecordValues 48	GetDirectory 2204
IRobotIterativePredconditionerType 1566	GetDirectoryExt 2205
IRobotIterativeSolverMemoryUsage 1571	GetLanguage 2205
IRobotIterativeSolverMethod 1570	IRobotKernelPreferences fields 2203
IRobotIterativeSolverParams 1566	IRobotKernelPreferences members 2203
about IRobotIterativeSolverParams 1566	IRobotKernelPreferences methods 2204
AggregationLevelsCount 1567	Multiprocessing 2204
AnalyseDiagonale 1568	SetDirectoryExt 2205
CalcKMatrix 1568	SetLanguage 2206
InternalIterationsCount 1568	IRobotLabel 18
IRobotIterativeSolverParams fields 1567	about IRobotLabel 18
IRobotIterativeSolverParams members 1567	Data 19
MemoryUsage 1569	IRobotLabel fields 19
Method 1569	IRobotLabel members 18
Multilevel 1569	Name 19
PredconditionerType 1570	Type 19
Tolerance 1570	Uniqueld 20
IRobotKernel 2195	IRobotLabelServer 20
about IRobotKernel 2195	about IRobotLabelServer 20

Create 23	about IRobotLinearReleaseDef 707
CreateLike 23	Edgeldx 708
Delete 23	EdgeObject 708
Exist 24	IRobotLinearReleaseDef fields 708
FindWithId 24	IRobotLinearReleaseDef members 707
Get 24	LabelName 709
GetAll 24	Object 709
GetAvailableNames 25	PartIdx 709
GetDefault 25	IRobotLinearReleaseDefinitionType 701
GetMany 25	IRobotLinearReleaseDefList 710
GetPredefinedName 26	about IRobotLinearReleaseDefList 710
GetUniqueld 26	Add 711
IRobotLabelServer members 20	Clear 711
IRobotLabelServer methods 21	Count 710
IsAvailable 26	Get 711
IsPredefinedName 27	IRobotLinearReleaseDefList fields 710
IsUsed 27	IRobotLinearReleaseDefList members 710
SetDefault 27	IRobotLinearReleaseDefList methods 711
Store 28	Remove 712
StoreWithName 28	IRobotLinearReleaseServer 702
IRobotLabelText 17	about IRobotLinearReleaseServer 702
IRobotLanguage 1537	Count 703
IRobotLayoutId 1468	Find 704
IRobotLicenseEntitlement 1539	FindEdge 704
IRobotLicenseEntitlementStatus 1539	FindLabel 705
IRobotLimitState 431	FindObject 705
IRobotLinear3DRecordValues 46	Get 705
IRobotLinearOnEdgesRecordValues 50	GetLabel 706
IRobotLinearRecordValues 45	IRobotLinearReleaseServer fields 703
IRobotLinearReleaseData 698	IRobotLinearReleaseServer members 702
about IRobotLinearReleaseData 698	IRobotLinearReleaseServer methods 703
HX 699	Remove 706
IRobotLinearReleaseData fields 698	Set 706
IRobotLinearReleaseData members 698	SetMany 707
KX 699	IRobotLoadRecord 55
KY 699	about IRobotLoadRecord 55
KZ 700	Description 56
RX 700	GetValue 58
UX 700	IRobotLoadRecord fields 56
UY 701	IRobotLoadRecord members 56
UZ 701	IRobotLoadRecord methods 57
IRobotLinearReleaseDef 707	Objects 57

ObjectType 57	GetGeoLimits 70
SetValue 58	GetPoint 70
Type 57	GetVector 71
IRobotLoadRecord2 74	IRobotLoadRecordInContour fields 69
about IRobotLoadRecord2 74	IRobotLoadRecordInContour members 68
GetGeoLimits 75	IRobotLoadRecordInContour methods 69
IRobotLoadRecord2 fields 75	SetContourPoint 71
IRobotLoadRecord2 members 74	SetGeoLimits 71
IRobotLoadRecord2 methods 75	SetPoint 71
SetGeoLimits 76	SetVector 72
UniqueId 75	UniqueId 69
IRobotLoadRecordBarTrapezoidal 76	IRobotLoadRecordLinear 72
about IRobotLoadRecordBarTrapezoidal 76	about IRobotLoadRecordLinear 72
GetPoint 78	GetPoint 73
IRobotLoadRecordBarTrapezoidal fields 77	IRobotLoadRecordLinear fields 73
IRobotLoadRecordBarTrapezoidal members 76	IRobotLoadRecordLinear members 72
IRobotLoadRecordBarTrapezoidal methods 77	IRobotLoadRecordLinear methods 73
PointCount 77	SetPoint 73
SetPoint 78	UniqueId 73
IRobotLoadRecordCommon 80	IRobotLoadRecordLinear3D 61
about IRobotLoadRecordCommon 80	about IRobotLoadRecordLinear3D 61
IRobotLoadRecordCommon fields 80	GetPoint 62
IRobotLoadRecordCommon members 80	IRobotLoadRecordLinear3D fields 61
IsAutoGenerated 80	IRobotLoadRecordLinear3D members 61
IRobotLoadRecordDead 78	IRobotLoadRecordLinear3D methods 62
about IRobotLoadRecordDead 78	SetPoint 62
FiniteElems 79	UniqueId 62
IRobotLoadRecordDead fields 79	IRobotLoadRecordMngr 58
IRobotLoadRecordDead members 79	about IRobotLoadRecordMngr 58
IRobotLoadRecordIn3Points 63	Count 59
about IRobotLoadRecordIn3Points 63	Create 60
GetGeoLimits 64	Delete 60
GetPoint 64	Get 60
IRobotLoadRecordIn3Points fields 63	IRobotLoadRecordMngr fields 59
IRobotLoadRecordIn3Points members 63	IRobotLoadRecordMngr members 59
IRobotLoadRecordIn3Points methods 64	IRobotLoadRecordMngr methods 59
SetGeoLimits 65	New 60
SetPoint 65	IRobotLoadRecordThermalIn3Points 65
UniqueId 64	about IRobotLoadRecordThermalIn3Points 65
IRobotLoadRecordInContour 68	GetGeoLimits 67
about IRobotLoadRecordInContour 68	GetPoint 67
GetContourPoint 70	IRobotLoadRecordThermalIn3Points fields 66

IRobotLoadRecordThermalIn3Points members	65	LoadFromDBase	666
IRobotLoadRecordThermalIn3Points methods	66	LX	660
SetGeoLimits	67	Name	661
SetPoint	68	NU	661
UniqueID	66	Nuance	661
IRobotLoadRecordType	54	PN_Deformation	661
IRobotMassActivationRecordValues	46	PN_E_Additional	662
IRobotMassEccentricities	204	PN_E_Trans	662
about IRobotMassEccentricities	204	RE	662
IRobotMassEccentricities fields	205	RE_AxCompr	662
IRobotMassEccentricities members	205	RE_AxTens	663
IsDirX	205	RE_Bending	663
IsDirY	205	RE_Shear	663
RelativeValues	206	RE_TrCompr	664
ValueDirX	206	RE_TrTens	664
ValueDirY	206	RO	664
IRobotMassSumServer	923	RT	664
about IRobotMassSumServer	923	SaveToDBase	666
Current	924	SecondName	665
IRobotMassSumServer members	923	Steel_Thermal	665
IRobotMassSumServer methods	924	Timber_Type	665
PartCoeff	924	Type	665
Relative	925	IRobotMaterialDatabase	1300
Total	925	about IRobotMaterialDatabase	1300
IRobotMaterialData	654	Get	1302
about IRobotMaterialData	654	GetAll	1302
CB71_Category	657	GetDefault	1302
CB71_Humidity	657	IRobotMaterialDatabase fields	1301
CB71_Nature	658	IRobotMaterialDatabase members	1300
CB71_Retreat	658	IRobotMaterialDatabase methods	1301
CS	658	Load	1303
Default	658	LoadFromFile	1303
DumpCoef	659	Name	1301
E	659	SetDefault	1303
E_5	659	IRobotMaterialElasticModel	668
E_Trans	659	about IRobotMaterialElasticModel	668
EC_Deformation	660	Coeff	669
GMean	660	IRobotMaterialElasticModel fields	669
IRobotMaterialData fields	656	IRobotMaterialElasticModel members	669
IRobotMaterialData members	655	Model	669
IRobotMaterialData methods	666	UnloadingCoeff	670
Kirchoff	660	UnloadingMethod	670

IRobotMaterialElasticType 668	ElementSize 813
IRobotMaterialElasticUnloadingMethod 670	IRobotMeshGeneration fields 812
IRobotMaterialModel 667	IRobotMeshGeneration members 811
IRobotMaterialQuantitySurvey 1074	Type 813
about IRobotMaterialQuantitySurvey 1074	IRobotMeshGenerationType 809
Count 1075	IRobotMeshImplementDegree 806
GetName 1075	IRobotMeshMethod 809
GetType 1076	about IRobotMeshMethod 809
GetVolume 1076	ForcingRatio 810
GetWeight 1076	ImplementDegree 810
IRobotMaterialQuantitySurvey fields 1075	IRobotMeshMethod fields 810
IRobotMaterialQuantitySurvey members 1074	IRobotMeshMethod members 810
IRobotMaterialQuantitySurvey methods 1075	Method 811
IRobotMaterialTimberType 667	IRobotMeshMethodType 808
IRobotMaterialType 667	IRobotMeshPanelDivType 806
IRobotMeshAccessType 827	IRobotMeshParams 825
IRobotMeshCoonsParams 813	about IRobotMeshParams 825
ForcingRatio 814	Flag 825
IRobotMeshCoonsParams fields 814	IRobotMeshParams fields 825
IRobotMeshCoonsParams members 814	IRobotMeshParams members 825
PanelDivisionType 814	MeshType 826
IRobotMeshDelaunayParams 816	SurfaceParams 826
about IRobotMeshDelaunayParams 816	VolumeParams 826
EmittersDefault 817	IRobotMeshRefinementType 827
EmittersSmoothing 818	IRobotMeshSurfaceFEType 807
EmittersUser 818	IRobotMeshSurfaceFiniteElems 815
H_max 818	about IRobotMeshSurfaceFiniteElems 815
H0 819	ConversionCoeff 815
IRobotMeshDelaunayParams fields 817	ForcingRatio 816
IRobotMeshDelaunayParams members 817	IRobotMeshSurfaceFiniteElems fields 815
MeshDensity 819	IRobotMeshSurfaceFiniteElems members 815
NumberOfLevels 819	Type 816
Q 820	IRobotMeshSurfaceParams 820
RegularMesh 820	about IRobotMeshSurfaceParams 820
Type 820	Coons 821
IRobotMeshDelaunayType 808	Delaunay 822
IRobotMeshForcingRatio 805	FiniteElems 822
IRobotMeshGeneration 811	Generation 822
about IRobotMeshGeneration 811	IRobotMeshSurfaceParams fields 821
Division1 812	IRobotMeshSurfaceParams members 821
Division2 812	Method 822
	IRobotMeshType 805

IRobotMeshVolumeParams 823	about IRobotMobileCaseRoute 333
about IRobotMeshVolumeParams 823	BeginingRouteLimit 334
AdditionalSurfaceMeshing 824	EndRouteLimit 334
FiniteElemsType 824	Geometry 334
IRobotMeshVolumeParams fields 823	GetFactors 336
IRobotMeshVolumeParams members 823	IRobotMobileCaseRoute fields 333
MeshDensity 824	IRobotMobileCaseRoute members 333
IRobotMeshVolumetricFEType 807	IRobotMobileCaseRoute methods 335
IRobotMobileCase 324	LoadDirection 335
about IRobotMobileCase 324	SetFactors 336
ApplicationPlaneBars 325	Step 335
ApplicationPlaneType 325	Tolerance 335
Components 326	IRobotMobileCaseSegmentFactors 336
FindByFlag 328	about IRobotMobileCaseSegmentFactors 336
Flag 326	Gamma 337
GetRoute 328	HL 338
IRobotMobileCase fields 325	HR 338
IRobotMobileCase members 324	IRobotMobileCaseSegmentFactors fields 337
IRobotMobileCase methods 328	IRobotMobileCaseSegmentFactors members 337
IsAuxiliary 326	LL 338
Label 327	LR 339
NatureName 327	VL 339
SetNatureExt 329	VR 339
SetRoute 329	IRobotMobileDistributedRecordValues 52
Uniqueld 327	IRobotMobilePointForceRecordValues 52
Vehicle 327	IRobotModalAnalysisAlgorithm 191
IRobotMobileCaseApplicationPlaneType 340	IRobotModalAnalysisBase 199
IRobotMobileCaseComponent 331	about IRobotModalAnalysisBase 199
about IRobotMobileCaseComponent 331	Add 199
IRobotMobileCaseComponent fields 331	Count 200
IRobotMobileCaseComponent members 331	Delete 200
Point 331	Get 200
Records 332	IRobotModalAnalysisBase members 199
IRobotMobileCaseComponentMngr 329	IRobotModalAnalysisBase methods 199
about IRobotMobileCaseComponentMngr 329	IRobotModalAnalysisLimits 201
Count 330	about IRobotModalAnalysisLimits 201
Get 330	DefineLimits 203
IRobotMobileCaseComponentMngr fields 330	FrequencyLimitValue 202
IRobotMobileCaseComponentMngr members 330	IRobotModalAnalysisLimits fields 201
IRobotMobileCaseComponentMngr methods 330	IRobotModalAnalysisLimits members 201
IRobotMobileCaseFlag 332	IRobotModalAnalysisLimits methods 202
IRobotMobileCaseRoute 333	PeriodLimitValue 202

PulsationLimitValue 202	IRobotModelGenerationParams 1577
IRobotModalAnalysisLimitType 192	about IRobotModelGenerationParams 1577
IRobotModalAnalysisMassMatrixType 191	AssemblingCase 1578
IRobotModalAnalysisMode 190	GenerateNodes_BarsAndFiniteElems 1578
IRobotModalAnalysisParams 192	GenerateNodes_DiagonalBars 1579
about IRobotModalAnalysisParams 192	GenerateNodes_VertHorizBars 1579
Acceleration 194	IRobotModelGenerationParams fields 1577
Base 194	IRobotModelGenerationParams members 1577
Damping 194	MaxElementLength 1579
DisregardDensity 194	NeglectedBars 1580
IncludeDampingInCalculations 195	NeglectedGeoObjects 1580
IRobotModalAnalysisParams fields 193	ToleranceAutomatic 1580
IRobotModalAnalysisParams members 193	ToleranceValue 1581
IterationsCount 195	IRobotModeSelection 15
Limits 195	about IRobotModeSelection 15
MassEccentricities 196	Combination 15
MassMatrix 196	IRobotModeSelection fields 15
MassParticipation 196	IRobotModeSelection members 15
Method 197	Mode 16
Mode 197	Type 16
ModesCount 197	IRobotModeSelectionType 1035
Shifts 198	IRobotMultiCollection 13
SturmVerification 198	about IRobotMultiCollection 13
Tolerance 198	Exist 14
IRobotModalAnalysisShifts 203	Get 14
about IRobotModalAnalysisShifts 203	IRobotMultiCollection members 13
IRobotModalAnalysisShifts fields 203	IRobotMultiCollection methods 14
IRobotModalAnalysisShifts members 203	Set 14
IRobotModalAnalysisShifts methods 204	IRobotMultiSelection 11
IterationsCount 204	about IRobotMultiSelection 11
SetDefault 204	CaseCmpnt 11
IRobotModalWithStaticForcesAnalysisParams 187	Exist 12
about IRobotModalWithStaticForcesAnalysisParams 187	Get 13
CreateFromStatic 189	IRobotMultiSelection fields 11
GetStaticStateParams 190	IRobotMultiSelection members 11
IRobotModalWithStaticForcesAnalysisParams fields 188	IRobotMultiSelection methods 12
IRobotModalWithStaticForcesAnalysisParams members 188	Modes 12
IRobotModalWithStaticForcesAnalysisParams methods 189	Set 13
Nonlinear 189	IRobotNamesArray 1629
SetStaticStateParams 190	about IRobotNamesArray 1629
IRobotModeCombinationType 1035	Count 1630
	Find 1631

Get 1631	HY 489
IRobotNamesArray fields 1630	HZ 490
IRobotNamesArray members 1630	IRobotNodeCompatibilityData fields 486
IRobotNamesArray methods 1631	IRobotNodeCompatibilityData members 485
Set 1631	KX 490
SetSize 1632	KY 490
IRobotNode 522	KZ 491
about IRobotNode 522	NonlinearModel 491
GetCalcSupport 526	RX 491
GetEmitter 526	RY 491
HasCalcSupport 524	RZ 492
HasEmitter 524	UX 492
IRobotNode fields 523	UY 492
IRobotNode members 523	UZ 493
IRobotNode methods 525	IRobotNodeCompatibilityDef 493
IsCalc 524	about IRobotNodeCompatibilityDef 493
RemoveEmitter 526	Bars 494
SetEmitter 527	Compatible 494
UniqueId 524	IRobotNodeCompatibilityDef fields 494
X 525	IRobotNodeCompatibilityDef members 493
Y 525	LabelName 494
Z 525	Main 495
IRobotNodeAccelerationRecordValues 54	IRobotNodeCompatibilityServer 495
IRobotNodeAuxiliaryRecordValues 40	about IRobotNodeCompatibilityServer 495
IRobotNodeBucklingServer 861	Count 496
about IRobotNodeBucklingServer 861	Find 497
EigenVector 862	FindCompatible 497
EigenVectorCmb 862	FindLabel 497
IRobotNodeBucklingServer members 862	FindMain 498
IRobotNodeBucklingServer methods 862	Get 498
IRobotNodeCompatibilityData 485	GetLabel 498
about IRobotNodeCompatibilityData 485	IRobotNodeCompatibilityServer fields 496
Alpha 486	IRobotNodeCompatibilityServer members 495
AX 487	IRobotNodeCompatibilityServer methods 496
AY 487	Remove 499
AZ 487	Set 499
Beta 488	IRobotNodeDisplacementRecordValues 40
BX 488	IRobotNodeDisplacementServer 859
BY 488	about IRobotNodeDisplacementServer 859
BZ 489	DynCombValue 860
Gamma 489	DynValue 861
HX 489	IRobotNodeDisplacementServer members 860

IRobotNodeDisplacementServer methods 860	Remove 484
Value 861	RemoveSlave 484
ValueEx 861	Set 484
IRobotNodeForceInPointRecordValues 49	IRobotNodeServer 527
IRobotNodeForceRecordValues 39	about IRobotNodeServer 527
IRobotNodeResultServer 848	CompatibleNodes 528
about IRobotNodeResultServer 848	Create 530
Buckling 848	FindWithId 530
Displacements 849	FreeNumber 529
IRobotNodeResultServer fields 848	GetCalcNodes 531
IRobotNodeResultServer members 848	GetCalcSupport 531
PseudostaticForces 849	GetConnectedBars 531
Reactions 849	GetUniqueId 532
IRobotNodeRigidLinkData 475	GetUserNodes 532
about IRobotNodeRigidLinkData 475	HasCalcSupport 532
IRobotNodeRigidLinkData fields 476	IRobotNodeServer fields 528
IRobotNodeRigidLinkData members 476	IRobotNodeServer members 527
RX 476	IRobotNodeServer methods 529
RY 477	IsCalc 533
RZ 477	NonlinearLinks 529
UX 477	RemoveEmitter 533
UY 478	RigidLinks 529
UZ 478	SetEmitter 533
IRobotNodeRigidLinkDef 478	IRobotNodeSupportData 499
about IRobotNodeRigidLinkDef 478	about IRobotNodeSupportData 499
IRobotNodeRigidLinkDef fields 479	Advanced 502
IRobotNodeRigidLinkDef members 479	Alpha 502
LabelName 479	AX 502
Master 479	AY 502
Slaves 480	AZ 503
IRobotNodeRigidLinkServer 480	Beta 503
about IRobotNodeRigidLinkServer 480	BX 503
Count 481	BY 503
Find 482	BZ 504
FindLabel 482	ElasticLinear 504
FindMaster 482	ElasticSurface 504
FindSlave 483	Gamma 505
Get 483	GetAdvanced 508
GetLabel 483	GetOneDir 509
IRobotNodeRigidLinkServer fields 481	GlobalCoordSystem 505
IRobotNodeRigidLinkServer members 480	HX 505
IRobotNodeRigidLinkServer methods 481	HY 505

HZ 506	NodeNumber 321
IRobotNodeSupportData fields 501	PDelta 321
IRobotNodeSupportData members 500	ResidualForcesRelativeCodeTolerance 321
IRobotNodeSupportData methods 508	ResultListEachIteration 322
IsFixed 509	SaveSettingsInPreferences 323
KX 506	Stiff 322
KY 506	IRobotNonlinearHingeComponentType 604
KZ 506	IRobotNonlinearHingeData 601
NonlinearModel 506	about IRobotNonlinearHingeData 601
RX 507	GetModel 602
RY 507	IRobotNonlinearHingeData fields 601
RZ 507	IRobotNonlinearHingeData members 601
SetAdvanced 509	IRobotNonlinearHingeData methods 602
SetFixed 509	IsActive 603
SetOneDir 510	NormalStress 602
UX 507	Remove 603
UY 507	SetModel 603
UZ 508	IRobotNonlinearHingeDef 604
IRobotNodeSupportFixingDirection 510	about IRobotNonlinearHingeDef 604
IRobotNodeSupportOneDirectionFixingType 510	Bar 605
IRobotNodeVelocityRecordValues 53	IRobotNonlinearHingeDef fields 605
IRobotNonlinearAnalysisAlgorithmType 314	IRobotNonlinearHingeDef members 604
IRobotNonlinearAnalysisParams 315	LabelName 605
about IRobotNonlinearAnalysisParams 315	Offset 606
Algorithm 317	Relative 606
DegreeOfFreedom 317	IRobotNonlinearHingeModel 590
DisplacementsRelativeCodeTolerance 317	about IRobotNonlinearHingeModel 590
GetSettingsFromPreferences 322	GetAxisParams 595
IncrementLengthReductionFactor 318	GetPoints 595
IncrementLengthReductionNumber 318	IRobotNonlinearHingeModel fields 591
IRobotNonlinearAnalysisParams fields 316	IRobotNonlinearHingeModel members 591
IRobotNonlinearAnalysisParams members 315	IRobotNonlinearHingeModel methods 595
IRobotNonlinearAnalysisParams methods 322	LimitCoordX 592
LineSearchMethodFactor 318	LimitCoordXValue 592
LoadIncrementNumber 318	LimitCoordY 592
MatrixUpdateAfterEachIteration 319	LimitCoordYValue 593
MatrixUpdateAfterEachSubdivision 319	MixedUnloadingValue 593
MaxDisplacement 319	ModelType 593
MaximumIterationNumberForOneIncrement 320	Name 594
MaximumNumberOfBFGSCorrections 320	SetAxisParams 596
MaximumNumberOfLineSearches 320	SetPoints 596
MaxLoadFactor 320	Symetry 594

UnloadingMethod 594	IRobotNonlinearLink methods 676
<b>IRobotNonlinearHingeModelAxisParams</b> 596	ModelType 675
about IRobotNonlinearHingeModelAxisParams 596	Name 675
ImmediateOccupancy 597	SetCurveType 677
<b>IRobotNonlinearHingeModelAxisParams</b> fields 597	SetParams 677
<b>IRobotNonlinearHingeModelAxisParams</b> members 597	Symetry 676
LifeSafety 597	<b>IRobotNonlinearLinkCurveType</b> 681
StructuralStability 598	<b>IRobotNonlinearLinkMngr</b> 694
<b>IRobotNonlinearHingeModelPoints</b> 598	about IRobotNonlinearLinkMngr 694
about IRobotNonlinearHingeModelPoints 598	Delete 694
Count 599	Get 695
Get 600	IRobotNonlinearLinkMngr members 694
<b>IRobotNonlinearHingeModelPoints</b> fields 599	IRobotNonlinearLinkMngr methods 694
<b>IRobotNonlinearHingeModelPoints</b> members 599	IsDefined 695
<b>IRobotNonlinearHingeModelPoints</b> methods 600	Set 695
Set 600	<b>IRobotNonlinearLinkModelType</b> 693
<b>IRobotNonlinearHingeModelServer</b> 588	<b>IRobotNonlinearLinkParams</b> 678
about IRobotNonlinearHingeModelServer 588	about IRobotNonlinearLinkParams 678
Count 589	CurveType 678
Create 589	<b>IRobotNonlinearLinkParams</b> fields 678
Delete 589	<b>IRobotNonlinearLinkParams</b> members 678
Find 590	<b>IRobotNonlinearLinkParamsBLinear</b> 679
Get 590	about IRobotNonlinearLinkParamsBLinear 679
<b>IRobotNonlinearHingeModelServer</b> fields 588	D1 680
<b>IRobotNonlinearHingeModelServer</b> members 588	<b>IRobotNonlinearLinkParamsBLinear</b> fields 680
<b>IRobotNonlinearHingeModelServer</b> methods 589	<b>IRobotNonlinearLinkParamsBLinear</b> members 680
<b>IRobotNonlinearHingeModelType</b> 598	K1 680
<b>IRobotNonlinearHingeModelUnloadingType</b> 600	K2 681
<b>IRobotNonlinearHingeServer</b> 606	<b>IRobotNonlinearLinkParamsCustom</b> 688
about IRobotNonlinearHingeServer 606	about IRobotNonlinearLinkParamsCustom 688
Count 607	Count 689
Get 608	Get 690
<b>IRobotNonlinearHingeServer</b> members 607	<b>IRobotNonlinearLinkParamsCustom</b> fields 689
<b>IRobotNonlinearHingeServer</b> methods 607	<b>IRobotNonlinearLinkParamsCustom</b> members 689
Remove 608	<b>IRobotNonlinearLinkParamsCustom</b> methods 690
Set 608	New 690
<b>IRobotNonlinearLink</b> 674	Remove 690
about IRobotNonlinearLink 674	Set 691
GetCurveType 676	<b>IRobotNonlinearLinkParamsCustomSegment</b> 691
GetParams 677	about IRobotNonlinearLinkParamsCustomSegment 691
<b>IRobotNonlinearLink</b> fields 675	Constant 692
<b>IRobotNonlinearLink</b> members 674	Expression 692

IRobotNonlinearLinkParamsCustomSegment fields	692	about IRobotNonlinearLinkServer	672
IRobotNonlinearLinkParamsCustomSegment members	691	Count	672
OriginPoint	692	Create	673
IRobotNonlinearLinkParamsGapHook	687	Find	673
about IRobotNonlinearLinkParamsGapHook	687	Get	673
D	688	IRobotNonlinearLinkServer fields	672
IRobotNonlinearLinkParamsGapHook fields	688	IRobotNonlinearLinkServer members	672
IRobotNonlinearLinkParamsGapHook members	688	IRobotNonlinearLinkServer methods	673
K	688	Remove	674
IRobotNonlinearLinkParamsLinear	679	IRobotNumbersArray	1632
about IRobotNonlinearLinkParamsLinear	679	about IRobotNumbersArray	1632
IRobotNonlinearLinkParamsLinear fields	679	Count	1633
IRobotNonlinearLinkParamsLinear members	679	Get	1633
K	679	IRobotNumbersArray fields	1633
IRobotNonlinearLinkParamsParabolic	681	IRobotNumbersArray members	1632
about IRobotNonlinearLinkParamsParabolic	681	IRobotNumbersArray methods	1633
Dlim	682	Set	1634
Dmax	683	SetSize	1634
Flim	683	IRobotNumbersCollection	1636
IRobotNonlinearLinkParamsParabolic fields	682	about IRobotNumbersCollection	1636
IRobotNonlinearLinkParamsParabolic members	682	Count	1637
K	683	Get	1637
IRobotNonlinearLinkParamsPlastic	683	IRobotNumbersCollection fields	1637
about IRobotNonlinearLinkParamsPlastic	683	IRobotNumbersCollection members	1636
Dlim	684	IRobotNumbersCollection methods	1637
Flim	684	IRobotNumbersDictionary	1653
IRobotNonlinearLinkParamsPlastic fields	684	about IRobotNumbersDictionary	1653
IRobotNonlinearLinkParamsPlastic members	684	Count	1654
K	685	Delete	1654
W	685	Find	1655
IRobotNonlinearLinkParamsPlasticHardening	685	FindGet	1655
about IRobotNonlinearLinkParamsPlasticHardening	685	Get	1655
Dlim	686	IRobotNumbersDictionary fields	1654
Flim	686	IRobotNumbersDictionary members	1653
IRobotNonlinearLinkParamsPlasticHardening fields	686	IRobotNumbersDictionary methods	1654
IRobotNonlinearLinkParamsPlasticHardening members	685	Set	1656
K0	686	IRobotObjAttributes	757
K1	687	about IRobotObjAttributes	757
W	687	DirZ	759
IRobotNonlinearLinkSemiAxisType	693	GetDirX	760
IRobotNonlinearLinkServer	672	GetLabelName	760
		GetLabels	760

GetLCS 760	Get 803
GetLCSDisplayPosition 761	IRobotObjEdgeSelection fields 802
HasLabel 761	IRobotObjEdgeSelection members 801
IRobotObjAttributes fields 758	IRobotObjEdgeSelection methods 802
IRobotObjAttributes members 758	ToText 803
IRobotObjAttributes methods 759	IRobotObjLocalXDirDefinitionType 788
Meshed 759	IRobotObjMesh 797
RemoveLabel 761	about IRobotObjMesh 797
SetDirX 761	Freeze 798
SetLabel 762	Generate 799
IRobotObjectsArray 1634	GetBasePoints 800
about IRobotObjectsArray 1634	GetQuality 800
Count 1635	IRobotObjMesh fields 798
Get 1635	IRobotObjMesh members 798
IRobotObjectsArray fields 1635	IRobotObjMesh methods 799
IRobotObjectsArray members 1634	IsGenerated 798
IRobotObjectsArray methods 1635	Params 799
Set 1636	Remove 800
SetSize 1636	SetBasePoints 801
IRobotObjectStructuralType 846	IRobotObjModifExtrusion 743
IRobotObjectType 36	about IRobotObjModifExtrusion 743
IRobotObjEdge 754	IRobotObjModifExtrusion fields 744
about IRobotObjEdge 754	IRobotObjModifExtrusion members 744
Divide 755	Vector 744
DivideByPlane 756	IRobotObjModification 739
DivideN 756	about IRobotObjModification 739
GetLabel 756	Add 742
GetLabelName 756	Clear 742
GetLabels 757	Filled 740
HasLabel 757	IRobotObjModification fields 740
IRobotObjEdge fields 754	IRobotObjModification members 740
IRobotObjEdge members 754	IRobotObjModification methods 741
IRobotObjEdge methods 755	NDiv 741
Path 754	Operations 741
RemoveLabel 757	Type 741
SetLabel 757	IRobotObjModificationCollection 763
IRobotObjEdgeCollection 764	about IRobotObjModificationCollection 763
about IRobotObjEdgeCollection 764	IRobotObjModificationType 739
IRobotObjEdgeSelection 801	IRobotObjModifLathe 744
about IRobotObjEdgeSelection 801	about IRobotObjModifLathe 744
Count 802	Angle 745
FromText 802	AxsP1 745

AxsP2 746	BeginMultiOperation 778
IRobotObjModifLathe fields 745	CalcArea 778
IRobotObjModifLathe members 745	CalcVol 779
IRobotObjModifPyramid 746	Create 779
about IRobotObjModifPyramid 746	CreateArc 779
AddPoint 748	CreateCircle 780
ClearPoints 748	CreateCone 780
Factors 747	CreateContour 780
IRobotObjModifPyramid fields 747	CreateCube 781
IRobotObjModifPyramid members 746	CreateCylinder 781
IRobotObjModifPyramid methods 747	CreateOnFiniteElems 781
Points 747	CreatePolyline 782
IRobotObjObject 764	CreateSolid 782
about IRobotObjObject 764	EndMultiOperation 782
AnalyzeTTMethod 766	FindWithId 783
CalcArea 770	FreeNumber 776
CalcVol 770	GetAnalyzeTTMethodEnabled 783
FiniteElems 766	GetFiniteElemsData 783
GetFiniteElemsData 771	GetHost 784
GetHostedObjects 771	GetHostedObjects 784
GetPart 771	GetName 784
GetPartType 772	GetNameTemplate 785
Host 767	GetStructuralType 785
Initialize 772	GetUniqueld 785
IRobotObjObject fields 765	IRobotObjObjectServer fields 775
IRobotObjObject members 764	IRobotObjObjectServer members 773
IRobotObjObject methods 770	IRobotObjObjectServer methods 777
IsVolume 767	IsVolume 786
Main 767	LinearReleases 776
Mesh 767	Mesh 776
Name 768	SetAnalyzeTTMethod 786
NameTemplate 768	SetHost 786
Nodes 768	SetHostedObjects 787
PartsCount 769	SetNameTemplate 787
Reference 769	SetStructuralType 787
SetHostedObjects 772	IRobotObjOperation 742
StructuralType 769	about IRobotObjOperation 742
Uniqueld 769	IRobotObjOperation fields 743
Update 772	IRobotObjOperation members 742
IRobotObjObjectServer 773	Type 743
about IRobotObjObjectServer 773	IRobotObjOperationCollection 743
AutoRecalcHoles 775	about IRobotObjOperationCollection 743

IRobotObjOperationType 739	IRobotObjPart2 members 795
IRobotObjOperMeshing 752	IRobotObjPart2 methods 796
about IRobotObjOperMeshing 752	Nodes 796
Add 753	IRobotObjPartMain 788
Clear 753	about IRobotObjPartMain 788
IRobotObjOperMeshing fields 752	AddModification 792
IRobotObjOperMeshing members 752	CalcArea 792
IRobotObjOperMeshing methods 753	CalcVol 792
Points 752	ClearModifications 793
Vectors 753	CurveDiv 789
IRobotObjOperRotation 750	DefPoints 790
about IRobotObjOperRotation 750	Edges 790
Angle 751	FiniteElems 790
AxsP1 751	Geometry 790
AxsP2 751	IRobotObjPartMain fields 789
IRobotObjOperRotation fields 751	IRobotObjPartMain members 788
IRobotObjOperRotation members 750	IRobotObjPartMain methods 791
IRobotObjOperScaling 749	ModelPoints 791
about IRobotObjOperScaling 749	Modifications 791
Center 750	Nodes 791
Factor 750	IRobotObjPartReference 793
IRobotObjOperScaling fields 749	about IRobotObjPartReference 793
IRobotObjOperScaling members 749	CalcArea 794
IRobotObjOperTranslation 748	CalcVol 795
about IRobotObjOperTranslation 748	FiniteElems 794
IRobotObjOperTranslation fields 749	IRobotObjPartReference fields 793
IRobotObjOperTranslation members 748	IRobotObjPartReference members 793
Vector 749	IRobotObjPartReference methods 794
IRobotObjPart 762	Nodes 794
about IRobotObjPart 762	IRobotObjPartType 773
Attribs 763	IRobotOutputFormat 1495
GetGeometry 763	IRobotPageSetup 1484
IRobotObjPart fields 762	about IRobotPageSetup 1484
IRobotObjPart members 762	Footer 1486
IRobotObjPart methods 763	FromEdgeFooter 1487
Type 763	FromEdgeHeader 1487
IRobotObjPart2 795	GetTemplateName 1493
about IRobotObjPart2 795	Gutter 1487
CalcArea 797	Header 1488
CalcVol 797	IRobotPageSetup fields 1485
FiniteElems 796	IRobotPageSetup members 1484
IRobotObjPart2 fields 796	IRobotPageSetup methods 1493

IsCurrent 1488	Active 976
Load 1493	Color 977
LoadCurrent 1494	DefType 977
MarginBottom 1488	IRobotPanelCut fields 976
MarginLeft 1488	IRobotPanelCut members 976
MarginRight 1489	Point1 977
MarginTop 1489	Point2 978
PageOrientation 1489	Point3 978
PaperHeight 1490	IRobotPanelCutDefinitionType 978
PaperSize 1490	IRobotPanelCutMngr 972
PaperWidth 1490	about IRobotPanelCutMngr 972
Save 1494	Count 973
SaveAs 1494	Create 974
StartPageNumber 1491	Find 974
TemplateCount 1491	Get 974
TemplateName 1491	GetName 975
TextFrame 1492	IRobotPanelCutMngr fields 973
TitlePage 1492	IRobotPanelCutMngr members 973
Toc 1492	IRobotPanelCutMngr methods 974
Variables 1493	Remove 975
IRobotPageSetupFrameType 1507	Store 975
IRobotPageSetupOrientation 1507	IRobotParamCollection 2174
IRobotPageSetupTableOfContents 1508	about IRobotParamCollection 2174
about IRobotPageSetupTableOfContents 1508	Count 2175
Active 1508	Find 2176
IncludeTitle 1509	GetFullName 2176
IRobotPageSetupTableOfContents fields 1508	GetId 2176
IRobotPageSetupTableOfContents members 1508	GetName 2177
Location 1509	GetValue 2177
IRobotPageSetupTocLocation 1509	IRobotParamCollection fields 2175
IRobotPanelCalcModelData 714	IRobotParamCollection members 2175
about IRobotPanelCalcModelData 714	IRobotParamCollection methods 2175
Diaphragm 714	IRobotParamDef 2185
FinElemType 715	about IRobotParamDef 2185
IRobotPanelCalcModelData fields 714	DefaultValue 2187
IRobotPanelCalcModelData members 714	GuiGetName 2189
LoadTransfer 715	GuiReadOnly 2187
IRobotPanelCalcModelDiaphragm 713	GuiSetName 2189
IRobotPanelCalcModelFinElemType 712	GuiVisible 2187
IRobotPanelCalcModelLoadTransfer 713	IRobotParamDef fields 2186
IRobotPanelCut 976	IRobotParamDef members 2186
about IRobotPanelCut 976	IRobotParamDef methods 2189

Name 2188	RemoveByName 2185
UniqueId 2188	SchemaCount 2182
ValueList 2188	IRobotParamServer 2169
ValueType 2188	about IRobotParamServer 2169
IRobotParamSchema 2190	GetAllParams 2171
about IRobotParamSchema 2190	GetAllParamsForSchema 2171
Def 2191	GetObjectsWithParam 2171
GetAllParams 2192	GetObjectsWithParamVal 2172
GetObjectsWithParam 2192	GetParam 2172
GetObjectsWithParamVal 2193	IRobotParamServer fields 2170
GetParam 2193	IRobotParamServer members 2169
IRobotParamSchema fields 2191	IRobotParamServer methods 2170
IRobotParamSchema members 2191	ParamUniqueIdToName 2173
IRobotParamSchema methods 2192	RemoveParam 2173
RemoveAllParams 2193	RemoveSchemaParams 2173
RemoveParam 2194	ResetParam 2174
ResetParam 2194	Schemas 2170
SetParam 2194	SetParam 2174
IRobotParamSchemaDef 2177	IRobotParamValueType 2190
about IRobotParamSchemaDef 2177	IRobotPointAuxiliaryRecordValues 44
AddParam 2179	IRobotPointsArray 1649
AddSimpleParam 2180	about IRobotPointsArray 1649
GetParam 2180	Count 1650
GetParamName 2180	Get 1650
IRobotParamSchemaDef fields 2178	IRobotPointsArray fields 1650
IRobotParamSchemaDef members 2178	IRobotPointsArray members 1649
IRobotParamSchemaDef methods 2179	IRobotPointsArray methods 1650
Name 2178	Set 1651
ParamCount 2179	SetSize 1651
ParamNameToUniqueId 2181	IRobotPredefinedLabel 28
RemoveParam 2181	IRobotPredefinedSelection 1035
IRobotParamSchemaMngr 2181	IRobotPreferences 1533
about IRobotParamSchemaMngr 2181	about IRobotPreferences 1533
Clear 2183	CalculationsType 1534
Create 2183	CloudCalculationsEnabled 1534
Exist 2184	GetDirectory 1536
Get 2184	GetLanguage 1536
GetByName 2184	IRobotPreferences fields 1534
IRobotParamSchemaMngr fields 2182	IRobotPreferences members 1533
IRobotParamSchemaMngr members 2182	IRobotPreferences methods 1535
IRobotParamSchemaMngr methods 2183	Multiprocessing 1535
Remove 2185	OpenGL 1535

IRobotPreferencesEvents 1538	ActiveModel 1336
about IRobotPreferencesEvents 1538	AxisMngr 1337
IRobotPreferencesEvents members 1538	Backgrounds 1337
IRobotPreferencesEvents methods 1539	CalcEngine 1337
OnDialogOK 1539	Close 1342
IRobotPressureRecordValues 45	ComponentMngr 1337
IRobotPrintable 1470	ConcrReinfEngine 1338
about IRobotPrintable 1470	Connections 1338
Comment 1471	DimServer 1338
IncludeDateTime 1471	ExtFileName 1338
IRobotPrintable fields 1471	ExtFileParams 1339
IRobotPrintable members 1471	FileInsertParams 1339
IRobotPrintable methods 1472	FileName 1339
SaveToFile 1472	InsertExtFile 1343
StartFromNewPage 1472	IRobotProject fields 1335
Title 1472	IRobotProject members 1334
IRobotPrintEngine 1473	IRobotProject methods 1342
about IRobotPrintEngine 1473	IsActive 1339
AddScToReport 1476	Name 1340
AddTemplateToReport 1476	New 1343
AddToReport 1477	NewFromTemplate 1343
ClosePreview 1477	Open 1344
CreateReportFromOrganizer 1477	OpenExtFile 1344
ExternalPreviewReport 1478	Preferences 1340
IRobotPrintEngine fields 1474	PrintEngine 1340
IRobotPrintEngine members 1473	ReadExtFileParams 1344
IRobotPrintEngine methods 1475	Save 1345
IsWhilePreview 1474	SaveAs 1345
OrganizerItems 1474	SaveAsExtFile 1345
PageSetup 1475	SaveToFormat 1345
PreviewReport 1478	Structure 1341
PrintReport 1478	Type 1341
RemoveFromReport 1478	UniqueId 1341
RemoveScFromReport 1479	ViewMngr 1341
ReportTemplates 1475	IRobotProjectComponent 1262
ResetReport 1479	about IRobotProjectComponent 1262
SaveReportToFile 1479	Data 1263
SaveReportToOrganizer 1479	IRobotProjectComponent fields 1262
SaveReportToTemplate 1480	IRobotProjectComponent members 1262
ScreenCaptures 1475	Name 1263
IRobotProject 1334	Type 1263
about IRobotProject 1334	IRobotProjectComponentMngr 1263

about IRobotProjectComponentMngr 1263	IRobotProjectPreferencesEvents 1307
Create 1265	about IRobotProjectPreferencesEvents 1307
Get 1265	IRobotProjectPreferencesEvents members 1307
GetLevelName 1266	IRobotProjectPreferencesEvents methods 1307
IRobotProjectComponentMngr fields 1264	OnDialogOK 1307
IRobotProjectComponentMngr members 1264	IRobotProjectSaveFormat 1349
IRobotProjectComponentMngr methods 1265	IRobotProjectType 1333
LevelCount 1264	IRobotProtectionInfo 1644
StdLevelName 1265	about IRobotProtectionInfo 1644
IRobotProjectComponentType 1261	IRobotProtectionInfo members 1645
IRobotProjectEvents 1350	IRobotProtectionInfo methods 1645
about IRobotProjectEvents 1350	IsExtensionEnabled 1645
IRobotProjectEvents members 1350	IRobotPseudostaticForceServer 863
IRobotProjectEvents methods 1350	about IRobotPseudostaticForceServer 863
OnClose 1350	CombValue 863
OnSave 1351	IRobotPseudostaticForceServer members 863
IRobotProjectPreferences 1288	IRobotPseudostaticForceServer methods 863
about IRobotProjectPreferences 1288	Value 864
CalcModelCoherence 1290	IRobotPushOverAnalysisParams 388
EurocodeFactors 1290	about IRobotPushOverAnalysisParams 388
GetActiveCode 1294	Direction 389
GetActiveCodeNumber 1295	IRobotPushOverAnalysisParams fields 389
GetCurrentDatabase 1295	IRobotPushOverAnalysisParams members 389
IRobotProjectPreferences fields 1289	LoadDefinition 390
IRobotProjectPreferences members 1288	MaxDisplacement 390
IRobotProjectPreferences methods 1294	Node 390
KinematicConstraints 1291	Nonlinear 391
Materials 1291	NonlinearParams 391
MeshAutoAdjust 1291	IRobotPushOverDirection 392
MeshAutoAdjustIterationCount 1291	IRobotPushOverLoadDefinitionMethod 391
MeshParams 1292	IRobotQuitOption 1537
MeshParamsFloors 1292	IRobotReactionData 880
MeshParamsWalls 1292	about IRobotReactionData 880
Save 1295	FX 881
SectionsActive 1293	FY 881
SectionsFound 1293	FZ 881
SetActiveCode 1296	IRobotReactionData fields 880
SetActiveCodeNumber 1296	IRobotReactionData members 880
SetCurrentDatabase 1296	MX 881
Units 1293	MY 881
VehiclesActive 1293	MZ 881
VehiclesFound 1294	IRobotReactionServer 849

about IRobotReactionServer 849	IRobotReportItem methods 1506
DDC 852	NoteAfter 1504
DDCEx 852	NoteBefore 1504
DDCLocal 852	StartFromNewPage 1505
DDCLocalEx 852	Title 1505
DDCSum 853	TitleText 1505
DDCSumEx 853	Type 1505
DynCombDDC 853	IRobotReportItemList 1500
DynCombDDCLocal 854	about IRobotReportItemList 1500
DynCombDDCSum 854	Count 1501
DynCombLocal 854	Get 1501
DynCombSum 854	IRobotReportItemList fields 1500
DynCombSumForce 855	IRobotReportItemList members 1500
DynCombValue 855	IRobotReportItemList methods 1501
DynDDC 855	IRobotReportItemType 1510
DynDDCLocal 856	IRobotReportStdElementRtf 1497
DynDDCSum 856	about IRobotReportStdElementRtf 1497
DynLocal 856	Active 1498
DynSum 856	Frame 1498
DynSumForce 857	IRobotReportStdElementRtf fields 1498
DynValue 857	IRobotReportStdElementRtf members 1498
IRobotReactionServer members 850	IRobotReportStdElementRtf methods 1499
IRobotReactionServer methods 851	LoadFromFile 1499
Local 857	RestoreDefaults 1499
LocalEx 857	SaveToFile 1500
Sum 858	IRobotReportTemplateMngr 1495
SumEx 858	about IRobotReportTemplateMngr 1495
SumForce 858	Count 1496
SumForceEx 859	Find 1496
Value 859	Get 1497
ValueEx 859	IRobotReportTemplateMngr fields 1496
IRobotReinforceCalcMethods 951	IRobotReportTemplateMngr members 1495
IRobotReportItem 1501	IRobotReportTemplateMngr methods 1496
about IRobotReportItem 1501	Remove 1497
CreateView 1506	IRobotResultParamType 1019
ExcludeFromReport 1503	IRobotResultQueryParams 1024
GetPageTemplate 1506	about IRobotResultQueryParams 1024
HasNoteAfter 1503	GetParam 1026
HasNoteBefore 1503	IRobotResultQueryParams fields 1024
IncludeDateTime 1504	IRobotResultQueryParams members 1024
IRobotReportItem fields 1502	IRobotResultQueryParams methods 1025
IRobotReportItem members 1502	IsParamSet 1026

Reset 1026	AppendFromFile 1457
ResultIds 1025	Evaluate 1457
Selection 1025	IRobotRtfView fields 1456
SetParam 1027	IRobotRtfView members 1456
IRobotResultQueryReturnType 1027	IRobotRtfView methods 1456
IRobotResultRow 1017	LoadFromFile 1457
about IRobotResultRow 1017	Printable 1456
GetParam 1018	SaveToFile 1458
GetValue 1018	IRobotScreenCaptureMngr 1458
GetValueType 1019	about IRobotScreenCaptureMngr 1458
IRobotResultRow members 1018	Count 1459
IRobotResultRow methods 1018	Find 1459
IsAvailable 1019	Get 1460
IRobotResultRowSet 1021	IRobotScreenCaptureMngr fields 1459
about IRobotResultRowSet 1021	IRobotScreenCaptureMngr members 1458
Clear 1023	IRobotScreenCaptureMngr methods 1459
CurrentRow 1022	Remove 1460
IRobotResultRowSet fields 1022	IRobotSectionDatabase 1304
IRobotResultRowSet members 1021	about IRobotSectionDatabase 1304
IRobotResultRowSet methods 1023	Description 1305
MoveFirst 1023	FullName 1305
MoveNext 1023	GetAll 1306
ResultIds 1022	IRobotSectionDatabase fields 1305
IRobotResultServer 997	IRobotSectionDatabase members 1304
about IRobotResultServer 997	IRobotSectionDatabase methods 1306
Advanced 998	Load 1306
Any 998	LoadFromFile 1307
Bars 999	Name 1305
CalculationResume 999	IRobotSectionDatabaseList 1297
Extremes 999	about IRobotSectionDatabaseList 1297
FiniteElems 1000	Add 1298
IRobotResultServer fields 998	AddFromFile 1299
IRobotResultServer members 997	ChangeIndex 1299
IRobotResultServer methods 1001	Count 1298
Nodes 1000	Find 1299
Query 1001	Get 1299
Status 1000	GetDatabase 1300
Storeys 1000	IRobotSectionDatabaseList fields 1297
Total 1001	IRobotSectionDatabaseList members 1297
IRobotResultStatusType 1027	IRobotSectionDatabaseList methods 1298
IRobotRtfView 1455	Remove 1300
about IRobotRtfView 1455	IRobotSeismicAnalysis_AFPS_90_Params 210

about IRobotSeismicAnalysis_AFPS_90_Params	210	Soil	220
BehaviorFactor	211	StructureType	220
Direction	211	ZoneType	220
DirectionType	211	IRobotSeismicAnalysis_CIRSOC_103_SoilType	257
Filter	212	IRobotSeismicAnalysis_CIRSOC_103_StructureType	258
IRobotSeismicAnalysis_AFPS_90_Params fields	210	IRobotSeismicAnalysis_CIRSOC_103_ZoneType	251
IRobotSeismicAnalysis_AFPS_90_Params members	210	IRobotSeismicAnalysis_DM_16_1_96_Params	221
ResidualMode	212	about IRobotSeismicAnalysis_DM_16_1_96_Params	221
Site	212	Direction	221
SpectrumType	212	Filter	222
StructureType	213	IRobotSeismicAnalysis_DM_16_1_96_Params fields	221
Topography	213	IRobotSeismicAnalysis_DM_16_1_96_Params members	221
ZoneType	213	SeismicCoeff	222
IRobotSeismicAnalysis_AFPS_90_SiteType	254	SeismicProtectionCoeff	222
IRobotSeismicAnalysis_AFPS_90_StructureType	254	IRobotSeismicAnalysis_DM_16_1_96_ProtectionCoeffType	259
IRobotSeismicAnalysis_AFPS_90_ZoneType	250	IRobotSeismicAnalysis_EAK_2000_GroundCategoryType	265
IRobotSeismicAnalysis_CHINESE_DesignType	256	IRobotSeismicAnalysis_EAK_2000_ImportanceFactorType	264
IRobotSeismicAnalysis_CHINESE_EarthquakeType	257	IRobotSeismicAnalysis_EAK_2000_Params	246
IRobotSeismicAnalysis_CHINESE_IntensityType	256	about IRobotSeismicAnalysis_EAK_2000_Params	246
IRobotSeismicAnalysis_CHINESE_Params	214	BehaviorFactor	247
about IRobotSeismicAnalysis_CHINESE_Params	214	Direction	247
DesignStandard	215	DirectionType	248
Direction	215	Filter	248
EarthquakeType	215	FoundationFactor	248
Factor	216	GroundCategory	249
Filter	216	ImportanceFactor	249
Intensity	216	IRobotSeismicAnalysis_EAK_2000_Params fields	247
IRobotSeismicAnalysis_CHINESE_Params fields	214	IRobotSeismicAnalysis_EAK_2000_Params members	246
IRobotSeismicAnalysis_CHINESE_Params members	214	VerticalBehaviorFactor	249
Site	217	VerticalFoundationFactor	249
SiteTg	217	ZoneType	250
StructureType	217	IRobotSeismicAnalysis_EAK_2000_ZoneType	253
IRobotSeismicAnalysis_CHINESE_SiteType	256	IRobotSeismicAnalysis_EC8_General_Params	293
IRobotSeismicAnalysis_CHINESE_StructureType	255	about IRobotSeismicAnalysis_EC8_General_Params	293
IRobotSeismicAnalysis_CIRSOC_103_Params	217	Ag	294
about IRobotSeismicAnalysis_CIRSOC_103_Params	217	B	295
Direction	218	BehaviorFactor	295
DirectionType	219	Direction	295
Filter	219	DirectionType	296
IRobotSeismicAnalysis_CIRSOC_103_Params fields	218	ExcitationDir	296
IRobotSeismicAnalysis_CIRSOC_103_Params members	218		
PlasticityCoeff	219		

Filter 296	ExcitationDir 281
IRobotSeismicAnalysis_EC8_General_Params fields 294	Filter 281
IRobotSeismicAnalysis_EC8_General_Params members 294	I 281
ResidualMode 296	IRobotSeismicAnalysis_IBC_2006_Params fields 280
S 297	IRobotSeismicAnalysis_IBC_2006_Params members 280
Spectrum 297	S1 282
Tb 297	SiteClass 282
Tc 297	Ss 282
Td 298	TL 283
IRobotSeismicAnalysis_EC8_GroundType 298	IRobotSeismicAnalysis_IBC_2006_SiteClassType 283
IRobotSeismicAnalysis_EC8_Params 299	IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction 276
about IRobotSeismicAnalysis_EC8_Params 299	IRobotSeismicAnalysis_ITALY_ORDINANZA_Params 273
Ag 300	about
BehaviorFactor 300	IRobotSeismicAnalysis_ITALY_ORDINANZA_Params 273
Direction 300	Direction 274
DirectionType 301	ExcitationDir 274
ExcitationDir 301	FactorQ 274
Filter 301	Filter 274
GroundType 301	IRobotSeismicAnalysis_ITALY_ORDINANZA_Params fields 273
IRobotSeismicAnalysis_EC8_Params fields 299	IRobotSeismicAnalysis_ITALY_ORDINANZA_Params members 273
IRobotSeismicAnalysis_EC8_Params members 299	Soil 274
ResidualMode 302	Spectrum 275
Spectrum 302	Zone 275
SpectrumType 302	IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType 275
IRobotSeismicAnalysis_EC8_SpectrumType 298	IRobotSeismicAnalysis_ITALY_ORDINANZA_Spectrum 276
IRobotSeismicAnalysis_IBC_2000_Params 242	IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType 275
about IRobotSeismicAnalysis_IBC_2000_Params 242	IRobotSeismicAnalysis_P_100_2006_Params 276
BehaviorFactor 243	about IRobotSeismicAnalysis_P_100_2006_Params 276
Direction 244	Agg 277
ExcitationDir 244	B0 277
Filter 244	BehaviorFactor 277
Ie 245	ExcitationDir 278
IRobotSeismicAnalysis_IBC_2000_Params fields 243	ImportanceFactor 278
IRobotSeismicAnalysis_IBC_2000_Params members 243	IRobotSeismicAnalysis_P_100_2006_Params fields 277
S1 245	IRobotSeismicAnalysis_P_100_2006_Params members 276
SiteClass 245	SpectrumType 278
Ss 246	Tb 278
IRobotSeismicAnalysis_IBC_2000_SiteClassType 264	Tc 279
IRobotSeismicAnalysis_IBC_2006_Params 279	Td 279
about IRobotSeismicAnalysis_IBC_2006_Params 279	IRobotSeismicAnalysis_P_100_92_ImportanceClassType 259
BehaviorFactor 280	
Direction 281	

IRobotSeismicAnalysis_P_100_92_Params	223	about
about IRobotSeismicAnalysis_P_100_92_Params	223	IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope
Direction	224	290
ExcitationDir	224	IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope
Filter	224	members
ImportanceClass	224	290
IRobotSeismicAnalysis_P_100_92_Params	fields	methods
IRobotSeismicAnalysis_P_100_92_Params	members	290
Psi	225	IRobotSeismicAnalysis_PS_92_2008_SiteType
Tc	225	289
ZoneType	225	IRobotSeismicAnalysis_PS_92_2008_StructureType
IRobotSeismicAnalysis_P_100_92_ZoneType	251	289
IRobotSeismicAnalysis_PS_69_DampingType	259	IRobotSeismicAnalysis_PS_92_2008_ZoneType
IRobotSeismicAnalysis_PS_69_Params	225	228
about IRobotSeismicAnalysis_PS_69_Params	225	IRobotSeismicAnalysis_PS_92_Params
Alpha	226	about IRobotSeismicAnalysis_PS_92_Params
Damping	227	228
Delta	227	BehaviorFactor
Direction	227	229
Filter	228	Direction
IRobotSeismicAnalysis_PS_69_Params	fields	230
IRobotSeismicAnalysis_PS_69_Params	members	DirectionType
Soil	228	ExcitationDir
IRobotSeismicAnalysis_PS_69_SoilType	260	Filter
IRobotSeismicAnalysis_PS_92_2008_Params	284	231
about IRobotSeismicAnalysis_PS_92_2008_Params	284	IRobotSeismicAnalysis_PS_92_Params
Ag	285	fields
BehaviorFactor	285	229
Direction	285	IRobotSeismicAnalysis_PS_92_Params
DirectionType	286	members
ExcitationDir	286	229
Filter	286	ResidualMode
IRobotSeismicAnalysis_PS_92_2008_Params	fields	231
IRobotSeismicAnalysis_PS_92_2008_Params	members	Site
ResidualMode	287	231
Site	287	SpectrumType
SiteEnvelope	287	232
SpectrumType	287	StructureType
StructureType	288	232
Topography	288	Topography
ZoneType	288	233
IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope	290	ZoneType
		IRobotSeismicAnalysis_PS_92_SiteEnvelope
		265
		about IRobotSeismicAnalysis_PS_92_SiteEnvelope
		265
		IRobotSeismicAnalysis_PS_92_SiteEnvelope
		members
		266
		IRobotSeismicAnalysis_PS_92_SiteEnvelope
		methods
		266
		IsActive
		266
		SetActive
		266
		IRobotSeismicAnalysis_PS_92_SiteType
		261
		IRobotSeismicAnalysis_PS_92_StructureType
		260
		IRobotSeismicAnalysis_PS_92_ZoneType
		252
		IRobotSeismicAnalysis_RPA_2003_Params
		270
		about IRobotSeismicAnalysis_RPA_2003_Params
		270
		BehaviorFactor
		271
		ExcitationDir
		271
		Filter
		272
		IRobotSeismicAnalysis_RPA_2003_Params
		fields
		271

IRobotSeismicAnalysis_RPA_2003_Params members	236
271	
QualityCoef	272
ResidualMode	272
Site	272
Usage	272
Zone	273
IRobotSeismicAnalysis_RPA_2003_SiteType	270
IRobotSeismicAnalysis_RPA_2003_UsageType	270
IRobotSeismicAnalysis_RPA_2003_ZoneType	270
IRobotSeismicAnalysis_RPA_88_CategoryType	262
IRobotSeismicAnalysis_RPA_88_Params	233
about IRobotSeismicAnalysis_RPA_88_Params	233
Category	234
Direction	234
Filter	234
IRobotSeismicAnalysis_RPA_88_Params fields	234
IRobotSeismicAnalysis_RPA_88_Params members	233
QualityFactor	235
Soil	235
Usage	235
ZoneType	236
IRobotSeismicAnalysis_RPA_88_SoilType	262
IRobotSeismicAnalysis_RPA_88_UsageType	261
IRobotSeismicAnalysis_RPA_88_ZoneType	252
IRobotSeismicAnalysis_RPS_2000_Params	267
about IRobotSeismicAnalysis_RPS_2000_Params	267
BehaviorFactor	268
DirectionType	268
ExcitationDir	268
Filter	269
IRobotSeismicAnalysis_RPS_2000_Params fields	268
IRobotSeismicAnalysis_RPS_2000_Params members	267
ResidualMode	269
Site	269
StructureClass	269
Zone	270
IRobotSeismicAnalysis_RPS_2000_SiteType	267
IRobotSeismicAnalysis_RPS_2000_StructureClass	267
IRobotSeismicAnalysis_RPS_2000_ZoneType	267
IRobotSeismicAnalysis_TURKISH_23098_Params	236
about IRobotSeismicAnalysis_TURKISH_23098_Params	
BehaviorFactor	237
Direction	237
Filter	238
IRobotSeismicAnalysis_TURKISH_23098_Params fields	237
IRobotSeismicAnalysis_TURKISH_23098_Params members	236
SoilType	238
StructureImportance	238
ZoneType	239
IRobotSeismicAnalysis_TURKISH_23098_SoilType	263
IRobotSeismicAnalysis_TURKISH_23098_ZoneType	253
IRobotSeismicAnalysis_UBC_97_Params	239
about IRobotSeismicAnalysis_UBC_97_Params	239
BehaviorFactor	240
ClosestDistance	240
Direction	240
ExcitationDir	241
Filter	241
I	241
IRobotSeismicAnalysis_UBC_97_Params fields	240
IRobotSeismicAnalysis_UBC_97_Params members	239
Soil	242
Source	242
ZoneType	242
IRobotSeismicAnalysis_UBC_97_SoilType	263
IRobotSeismicAnalysis_UBC_97_SourceType	264
IRobotSeismicAnalysis_UBC_97_ZoneType	252
IRobotSeismicAnalysisDirectionType	258
IRobotSeismicAnalysisSpectrumType	255
IRobotSeismicResidualMode	292
about IRobotSeismicResidualMode	292
AugmentationFactor	292
DefinitionMethod	293
IRobotSeismicResidualMode fields	292
IRobotSeismicResidualMode members	292
IsActive	293
LimitFrequency	293
IRobotSeismicResidualModeDefinitionType	291
IRobotSeismicResultsPanelDirection	1576
IRobotSeismicResultsSaveParams	1571
about IRobotSeismicResultsSaveParams	1571

Displacements	1573	Get	1034
Forces	1573	IRobotSelectionFactory members	1031
IRobotSeismicResultsSaveParams	fields	IRobotSelectionFactory methods	1032
IRobotSeismicResultsSaveParams	members	IRobotSELFS seismic_AS_1170_4_Params	463
LocalDisplacements	1573	about IRobotSELFS seismic_AS_1170_4_Params	463
NMQ	1574	IRobotSELFS seismic_AS_1170_4_Params fields	464
OnlyQuadraticCombs	1574	IRobotSELFS seismic_AS_1170_4_Params members	464
PanelsDir	1574	Kp	464
PointNumber	1575	Mi	464
Reactions	1575	Probability	465
Reduced	1575	SoilCategory	465
Save	1576	Sp	465
Stresses	1576	StructureTypeX	465
IRobotSelection	3	StructureTypeY	466
about IRobotSelection	3	Z	466
Add	5	IRobotSELFS seismic_AS_1170_4_ProbabilityType	467
AddOne	5	IRobotSELFS seismic_AS_1170_4_SoilCategoryType	466
AddText	6	IRobotSELFS seismic_AS_1170_4_StructureType	467
And	6	IRobotSELFS seismic_ASCE_7_10_Params	454
AndText	6	about IRobotSELFS seismic_ASCE_7_10_Params	454
Clear	7	I	455
Contains	7	IRobotSELFS seismic_ASCE_7_10_Params fields	455
Count	4	IRobotSELFS seismic_ASCE_7_10_Params members	455
Exclude	7	R	456
ExcludeOne	7	S1	456
ExcludeText	8	SiteClass	456
FromText	8	Ss	456
Get	8	StructureTypeX	457
IRobotSelection	fields	StructureTypeY	457
IRobotSelection	members	TBaseValueX	457
IRobotSelection	methods	TBaseValueY	457
ToText	9	TL	458
Type	4	IRobotSELFS seismic_ASCE_7_10_SiteClassType	454
IRobotSelectionFactory	1031	IRobotSELFS seismic_ASCE_7_10_StructureType	458
about IRobotSelectionFactory	1031	IRobotSELFS seismic_EC_8_Params	458
Create	1032	about IRobotSELFS seismic_EC_8_Params	458
CreateByLabel	1033	Ag	459
CreateByStorey	1033	Beta	460
CreateEdgeSelection	1033	IRobotSELFS seismic_EC_8_Params fields	459
CreateFull	1034	IRobotSELFS seismic_EC_8_Params members	459
CreateMulti	1034	IRobotSELFS seismic_EC_8_Params methods	461
CreatePredefined	1034	Q	460

SiteClass 460	BaseLevelDefMethod 449
SiteClassEnvelopeCheck 461	BaseLevelStorey 449
SiteClassEnvelopelsChecked 462	IRobotSELFSeismicStructureParams fields 448
SpectrumType 460	IRobotSELFSeismicStructureParams members 448
StructureTypeX 461	TopLevelCoordZ 449
StructureTypeY 461	TopLevelDefMethod 450
IRobotSELFSeismic_EC_8_SiteClass 462	TopLevelStorey 450
IRobotSELFSeismic_EC_8_SpectrumType 463	IRobotSELFSeismicTBaseMethod 451
IRobotSELFSeismic_EC_8_StructureType 462	IRobotSerializable 1651
IRobotSELFSeismicAnalysisParams 451	about IRobotSerializable 1651
about IRobotSELFSeismicAnalysisParams 451	IRobotSerializable members 1652
CodeName 452	IRobotSerializable methods 1652
CodeNumber 452	Read 1652
CodeParams 452	Write 1653
Eccentricities 453	IRobotSimpleCase 417
GetExcitationDir 454	about IRobotSimpleCase 417
IRobotSELFSeismicAnalysisParams fields 452	GetAnalysisParams 421
IRobotSELFSeismicAnalysisParams members 451	GetSeismicCode 421
IRobotSELFSeismicAnalysisParams methods 453	IRobotSimpleCase fields 418
StructureParams 453	IRobotSimpleCase members 417
TBaseMethod 453	IRobotSimpleCase methods 420
IRobotSELFSeismicEngine 444	IsAuxiliary 418
about IRobotSELFSeismicEngine 444	Label 418
GenerateLoadCases 445	MainMode 419
GenerationParams 444	ModesCount 419
IRobotSELFSeismicEngine fields 444	NatureName 419
IRobotSELFSeismicEngine members 444	Records 419
IRobotSELFSeismicEngine methods 445	SetAnalysisParams 421
IRobotSELFSeismicGenerationParams 445	SetNatureExt 422
about IRobotSELFSeismicGenerationParams 445	TimeStepCount 420
CodeName 446	UniqueId 420
CodeNumber 446	IRobotSnowWindEngine 95
ExcitationDir 447	about IRobotSnowWindEngine 95
IRobotSELFSeismicGenerationParams fields 446	CodeParams 96
IRobotSELFSeismicGenerationParams members 446	Generate 97
ModalCaseParams 447	Generate3D 97
SeismicParams 447	GenerateStruct3D 97
TBaseMethod 447	IRobotSnowWindEngine fields 96
IRobotSELFSeismicLevelDefinitionMethod 450	IRobotSnowWindEngine members 95
IRobotSELFSeismicStructureParams 448	IRobotSnowWindEngine methods 96
about IRobotSELFSeismicStructureParams 448	Params 96
BaseLevelCoordZ 448	ShowParamsDlg 98

IRobotSnowWindParams 92	about IRobotSpectralAnalysisPointsCollection 310
about IRobotSnowWindParams 92	Add 311
BaseOnGround 93	Clear 311
BaySpacing 93	Count 310
Envelope 93	Get 312
IRobotSnowWindParams fields 92	IRobotSpectralAnalysisPointsCollection fields 310
IRobotSnowWindParams members 92	IRobotSpectralAnalysisPointsCollection members 310
IsolatedRoofs 93	IRobotSpectralAnalysisPointsCollection methods 311
IsSnow 94	LoadFromFile 312
IsWind 94	Remove 312
TotalDepth 94	SaveToFile 313
WithCavities 95	Set 313
WithParapets 95	IRobotSpectralAnalysisSpectrum 305
IRobotSolidPropertiesData 735	about IRobotSpectralAnalysisSpectrum 305
about IRobotSolidPropertiesData 735	AbscissaXAxis 306
DampCoef 736	AbscissaXAxisLogarithmicScale 306
E 737	AddFromTimeHistory 308
IRobotSolidPropertiesData fields 736	Average 309
IRobotSolidPropertiesData members 736	Damping 306
IRobotSolidPropertiesData methods 738	IRobotSpectralAnalysisSpectrum fields 306
LoadFromDBase 738	IRobotSpectralAnalysisSpectrum members 305
LX 737	IRobotSpectralAnalysisSpectrum methods 308
MaterialModel 737	LoadFromFile 309
NU 738	Name 307
RO 738	OrdinateYAxis 307
IRobotSparseMSolverMethod 1565	OrdinateYAxisLogarithmicScale 307
IRobotSparseMSolverParams 1565	Points 308
about IRobotSparseMSolverParams 1565	SaveToFile 309
IRobotSparseMSolverParams fields 1565	IRobotSpectralCoefficients 925
IRobotSparseMSolverParams members 1565	about IRobotSpectralCoefficients 925
Method 1565	IRobotSpectralCoefficients members 926
IRobotSpectralAnalysisAbscissaXAxisType 313	IRobotSpectralCoefficients methods 926
IRobotSpectralAnalysisOrdinateYAxisType 314	ModeCoef 926
IRobotSpectralAnalysisParams 303	PartCoef 926
about IRobotSpectralAnalysisParams 303	SpectrCoef 927
Direction 304	IRobotStorey 1047
ExcitationDir 304	about IRobotStorey 1047
Filter 304	AutomaticSelection 1048
IRobotSpectralAnalysisParams fields 303	Color 1049
IRobotSpectralAnalysisParams members 303	Ex1 1049
Spectrum 304	Ey1 1049
IRobotSpectralAnalysisPointsCollection 310	Height 1049

IRobotStorey fields 1048	FX_ToColumns 1066
IRobotStorey members 1047	FX_ToWalls 1066
IRobotStorey methods 1051	FY 1066
Lx 1050	FY_ToColumns 1067
Ly 1050	FY_ToWalls 1067
Name 1050	FZ 1067
Objects 1051	FZ_ToColumns 1067
SetHeight 1051	FZ_ToWalls 1068
SetTopLevel 1052	IRobotStoreyReducedForces fields 1065
TopLevel 1051	IRobotStoreyReducedForces members 1065
IRobotStoreyDisplacements 1069	MX 1068
about IRobotStoreyDisplacements 1069	MY 1068
DrUX 1070	MZ 1068
DrUY 1070	IRobotStoreyResultServer 1072
IRobotStoreyDisplacements fields 1069	about IRobotStoreyResultServer 1072
IRobotStoreyDisplacements members 1069	Displacements 1073
MaxUX 1070	IRobotStoreyResultServer members 1072
MaxUY 1070	IRobotStoreyResultServer methods 1073
MinUX 1071	ReducedForces 1073
MinUY 1071	Values 1074
NodeMaxUX 1071	IRobotStoreySelection 1056
NodeMaxUY 1071	about IRobotStoreySelection 1056
NodeMinUX 1072	Add 1057
NodeMinUY 1072	AddAll 1058
IRobotStoreyMngr 1052	Clear 1058
about IRobotStoreyMngr 1052	GetNames 1058
BaseLevel 1053	GetNumbers 1059
Count 1053	IRobotStoreySelection members 1057
Create2 1055	IRobotStoreySelection methods 1057
Create2Ex 1055	Remove 1059
Delete 1055	IRobotStoreyValues 1059
DeleteAll 1056	about IRobotStoreyValues 1059
DisregardedObjects 1054	Ex0 1060
FilterStorey 1054	Ex1 1061
Find 1056	Ex2 1061
Get 1056	Ey0 1061
IRobotStoreyMngr fields 1053	Ey1 1062
IRobotStoreyMngr members 1052	Ey2 1062
IRobotStoreyMngr methods 1054	F 1062
IRobotStoreyReducedForces 1064	G 1063
about IRobotStoreyReducedForces 1064	IRobotStoreyValues fields 1060
FX 1066	IRobotStoreyValues members 1060

Ix 1063	IRobotStructuralAxisGrid members 1518
ly 1063	IRobotStructuralAxisGrid methods 1519
lz 1063	Name 1519
Mass 1064	Save 1520
R 1064	Type 1519
IRobotSTRFileAnalyser 2206	IRobotStructuralAxisGridCartesian 1520
about IRobotSTRFileAnalyser 2206	about IRobotStructuralAxisGridCartesian 1520
InsertParams 2207	GetRelativeToPoint 1523
InsertToProject 2208	IncludeStoreysInZ 1521
IRobotSTRFileAnalyser fields 2207	IRobotStructuralAxisGridCartesian fields 1521
IRobotSTRFileAnalyser members 2206	IRobotStructuralAxisGridCartesian members 1520
IRobotSTRFileAnalyser methods 2208	IRobotStructuralAxisGridCartesian methods 1523
Params 2207	RotationAngle 1521
ReadParams 2208	RotationAxis 1522
IRobotSTRParameter 1268	SetRelativeToPoint 1523
about IRobotSTRParameter 1268	StoreysInZ 1522
Description 1270	X 1522
DoubleVal_1 1270	Y 1522
DoubleVal_2 1270	Z 1523
DoubleVal_3 1271	IRobotStructuralAxisGridMngr 1524
FilePathVal 1271	about IRobotStructuralAxisGridMngr 1524
IntegerVal 1271	Activate 1525
IRobotSTRParameter fields 1269	Clear 1525
IRobotSTRParameter members 1269	Count 1525
IsActive 1271	Create 1526
Name 1272	Delete 1526
SelectionVal 1272	FindByNamed 1526
TextList 1272	Get 1527
TextVal 1272	GetByName 1527
Type 1273	IRobotStructuralAxisGridMngr fields 1524
IRobotSTRParams 1267	IRobotStructuralAxisGridMngr members 1524
about IRobotSTRParams 1267	IRobotStructuralAxisGridMngr methods 1525
Count 1267	IsActive 1527
FindParameter 1268	IRobotStructuralAxisGridType 1518
GetParameter 1268	IRobotStructuralAxisLabelType 1513
IRobotSTRParams fields 1267	IRobotStructuralAxisSequenceList 1513
IRobotSTRParams members 1267	about IRobotStructuralAxisSequenceList 1513
IRobotSTRParams methods 1268	AddSequence 1515
IRobotSTRParamType 1273	AxisCount 1514
IRobotStructuralAxisGrid 1518	Clear 1516
about IRobotStructuralAxisGrid 1518	DeleteSequence 1516
IRobotStructuralAxisGrid fields 1519	FindAxisByPos 1516

GetAxis 1516	DSCAlgorithm 1562
GetSequence 1517	EquationSolvingMethod 1562
IRobotStructuralAxisSequenceList fields 1514	FictitiousRigidityCoeff 1562
IRobotStructuralAxisSequenceList members 1513	IgnoreWarnings 1563
IRobotStructuralAxisSequenceList methods 1515	IRobotStructureAnalysisParams fields 1561
SequenceCount 1514	IRobotStructureAnalysisParams members 1560
SetAxisLabel 1517	IterativeParams 1563
SetLabelFormat 1517	ModalParticipationCoeff 1563
SingleOutAxis 1518	RLINKElements 1564
StartPosition 1515	SparseMParams 1564
IRobotStructure 1085	IRobotStructureApplyInfo 1096
about IRobotStructure 1085	about IRobotStructureApplyInfo 1096
ApplyCache 1091	Bars 1096
Bars 1086	IRobotStructureApplyInfo fields 1096
Cases 1087	IRobotStructureApplyInfo members 1096
Clear 1092	Nodes 1097
CreateCache 1092	IRobotStructureAutoVerificationType 1564
Edit 1087	IRobotStructureCache 1093
ExportXml 1092	about IRobotStructureCache 1093
ExtParams 1087	AddBar 1094
FiniteElems 1088	AddNode 1094
GroupObjects 1088	EnsureNodeExist 1095
Groups 1088	IRobotStructureCache members 1093
IRobotStructure fields 1086	IRobotStructureCache methods 1094
IRobotStructure members 1085	SetBarLabel 1095
IRobotStructure methods 1091	SetBarName 1095
IsCalcModelGenerated 1088	IRobotStructureEditTools 1043
Labels 1089	about IRobotStructureEditTools 1043
Merge 1093	DivideBar 1044
Nodes 1089	IRobotStructureEditTools members 1043
Objects 1089	IRobotStructureEditTools methods 1044
QuantitySurvey 1090	SelMirror 1045
Results 1090	SelRotate 1045
ResultsFreeze 1090	SelScale 1045
Selections 1090	SelTranslate 1046
Storeys 1091	TranslateBar 1046
Type 1091	TranslateNode 1046
IRobotStructureAnalysisModalParticipationCoeff 1571	IRobotStructureEvents 1099
IRobotStructureAnalysisParams 1560	about IRobotStructureEvents 1099
about IRobotStructureAnalysisParams 1560	IRobotStructureEvents members 1099
AutoBarMerging 1561	IRobotStructureEvents methods 1099
AutoVerification 1561	ResultStatusChanged 1100

IRobotStructureGeoAnalyser 2210	IRobotSupportColumnFixingType 522
about IRobotStructureGeoAnalyser 2210	IRobotSupportEquivalentColumnElasticity 520
CanEliminateIsolatedNodes 2211	about IRobotSupportEquivalentColumnElasticity 520
CanExtendBars 2211	Fixing1 521
CanUseRigidLinks 2212	Fixing2 521
Correct 2213	IRobotSupportEquivalentColumnElasticity fields 520
DefineIntersection 2213	IRobotSupportEquivalentColumnElasticity members 520
IRobotStructureGeoAnalyser fields 2211	L1 521
IRobotStructureGeoAnalyser members 2210	L2 521
IRobotStructureGeoAnalyser methods 2212	ThroughTwoStories 522
Precision 2212	IRobotSupportEquivalentElasticity 516
SetEdgeSize 2213	about IRobotSupportEquivalentElasticity 516
SetNodeSize 2214	E 516
StartCollectInfo 2214	IRobotSupportEquivalentElasticity fields 516
StopCollectInfo 2214	IRobotSupportEquivalentElasticity members 516
IRobotStructureMergeData 1097	IRobotSupportEquivalentElasticity methods 517
about IRobotStructureMergeData 1097	PoissonRatio 517
CreateStructure 1098	SetMaterial 517
IRobotStructureMergeData fields 1098	Type 517
IRobotStructureMergeData members 1097	IRobotSupportEquivalentElasticityType 518
IRobotStructureMergeData methods 1098	IRobotSupportEquivalentWallElasticity 518
LoadStructure 1099	about IRobotSupportEquivalentWallElasticity 518
Structure 1098	IRobotSupportEquivalentWallElasticity fields 519
IRobotStructureQuantitySurvey 1083	IRobotSupportEquivalentWallElasticity members 518
about IRobotStructureQuantitySurvey 1083	L1 519
BarSections 1084	L2 519
IRobotStructureQuantitySurvey fields 1084	ThroughTwoStories 519
IRobotStructureQuantitySurvey members 1083	IRobotSurfaceOnObjectRecordValues 51
Materials 1084	IRobotSWCodeECCdType 147
PanelThickness 1084	IRobotSWCodeECGroundType 146
IRobotStructureValues 1013	IRobotSWCodeECPParams 130
about IRobotStructureValues 1013	about IRobotSWCodeECPParams 130
GetEx2 1015	Altitude 133
GetEy2 1015	GlobalCDIR 133
GetG 1015	GroundType 133
GetIx 1016	IRobotSWCodeECPParams fields 131
GetLy 1016	IRobotSWCodeECPParams members 130
GetLz 1016	IRobotSWCodeECPParams methods 144
GetMass 1017	LeftWind2NordAngle 133
GetT 1017	NodalLoadsForAllBars 134
IRobotStructureValues members 1014	NodalLoadsForBarsList 134
IRobotStructureValues methods 1014	PermDoorFront 134

PermDoorFrontPresent 134	IRobotSWCodeECSiteType 146
PermDoorLeftSide 135	IRobotSWCodeFRParams 109
PermDoorLeftSidePresent 135	about IRobotSWCodeFRParams 109
PermDoorRear 135	Altitude 113
PermDoorRearPresent 136	IRobotSWCodeFRParams fields 111
PermDoorRightSide 136	IRobotSWCodeFRParams members 109
PermDoorRightSidePresent 136	IRobotSWCodeFRParams methods 126
PermFront 136	IsolatedRoofGetLocation 126
PermLeftSide 137	IsolatedRoofs 113
PermRear 137	IsolatedRoofSetLocation 127
PermRightSide 137	NodalLoadsForAllBars 113
ReferenceLevel 137	NodalLoadsForBarsList 114
SnowBarCoeffGet 145	OpenStructure 114
SnowBarCoeffSet 145	PermDoorFront 114
SnowGutterBars 138	PermDoorFrontPresent 114
SnowPressureExtreme 138	PermDoorLeftSide 115
SnowPressureNormal 138	PermDoorLeftSidePresent 115
SnowRedistribution 138	PermDoorRear 115
StructureAge 139	PermDoorRearPresent 115
StructureAgeCode 139	PermDoorRightSide 116
StructureHeight 139	PermDoorRightSidePresent 116
StructureP 139	PermFront 116
WindAutoCd 140	PermLeftSide 116
WindBarCoeffGet 145	PermRear 117
WindBarCoeffSet 145	PermRightSide 117
WindCALT 140	ReferenceLevel 117
WindCd 140	RiseOfRoof 117
WindCDIR 141	RiseOfRoofAutomatic 118
WindCdType 141	SnowBarCoeffGet 127
WindCt 141	SnowBarCoeffSet 127
WindCtAuto 141	SnowGutterBars 118
WindCTEM 142	SnowIlsWaterOutflow 118
WindE 142	SnowObstacles 119
WindKT 142	SnowPressureExtreme 119
WindPressureAutomatic 142	SnowPressureExtremeAutomatic 119
WindQref 143	SnowPressureNormal 119
WindQref0 143	SnowPressureNormalAutomatic 120
WindQref0p 143	SnowRedistribution 120
WindSiteType 143	SnowRegion 120
WindVref0 144	SnowType 120
WindZ0 144	SnowWaterOutflow 121
WindZMin 144	StructureHeight 121

SurfaceLower 121	SnowBarCoeffSet 107
SurfaceUpper 121	SnowPressure 102
WindBarCoeffGet 128	SnowPressureAutomatic 103
WindBarCoeffSet 128	SnowRedistribution 103
WindCoastalArea 122	SnowZone 103
WindDeltaCoeff 122	StructureHeight 103
WindDeltaCoeffAutomatic 122	WindBarCoeffGet 107
WindDynamicAction 122	WindBarCoeffSet 108
WindDynamicActionCoeff 123	WindDynamicAction 104
WindDynamicActionCoeffAutomatic 123	WindDynamicDecrement 104
WindDynamicActionPeriod 123	WindDynamicPeriod 104
WindDynamicActionSteelStructure 124	WindMultipleRoofs 104
WindFacadeOffset 124	WindPressure 105
WindMultipleRoof 124	WindPressureAutomatic 105
WindPressure 124	WindPressureDistribOnHeight 105
WindPressureAutomatic 125	WindSite 106
WindPressureCeCiMinimum 125	WindZone 106
WindPressureVariable 125	IRobotSWCodePLSnowZone 109
WindRegion 125	IRobotSWCodePLWindPressDistribType 129
WindSite 126	IRobotSWCodePLWindSite 108
WindType 126	IRobotSWCodePLWindZone 108
IRobotSWCodeFRSnowType 129	IRobotSWStruct3D 82
IRobotSWCodeFRSurfaceType 129	about IRobotSWStruct3D 82
IRobotSWCodeFRWindSite 128	FrameCount 82
IRobotSWCodeFRWindType 128	FrameElemCount 83
IRobotSWCodePLParams 98	GetFrame 83
about IRobotSWCodePLParams 98	IRobotSWStruct3D fields 82
Altitude 100	IRobotSWStruct3D members 82
IRobotSWCodePLParams fields 99	IRobotSWStruct3D methods 83
IRobotSWCodePLParams members 98	SetFrame 83
IRobotSWCodePLParams methods 106	IRobotSWStruct3DElement 84
IsolatedRoofGetLocation 106	about IRobotSWStruct3DElement 84
IsolatedRoofs 100	Bar 84
IsolatedRoofSetLocation 107	IRobotSWStruct3DElement fields 84
NodalLoadsForAllBars 101	IRobotSWStruct3DElement members 84
NodalLoadsForBarsList 101	IsFacadeLoaded 85
PermFront 101	IsFacadeOnly 85
PermLeftSide 101	Purlins 85
PermRear 102	IRobotSWStruct3DFrame 85
PermRightSide 102	about IRobotSWStruct3DFrame 85
ReferenceLevel 102	ElemCount 86
SnowBarCoeffGet 107	GetElem 87

IRobotSWStruct3DFrame fields 86	SelectModeFromStore 1365
IRobotSWStruct3DFrame members 86	SetColWidth 1365
IRobotSWStruct3DFrame methods 86	SetRowHeight 1365
SetElem 87	StoreModeSelection 1365
IRobotSWStruct3DGenParams 87	StoreSelection 1366
about IRobotSWStruct3DGenParams 87	Title 1360
Bars 88	UserControl 1360
FacadeLoadedBars 88	Visible 1361
FacadeOnlyBars 89	Window 1361
FrameCount 89	IRobotTableConfig 1356
GetPurlins 90	about IRobotTableConfig 1356
IRobotSWStruct3DGenParams fields 88	IRobotTableConfig members 1357
IRobotSWStruct3DGenParams members 87	IRobotTableConfig methods 1357
IRobotSWStruct3DGenParams methods 90	SetFlag 1357
Offsets 89	SetValue 1357
SetPurlins 90	IRobotTableConfigFlag 1355
Spacings 89	IRobotTableConfigValue 1356
IRobotSWStruct3DPurlinGenParams 90	IRobotTableDataType 1354
about IRobotSWStruct3DPurlinGenParams 90	IRobotTableFrame 1366
IRobotSWStruct3DPurlinGenParams fields 91	about IRobotTableFrame 1366
IRobotSWStruct3DPurlinGenParams members 91	Count 1367
Locations 91	Current 1367
RelativeLocations 91	Get 1368
SectionName 92	GetName 1368
IRobotTable 1357	IRobotTableFrame fields 1367
about IRobotTable 1357	IRobotTableFrame members 1366
AddColumn 1362	IRobotTableFrame methods 1368
ColCount 1359	SetName 1369
Configuration 1359	Window 1368
GetColWidth 1362	IRobotTableScreenCaptureParams 1369
GetDataType 1362	about IRobotTableScreenCaptureParams 1369
GetRowHeight 1363	AddInfoTab 1370
IRobotTable fields 1359	AdjustColumnWidth 1370
IRobotTable members 1358	Comment 1371
IRobotTable methods 1361	IncludeDateAndTime 1371
MakeScreenCapture 1363	IRobotTableScreenCaptureParams fields 1370
Printable 1360	IRobotTableScreenCaptureParams members 1369
RowCount 1360	Name 1371
Select 1363	UpdateType 1371
SelectAllFromStore 1364	IRobotTableScreenCaptureUpdateType 1372
SelectFromStore 1364	IRobotTableType 1352
SelectMode 1364	IRobotThermalIn3PointsRecordValues 47

IRobotThermalRecordValues 49	B1 727
IRobotThicknessData 715	DirType 727
about IRobotThicknessData 715	DisregardBendStiffDirY 728
Data 716	ES 728
ElasticFoundation 716	GetVector 733
IRobotThicknessData fields 716	H 728
IRobotThicknessData members 715	H0 729
MaterialName 717	H1 729
ThicknessType 717	H2 729
Uplift 717	HA 729
IRobotThicknessHomoData 718	HB 730
about IRobotThicknessHomoData 718	HC 730
GetP1 721	IRobotThicknessOrthoData fields 725
GetP2 721	IRobotThicknessOrthoData members 725
GetP3 721	IRobotThicknessOrthoData methods 733
GetReduction 722	Matrix 730
IRobotThicknessHomoData fields 719	N1 731
IRobotThicknessHomoData members 718	N2 731
IRobotThicknessHomoData methods 720	SetVector 733
SetP1 722	T 731
SetP2 722	Thick1 731
SetP3 722	Thick2 732
SetReduction 723	Thick3 732
Thick1 719	Type 732
Thick2 719	VS 732
Thick3 720	IRobotThicknessOrthoDirType 724
ThickConst 720	IRobotThicknessOrthoType 723
Type 720	IRobotThicknessQuantitySurvey 1080
about IRobotThicknessQuantitySurvey 1080	
IRobotThicknessHomoType 723	
IRobotThicknessMatrix 733	Count 1081
about IRobotThicknessMatrix 733	GetArea 1081
GetValue 734	GetName 1082
IRobotThicknessMatrix members 734	GetUnitWeight 1082
IRobotThicknessMatrix methods 734	GetVolume 1082
SetValue 734	GetWeight 1083
IRobotThicknessMatrixValue 735	IRobotThicknessQuantitySurvey fields 1081
IRobotThicknessOrthoData 724	IRobotThicknessQuantitySurvey members 1080
A 726	IRobotThicknessQuantitySurvey methods 1081
A1 726	IRobotThicknessType 717
A2 727	IRobotThicknessUpliftType 718
about IRobotThicknessOrthoData 724	IRobotTimeHistoryAnalysisMethod 366
B 727	IRobotTimeHistoryAnalysisParams 361

about IRobotTimeHistoryAnalysisParams	361	Delete	369
Count	362	DeleteMode	370
Delete	365	Get	370
Division	362	GetDamping	370
End	363	IRobotTimeHistoryModalDecompositionParams	fields 369
Find	365	IRobotTimeHistoryModalDecompositionParams	members 368
Get	365	IRobotTimeHistoryModalDecompositionParams	methods 369
InitialCase	363	IsDefined	371
IRobotTimeHistoryAnalysisParams	fields 362	SetDamping	371
IRobotTimeHistoryAnalysisParams	members 361	IRobotTimeHistoryNewmarkAccelParams	385
IRobotTimeHistoryAnalysisParams	methods 364	about IRobotTimeHistoryNewmarkAccelParams	385
Method	363	Alpha	385
MethodParams	364	Beta	386
Set	366	IRobotTimeHistoryNewmarkAccelParams	fields 385
TimeStep	364	IRobotTimeHistoryNewmarkAccelParams	members 385
IRobotTimeHistoryFunctionList	375	Nonlinearity	386
about IRobotTimeHistoryFunctionList	375	NonlinearParams	386
AddFromFile	377	IRobotTimeHistoryNewmarkParams	366
Count	376	about IRobotTimeHistoryNewmarkParams	366
Create	377	Alpha	367
CreateSum	378	Beta	367
Delete	378	IRobotTimeHistoryNewmarkParams	fields 367
Find	378	IRobotTimeHistoryNewmarkParams	members 367
Get	379	MassMatrixType	368
GetName	379	IRobotTimeHistoryNonlinearParams	380
IRobotTimeHistoryFunctionList	fields 376	about IRobotTimeHistoryNonlinearParams	380
IRobotTimeHistoryFunctionList	members 376	IRobotTimeHistoryNonlinearParams	fields 381
IRobotTimeHistoryFunctionList	methods 376	IRobotTimeHistoryNonlinearParams	members 380
SaveToFile	379	MatrixUpdateAfterEachIteration	381
Store	380	MatrixUpdateAfterEachSubdivision	381
IRobotTimeHistoryHHTParams	382	MaximumIterationNumberForOneIncrement	381
about IRobotTimeHistoryHHTParams	382	PDelta	382
Alpha	383	ResidualForcesRelativeCodeTolerance	382
Beta	383	IRobotTimeHistoryPointsCollection	371
CoeffAlpha	384	about IRobotTimeHistoryPointsCollection	371
IRobotTimeHistoryHHTParams	fields 383	Add	373
IRobotTimeHistoryHHTParams	members 383	Clear	373
Nonlinearity	384	Count	372
NonlinearParams	384	Delete	374
IRobotTimeHistoryModalDecompositionParams	368	Find	374
about IRobotTimeHistoryModalDecompositionParams	368	Get	374
Count	369		

IRobotTimeHistoryPointsCollection fields	372	IRobotUnitEditionData	1286
IRobotTimeHistoryPointsCollection members	372	about IRobotUnitEditionData	1286
IRobotTimeHistoryPointsCollection methods	373	Coefficient	1286
LoadFromFile	375	IRobotUnitEditionData fields	1286
SaveToFile	375	IRobotUnitEditionData members	1286
IRobotTimeHistoryResults	900	Type	1287
about IRobotTimeHistoryResults	900	Unit	1287
ARX	901	IRobotUnitEditionServer	1283
ARY	901	about IRobotUnitEditionServer	1283
ARZ	901	Count	1284
AX	902	Delete	1284
AY	902	Find	1284
AZ	902	Get	1285
IRobotTimeHistoryResults fields	900	IRobotUnitEditionServer members	1283
IRobotTimeHistoryResults members	900	IRobotUnitEditionServer methods	1283
Time	903	New	1285
VRX	903	Set	1285
VRY	903	IRobotUnitEditionType	1277
VRZ	903	IRobotUnitMngr	1278
VX	904	about IRobotUnitMngr	1278
VY	904	Count	1280
VZ	904	Get	1280
IRobotTimeHistoryResultServer	928	GetCoeff	1280
about IRobotTimeHistoryResultServer	928	GetCoeff2	1281
IRobotTimeHistoryResultServer members	928	GetName	1281
IRobotTimeHistoryResultServer methods	929	IRobotUnitMngr fields	1279
Value	929	IRobotUnitMngr members	1278
IRobotTranslateOptions	1043	IRobotUnitMngr methods	1279
IRobotUniformRecordValues	48	Refresh	1281
IRobotUnitComplexData	1282	Set	1282
about IRobotUnitComplexData	1282	UnitEdition	1279
IRobotUnitComplexData fields	1282	UseMetricAsDefault	1279
IRobotUnitComplexData members	1282	IRobotUnitMngrEvents	1287
Name2	1283	about IRobotUnitMngrEvents	1287
IRobotUnitData	1275	IRobotUnitMngrEvents members	1288
about IRobotUnitData	1275	IRobotUnitMngrEvents methods	1288
E	1276	UnitsChanged	1288
IRobotUnitData fields	1276	IRobotUnitType	1275
IRobotUnitData members	1276	IRobotUniversalResultAccess	1002
Name	1276	about IRobotUniversalResultAccess	1002
Precision	1277	Available	1004
Type	1277	Bar	1004

CalcPoint 1004	Delete 1481
DivCount 1005	Exist 1482
DivPoint 1005	GetPredefinedValue 1482
Element 1005	GetValue 1482
GetDirX 1010	IRobotVariableMngr members 1480
IRobotUniversalResultAccess fields 1003	IRobotVariableMngr methods 1481
IRobotUniversalResultAccess members 1002	RemoveExtension 1483
IRobotUniversalResultAccess methods 1010	SetPredefinedValue 1483
Layer 1005	SetValue 1483
LayerArbitraryValue 1006	IRobotVariableMngrExtension 1511
LinearSupports 1006	about IRobotVariableMngrExtension 1511
LoadCase 1006	GetIndexedValue 1512
LoadCaseCmpt 1007	GetValue 1512
Mode 1007	IRobotVariableMngrExtension members 1511
ModeCmb 1007	IRobotVariableMngrExtension methods 1511
Node 1007	IRobotVariablePredefinedId 1483
Panel 1008	IRobotVehicleData 340
ReducedCutPos 1008	about IRobotVehicleData 340
ReinforceCalcMethod 1008	b 341
RelativePoint 1009	d1 341
Reset 1011	d2 341
ResultId 1009	IRobotVehicleData fields 341
ResultType 1009	IRobotVehicleData members 340
ResultValue 1009	IRobotVehicleData methods 342
ResultValue3D 1011	LoadFromDBase 342
SetDirX 1011	Loads 342
Storey 1010	StoreToDBase 343
IRobotUniversalResultType 1013	IRobotVehicleDatabase 353
IRobotUpliftDirection 540	about IRobotVehicleDatabase 353
IRobotUpliftSense 541	Description 354
IRobotValuesArray 1627	GetAll 355
about IRobotValuesArray 1627	IRobotVehicleDatabase fields 354
Count 1628	IRobotVehicleDatabase members 353
Get 1629	IRobotVehicleDatabase methods 355
IRobotValuesArray fields 1628	Load 355
IRobotValuesArray members 1628	LoadFromFile 356
IRobotValuesArray methods 1628	LongName 354
Set 1629	Name 354
SetSize 1629	IRobotVehicleDatabaseList 349
IRobotVariableMngr 1480	about IRobotVehicleDatabaseList 349
about IRobotVariableMngr 1480	Add 351
AddExtension 1481	AddFromFile 351

Count 350	IsLocal 1396
Create 351	ParamsBarMap 1391
Default 350	ParamsDiagram 1391
Find 352	ParamsDisplay 1391
Get 352	ParamsFeMap 1392
GetDatabase 352	Printable 1392
IRobotVehicleDatabaseList fields 349	Projection 1392
IRobotVehicleDatabaseList members 349	Redraw 1396
IRobotVehicleDatabaseList methods 350	Rotate 1396
Remove 353	Selection 1392
IRobotVehicleLoad 345	SetGlobal 1396
about IRobotVehicleLoad 345	SetLocal 1397
DX 346	SetRotationPoint 1397
DY 347	SetScale 1397
F 347	SetSize 1397
IRobotVehicleLoad fields 346	SetWorkPoint 1398
IRobotVehicleLoad members 345	SetZoom 1398
S 347	Title 1393
Type 348	Type 1393
X 348	Visible 1393
IRobotVehicleLoadMngr 343	IRobotView2 1438
about IRobotVehicleLoadMngr 343	about IRobotView2 1438
Count 344	IRobotView2 fields 1440
Delete 344	IRobotView2 members 1439
Get 344	ParamsPanelCut 1440
IRobotVehicleLoadMngr fields 343	UserControl 1440
IRobotVehicleLoadMngr members 343	Window 1440
IRobotVehicleLoadMngr methods 344	IRobotView3 1453
New 345	about IRobotView3 1453
IRobotVehicleLoadType 348	IRobotView3 members 1453
IRobotView 1389	IRobotView3 methods 1454
about IRobotView 1389	MakeScreenCapture 1454
CopyToClipboard 1394	IRobotViewBarMapParams 1424
DiagramMagnification 1391	about IRobotViewBarMapParams 1424
GetRotationPoint 1394	CurrentResult 1425
GetSize 1394	Descriptions 1425
GetSize 1395	IRobotViewBarMapParams fields 1424
GetWorkPoint 1395	IRobotViewBarMapParams members 1424
GetZoom 1395	MapThicknessCoeff 1425
IRobotView fields 1390	StructureDeformation 1425
IRobotView members 1389	IRobotViewBarMapResultType 1423
IRobotView methods 1393	IRobotViewBarMaps 1422

about IRobotViewBarMaps 1422	SetScale 1419
IRobotViewDetailedAnalysis 1406	Values 1417
about IRobotViewDetailedAnalysis 1406	IRobotViewDiagramPositionType 1441
CurrentTableTab 1407	IRobotViewDiagramResultType 1420
IRobotViewDetailedAnalysis fields 1407	IRobotViewDiagrams 1422
IRobotViewDetailedAnalysis members 1406	about IRobotViewDiagrams 1422
IRobotViewDetailedAnalysis methods 1408	IRobotViewDiagramSignDifferType 1422
MakeScreenCapture 1408	IRobotViewDiagramValueType 1454
ParamsDetailed 1407	IRobotViewDisplayAttributes 1411
UserControl 1408	IRobotViewDisplayParams 1409
Window 1408	about IRobotViewDisplayParams 1409
IRobotViewDetailedAnalysisParams 1427	HiddenLines 1410
about IRobotViewDetailedAnalysisParams 1427	IRobotViewDisplayParams fields 1410
Descriptions 1428	IRobotViewDisplayParams members 1409
Filling 1428	IRobotViewDisplayParams methods 1410
GetColor 1429	IsOn 1410
IRobotViewDetailedAnalysisParams fields 1427	Set 1411
IRobotViewDetailedAnalysisParams members 1427	SymbolSize 1410
IRobotViewDetailedAnalysisParams methods 1429	IRobotViewFeMapCrossPresentationType 1437
IsOn 1429	IRobotViewFeMapLayerType 1438
PositiveNegative 1428	IRobotViewFeMapLocalSystemType 1437
ReinforceShowTheoreticAndRealVals 1428	IRobotViewFeMapParams 1431
Set 1429	about IRobotViewFeMapParams 1431
SetColor 1430	CrossPresentation 1433
IRobotViewDetailedAnalysisResultType 1426	CurrentResult 1433
IRobotViewDetailedAnalysisTableTab 1409	DeformationActive 1433
IRobotViewDiagramDescriptionType 1421	DeformationConstScale 1434
IRobotViewDiagramFillingType 1421	Direction 1434
IRobotViewDiagramParams 1415	DirectionData 1434
about IRobotViewDiagramParams 1415	DirectionNode 1434
Descriptions 1416	IRobotViewFeMapParams fields 1432
Filling 1416	IRobotViewFeMapParams members 1432
GetColor 1418	IRobotViewFeMapParams methods 1436
GetScale 1418	Isolines 1435
IRobotViewDiagramParams fields 1416	Layer 1435
IRobotViewDiagramParams members 1415	LayerArbitraryVal 1435
IRobotViewDiagramParams methods 1417	MapScale 1436
IsOn 1418	SetDirectionData 1436
PositiveNegative 1417	Smoothing 1436
ReactionsInLocalSystem 1417	WithDescription 1436
Set 1419	IRobotViewFeMapResultType 1430
SetColor 1419	IRobotViewFeMaps 1423

about IRobotViewFeMaps 1423	IRobotViewInfluenceLinesParams fields 1382
IRobotViewFeMapSmoothingType 1438	IRobotViewInfluenceLinesParams members 1381
IRobotViewGlobalAnalysis 1447	IRobotViewInfluenceLinesParams methods 1385
about IRobotViewGlobalAnalysis 1447	IsOn 1385
IRobotViewGlobalAnalysis fields 1448	Layer 1383
IRobotViewGlobalAnalysis members 1447	LayerArbitraryVal 1384
IRobotViewGlobalAnalysis methods 1449	Position 1384
MakeScreenCapture 1449	RangeFrom 1384
ParamsGlobalAnalysis 1448	RangeTo 1385
IRobotViewGlobalAnalysisParams 1442	Set 1385
about IRobotViewGlobalAnalysisParams 1442	IRobotViewInfluenceLinesResultType 1386
IRobotViewGlobalAnalysisParams fields 1443	IRobotViewMngr 1462
IRobotViewGlobalAnalysisParams members 1442	about IRobotViewMngr 1462
IRobotViewGlobalAnalysisParams methods 1443	CreateFromSc 1466
IsOn 1443	CreateRtfView 1466
Results 1443	CreateTable 1466
Set 1444	CreateView 1466
IRobotViewGlobalAnalysisParamsType 1446	CurrentLayout 1464
IRobotViewGlobalAnalysisResultsParams 1444	GetTable 1467
about IRobotViewGlobalAnalysisResultsParams 1444	GetView 1467
IRobotViewGlobalAnalysisResultsParams fields 1444	IRobotViewMngr fields 1463
IRobotViewGlobalAnalysisResultsParams members 1444	IRobotViewMngr members 1463
NPointsValue 1445	IRobotViewMngr methods 1465
RelativeValue 1445	NewViewAsTab 1464
Type 1445	RecycleViews 1464
IRobotViewGlobalAnalysisResultsType 1446	Refresh 1467
IRobotViewHiddenLinesDisplayType 1415	ShowDialog 1468
IRobotViewInfluenceLines 1379	TableCount 1464
about IRobotViewInfluenceLines 1379	ViewCount 1465
IRobotViewInfluenceLines fields 1380	IRobotViewPanelCutParams 1374
IRobotViewInfluenceLines members 1379	about IRobotViewPanelCutParams 1374
IRobotViewInfluenceLines methods 1380	CurrentReinforcementResult 1375
MakeScreenCapture 1381	CurrentResult 1376
ParamsInfluenceLines 1380	Descriptioms 1376
IRobotViewInfluenceLinesLayerType 1388	Filling 1376
IRobotViewInfluenceLinesLocalSystemType 1387	IntegralValue 1377
IRobotViewInfluenceLinesParams 1381	IRobotViewPanelCutParams fields 1375
about IRobotViewInfluenceLinesParams 1381	IRobotViewPanelCutParams members 1375
Direction 1382	Layer 1377
DirectionData 1382	LayerArbitraryValue 1377
DirectionNode 1383	Position 1378
Element 1383	Smoothing 1378

IRobotViewPanelCuts 1374	IRobotWindLoadsSimulationParams members 469
about IRobotViewPanelCuts 1374	OpeningsClosed 473
IRobotViewProjection 1398	TerrainLevel 473
IRobotViewReinforcementResultType 1441	Velocity 474
IRobotViewScaleType 1405	IRobotWindow 1641
IRobotViewScreenCaptureParams 1449	about IRobotWindow 1641
about IRobotViewScreenCaptureParams 1449	Activate 1643
Comment 1450	Caption 1642
IncludeDateAndTime 1450	Handle 1642
IRobotViewScreenCaptureParams fields 1450	IRobotWindow fields 1642
IRobotViewScreenCaptureParams members 1449	IRobotWindow members 1641
Name 1451	IRobotWindow methods 1643
Orientation 1451	IsActive 1643
Resolution 1451	SendMessage 1644
ScaleAutomatic 1451	State 1643
ScaleValue 1452	IRobotWindowState 1644
UpdateType 1452	
IRobotViewScreenCaptureResolution 1455	<b>J</b>
IRobotViewScreenCaptureUpdateType 1452	Job preferences 1274
IRobotViewType 1388	<b>K</b>
IRobotViewVisibilityStatusType 1399	Knee connection 1930
IRobotViewVisibilityStatusValue 1400	
IRobotWindLoadsSimulationEngine 468	<b>L</b>
about IRobotWindLoadsSimulationEngine 468	Labels - complex attributes of objects 16
Generate 469	Linear releases 697
IRobotWindLoadsSimulationEngine fields 468	Load cases 37
IRobotWindLoadsSimulationEngine members 468	Load records 38
IRobotWindLoadsSimulationEngine methods 469	
Params 468	
IRobotWindLoadsSimulationParams 469	<b>M</b>
about IRobotWindLoadsSimulationParams 469	Material 654
DeviationPercent 470	Member section definition module 1928
DirectionXNEnabled 471	Modal analysis parameters 187
DirectionXNYNENabled 471	Modal analysis parameters recognizing static forces 187
DirectionXNYPEEnabled 471	
DirectionXPEnabled 471	Moving loads 323
DirectionXPYNENabled 472	
DirectionXPYPEnabled 472	
DirectionYNEnabled 472	
DirectionYPEnabled 472	
Elements 473	<b>N</b>
IRobotWindLoadsSimulationParams fields 470	Nodes 474
	Non-linear analysis parameters 314
	Non-linear constraints 671
	Non-linear hinges 587

**O**

Objects 696

**P**

Panel calculation model 712  
 Panel cuts 1374  
 Parameters of simplified seismic analysis 443  
 Pinned column base 1961  
 Plate and shell reinforcement 1692  
 Presentation of data 1351  
 Printouts 1469  
 Project 1100  
 Project components 1100  
 Push-over analysis parameters 388

RobotOM.IRBestCalcParamsDataList.Create 1673  
 RobotOM.IRBestCalcParamsDataList.Delete 1673  
 RobotOM.IRBestCalcParamsDataList.Get 1673  
 RobotOM.IRBestCalcParamsDataList.LabNames 1672  
 RobotOM.IRBestCalcParamsDataList.Selected 1672  
 RobotOM.IRBestCalcParamsDataList.Store 1674  
 RobotOM.IRBestCalcParamsStringValue 1671  
 RobotOM.IRBestCalcParamsDlg 1660  
 RobotOM.IRBestCalcParamsDlg.DoModal 1660  
 RobotOM.IRBestCalcParamsDlg.SetStandard 1661  
 RobotOM.IRBestCalculationType 1688  
 RobotOM.IRBestCalcWarnings 1678  
 RobotOM.IRBestCodeCalcEngine 1682  
 RobotOM.IRBestCodeCalcEngine.Calculate 1683  
 RobotOM.IRBestCodeCalcEngine.GetResults 1683  
 RobotOM.IRBestCodeCalcEngine.SetForces 1684  
 RobotOM.IRBestCodeCalcEngine.SetForcesArray 1684  
 RobotOM.IRBestCodeCalcEngine.SetForcesSlender 1684  
 RobotOM.IRBestCodeCalcEngine.SetGeometry 1684  
 RobotOM.IRBestCodeCalcEngine.SetParams 1685  
 RobotOM.IRBestCodeCalculationType 1687  
 RobotOM.IRBestCodeService 1658

**Q**

Quantity survey 1074  
 Querying mechanism 2

RobotOM.IRBestCodeService 1659  
 RobotOM.IRBestCodeService.CalcEngine 1659  
 RobotOM.IRBestCodeService.CalcParamsDlg 1659  
 RobotOM.IRBestCodeService.MemberDlg 1659  
 RobotOM.IRBestCodeService2 1691  
 RobotOM.IRBestCodeService2.RobotVersion 1692  
 RobotOM.IRBestCodeService2.ShowBucklingButton 1692  
 RobotOM.IRBestCodeServiceExt 1687  
 RobotOM.IRBestCodeServiceExt.GetCalcParamsDlg 1688  
 RobotOM.IRBestCodeServiceExt.IsServed 1688  
 RobotOM.IRBestCoordSystem 1690  
 RobotOM.IRBestDimParams 1681  
 RobotOM.IRBestDimParamsDoubleValue 1689  
 RobotOM.IRBestDimParamsIntegerValue 1681  
 RobotOM.IRBestDirection 1689  
 RobotOM.IRBestForceData 1679  
 RobotOM.IRBestForceData.LimitState 1680  
 RobotOM.IRBestForceData.Values 1681  
 RobotOM.IRBestForceDataDoubleValue 1690  
 RobotOM.IRBestForceDataIntegerValue 1690  
 RobotOM.IRBestForceDataSLSCombType 1691

**R**

RC Beam 1123  
 RC Column 1179  
 RC Deep Beam 1221  
 RC Slab 1207  
 RC Wall 1225  
 Reinforcement 1238  
 Releases 575  
 Results 847  
 Rigid links 475  
 Robot Kernel 2195  
 Robot Object Model 1  
 RobotOM 1  
 RobotOM namespace 1  
 RobotOM.IRBestBendType 1689  
 RobotOM.IRBestCalcErrors 1677  
 RobotOM.IRBestCalcParamsData 1668  
 RobotOM.IRBestCalcParamsData.ModularityList 1669  
 RobotOM.IRBestCalcParamsDataDoubleValue 1669  
 RobotOM.IRBestCalcParamsDataIntegerValue 1670  
 RobotOM.IRBestCalcParamsDataList 1671

RobotOM.IRBestLevel 1689	RobotOM.IRConcr_ACI318_ReinforceData.SetMetricBarDim 1739
RobotOM.IRBestMemberData 1674	RobotOM.IRConcr_ACI318_ReinforceData.SLS_Cracking 1735
RobotOM.IRBestMemberDataDoubleValue 1674	RobotOM.IRConcr_ACI318_ReinforceData.SLS_CreepingCoe f 1735
RobotOM.IRBestMemberDataIntegerValue 1675	RobotOM.IRConcr_ACI318_ReinforceData.SLS_CreepingCoe fValue 1735
RobotOM.IRBestMemberDataStringValue 1685	RobotOM.IRConcr_ACI318_ReinforceData.SLS_Deflection 1736
RobotOM.IRBestMemberDlg 1661	RobotOM.IRConcr_ACI318_ReinforceData.SLS_DeflectionRe infCorrection 1736
RobotOM.IRBestMemberDlg.DoModal 1662	RobotOM.IRConcr_ACI318_ReinforceData.SLS_Factor 1736
RobotOM.IRBestMemberDlg.SetStandard 1662	RobotOM.IRConcr_ACI318_ReinforceData.SLS_FactorValue 1737
RobotOM.IRBestMemberType 1685	RobotOM.IRConcr_ACI318_ReinforceData.SLS_LoadActionP eriod 1737
RobotOM.IRBestParamSet 1662	RobotOM.IRConcr_ACI318_ReinforceData.SLS_LoadRatio 1737
RobotOM.IRBestParamSet.Clear 1664	RobotOM.IRConcr_ACI318_ReinforceData.SLS_MaxCracking 1738
RobotOM.IRBestParamSet.ClearDouble 1664	RobotOM.IRConcr_ACI318_ReinforceData.SLS_MaxDeflectio n 1738
RobotOM.IRBestParamSet.ClearInteger 1665	RobotOM.IRConcr_ACI318_SteelGrades 1740
RobotOM.IRBestParamSet.ClearString 1665	RobotOM.IRConcr_BAEL_ConcreteGrades 1731
RobotOM.IRBestParamSet.GetDouble 1665	RobotOM.IRConcr_BAEL_CrackingType 1730
RobotOM.IRBestParamSet.GetInteger 1665	RobotOM.IRConcr_BAEL_EnvironmentType 1731
RobotOM.IRBestParamSet.GetString 1666	RobotOM.IRConcr_BAEL_ReinforceData 1721
RobotOM.IRBestParamSet.IsValidDouble 1666	RobotOM.IRConcr_BAEL_ReinforceData.CrackAlpha 1724
RobotOM.IRBestParamSet.IsValidInteger 1666	RobotOM.IRConcr_BAEL_ReinforceData.CrackAlphaBot 1724
RobotOM.IRBestParamSet.IsValidString 1666	RobotOM.IRConcr_BAEL_ReinforceData.CrackAlphaTop 1725
RobotOM.IRBestParamSet.SetDouble 1667	RobotOM.IRConcr_BAEL_ReinforceData.CrackEnv 1725
RobotOM.IRBestParamSet.SetIntege 1667	RobotOM.IRConcr_BAEL_ReinforceData.CrackEnvBot 1725
RobotOM.IRBestParamSet.SetString 1667	RobotOM.IRConcr_BAEL_ReinforceData.CrackEnvTop 1725
RobotOM.IRBestPlateCalcParamsDlg 1686	RobotOM.IRConcr_BAEL_ReinforceData.CrackExtraParams 1726
RobotOM.IRBestPlateCalcParamsDlg.DoModal 1686	RobotOM.IRConcr_BAEL_ReinforceData.CrackExtraParamsB ot 1726
RobotOM.IRBestPlateCalcParamsDlg.SetStandard 1687	RobotOM.IRConcr_BAEL_ReinforceData.CrackExtraParamsT op 1726
RobotOM.IRBestResults 1675	RobotOM.IRConcr_BAEL_ReinforceData.Cracking 1726
RobotOM.IRBestResultsDoubleValue 1676	
RobotOM.IRBestResultsIntegerValue 1677	
RobotOM.IRBestResultsStringValue 1678	
RobotOM.IRConcr_ACI318_BarDim 1742	
RobotOM.IRConcr_ACI318_ConcreteParams 1741	
RobotOM.IRConcr_ACI318_ConcreteParams.AsInStructure 1742	
RobotOM.IRConcr_ACI318_LoadActionPeriodType 1740	
RobotOM.IRConcr_ACI318_MetricBarDim 1743	
RobotOM.IRConcr_ACI318_ReinforceData 1733	
RobotOM.IRConcr_ACI318_ReinforceData.GetBarDim 1738	
RobotOM.IRConcr_ACI318_ReinforceData.GetMetricBarDim 1739	
RobotOM.IRConcr_ACI318_ReinforceData.IsMetric 1734	
RobotOM.IRConcr_ACI318_ReinforceData.Main 1735	
RobotOM.IRConcr_ACI318_ReinforceData.SetBarDim 1739	

RobotOM.IRConcr_BAEL_ReinforceData.CrackingBot	1727	ef 1747
RobotOM.IRConcr_BAEL_ReinforceData.CrackingTop	1727	RobotOM.IRConcr_BS8110_ReinforceData.SLS_CreepingCoefValue 1747
RobotOM.IRConcr_BAEL_ReinforceData.CrackWaterLevel	1727	RobotOM.IRConcr_BS8110_ReinforceData.SLS_Deflection 1747
RobotOM.IRConcr_BAEL_ReinforceData.CrackWaterLevelBo	1727	RobotOM.IRConcr_BS8110_ReinforceData.SLS_DeflectionR
RobotOM.IRConcr_BAEL_ReinforceData.CrackWaterLevelTo	1728	einfCorection 1748
RobotOM.IRConcr_BAEL_ReinforceData.Main	1728	RobotOM.IRConcr_BS8110_ReinforceData.SLS_EnvHumidity
RobotOM.IRConcr_BAEL_ReinforceData.SLS_ConcreteStres	1728	sesBot 1748
RobotOM.IRConcr_BAEL_ReinforceData.SLS_ConcreteStres	1728	sesTop 1749
RobotOM.IRConcr_BAEL_ReinforceData.SLS_Deflection	1729	RobotOM.IRConcr_BS8110_ReinforceData.SLS_LoadRatio 1749
RobotOM.IRConcr_BAEL_ReinforceData.SLS_DeflectionRein	1729	fCorrect 1749
RobotOM.IRConcr_BAEL_ReinforceData.SLS_MaxDeflection	1729	RobotOM.IRConcr_BS8110_ReinforceData.SLS_MaxDeflecti
RobotOM.IRConcr_BAEL_ReinforceData.SLS_SteelStresses	1730	on 1749
RobotOM.IRConcr_BAEL_ReinforceData.SLS_SteelStresses	1730	Top 1750
RobotOM.IRConcr_BAEL_ReinforceData.SteelSymbol	1730	RobotOM.IRConcr_EC2_ConcreteGrades 1766
RobotOM.IRConcr_BAEL_SteelGrades	1732	RobotOM.IRConcr_EC2_ExposureRatings 1766
RobotOM.IRConcr_BAEL_WaterLevel	1731	RobotOM.IRConcr_EC2_ITALIAN_SteelGrades 1766
RobotOM.IRConcr_BS8110_ConcreteAge	1750	RobotOM.IRConcr_EC2_NAD 1771
RobotOM.IRConcr_BS8110_ConcreteGrades	1751	RobotOM.IRConcr_EC2_ReinforceData 1766
RobotOM.IRConcr_BS8110_ExposureRatings	1750	RobotOM.IRConcr_EC2_ReinforceData.NAD 1768
RobotOM.IRConcr_BS8110_PartialSafetyFactors	1751	RobotOM.IRConcr_EC2_ReinforceData.SLS_ConcreteAge 1769
RobotOM.IRConcr_BS8110_ReinforceData	1744	RobotOM.IRConcr_EC2_ReinforceData.SLS_Cracking 1769
RobotOM.IRConcr_BS8110_ReinforceData.Main	1745	RobotOM.IRConcr_EC2_ReinforceData.SLS_CrackingReinfC
RobotOM.IRConcr_BS8110_ReinforceData.PartialSafetyFact	1745	orrect 1769
RobotOM.IRConcr_BS8110_ReinforceData.SLS_ConcreteAg	1746	RobotOM.IRConcr_EC2_ReinforceData.SLS_CreepingCoef 1769
RobotOM.IRConcr_BS8110_ReinforceData.SLS_Cracking	1746	RobotOM.IRConcr_EC2_ReinforceData.SLS_CreepingCoefV
RobotOM.IRConcr_BS8110_ReinforceData.SLS_CrackingRei	1746	alue 1769
RobotOM.IRConcr_BS8110_ReinforceData.SLS_CreepingCo	1771	RobotOM.IRConcr_EC2_ReinforceData.SLS_Deflection 1770

RobotOM.IRConcr_IS2000_EnvironmentType 1771	RobotOM.IRConcr_PN84_ReinforceData.LongActionAbove100 1708
RobotOM.IRConcr_IS2000_ReinforceData 1772	RobotOM.IRConcr_PN84_ReinforceData.Main 1708
RobotOM.IRConcr_IS2000_ReinforceData.Main 1774	RobotOM.IRConcr_PN84_ReinforceData.SingleShortLoad 1708
RobotOM.IRConcr_IS2000_ReinforceData.SLS_ConcreteAge 1774	RobotOM.IRConcr_PN84_ReinforceData.SLS_ConcreteAge 1708
RobotOM.IRConcr_IS2000_ReinforceData.SLS_Cracking 1775	RobotOM.IRConcr_PN84_ReinforceData.SLS_Cracking 1709
RobotOM.IRConcr_IS2000_ReinforceData.SLS_CrackingReinfCorrect 1775	RobotOM.IRConcr_PN84_ReinforceData.SLS_CrackingReinfCorrect 1709
RobotOM.IRConcr_IS2000_ReinforceData.SLS_CreepingCoeff 1775	RobotOM.IRConcr_PN84_ReinforceData.SLS_CreepingCoeff 1709
RobotOM.IRConcr_IS2000_ReinforceData.SLS_CreepingCoeffValue 1775	RobotOM.IRConcr_PN84_ReinforceData.SLS_Deflection 1709
RobotOM.IRConcr_IS2000_ReinforceData.SLS_Deflection 1776	RobotOM.IRConcr_PN84_ReinforceData.SLS_DeflectionReinfCorrect 1710
RobotOM.IRConcr_IS2000_ReinforceData.SLS_DeflectionReinfCorrect 1776	RobotOM.IRConcr_PN84_ReinforceData.SLS_EnvHumidity 1710
RobotOM.IRConcr_IS2000_ReinforceData.SLS_ExposureBottom 1776	RobotOM.IRConcr_PN84_ReinforceData.SLS_Exposure 1710
RobotOM.IRConcr_IS2000_ReinforceData.SLS_LoadRatio 1777	RobotOM.IRConcr_PN84_ReinforceData.SLS_LoadsRatio 1711
RobotOM.IRConcr_IS2000_ReinforceData.SLS_MaxCrackingBot 1777	RobotOM.IRConcr_PN84_ReinforceData.SLS_MaxCracking 1711
RobotOM.IRConcr_IS2000_ReinforceData.SLS_MaxCrackingBotEnabled 1777	RobotOM.IRConcr_PN84_ReinforceData.SLS_MaxDeflection 1711
RobotOM.IRConcr_IS2000_ReinforceData.SLS_MaxCrackingTop 1777	RobotOM.IRConcr_PN99_ConcreteGrades 1721
RobotOM.IRConcr_IS2000_ReinforceData.SLS_MaxCrackingTopEnabled 1778	RobotOM.IRConcr_PN99_ExposureRatings 1713
RobotOM.IRConcr_IS2000_ReinforceData.SLS_MaxDeflection 1778	RobotOM.IRConcr_PN99_ReinforceData 1713
RobotOM.IRConcr_PN_SteelGrades 1711	RobotOM.IRConcr_PN99_ReinforceData.IsSpecialStructure 1716
RobotOM.IRConcr_PN84_ConcreteGrades 1712	RobotOM.IRConcr_PN99_ReinforceData.Main 1716
RobotOM.IRConcr_PN84_ExposureRatings 1712	RobotOM.IRConcr_PN99_ReinforceData.SLS_Concrete 1716
RobotOM.IRConcr_PN84_HumidityType 1732	RobotOM.IRConcr_PN99_ReinforceData.SLS_ConcreteAge 1716
RobotOM.IRConcr_PN84_ReinforceData 1705	RobotOM.IRConcr_PN99_ReinforceData.SLS_Cracking 1717
RobotOM.IRConcr_PN84_ReinforceData.CheckConstructionStages 1707	RobotOM.IRConcr_PN99_ReinforceData.SLS_CrackingReinfCorrect 1717
	RobotOM.IRConcr_PN99_ReinforceData.SLS_CreepingCoeff 1717
	RobotOM.IRConcr_PN99_ReinforceData.SLS_CreepingCoeffValue 1717
	RobotOM.IRConcr_PN99_ReinforceData.SLS_Deflection 1718
	RobotOM.IRConcr_PN99_ReinforceData.SLS_DeflectionReinfCorrect 1718

RobotOM.IRConcr_PN99_ReinforceData.SLS_EnvHumidityValue 1718	RobotOM.IRConcrBarSectionData.A 1787 RobotOM.IRConcrBarSectionData.Ac 1787
RobotOM.IRConcr_PN99_ReinforceData.SLS_ExposureBottom 1718	RobotOM.IRConcrBarSectionData.Ap 1787 RobotOM.IRConcrBarSectionData.B 1788
RobotOM.IRConcr_PN99_ReinforceData.SLS_ExposureTop 1719	RobotOM.IRConcrBarSectionData.Bf 1788 RobotOM.IRConcrBarSectionData.Bfb 1788
RobotOM.IRConcr_PN99_ReinforceData.SLS_LoadRatio 1719	RobotOM.IRConcrBarSectionData.H 1788
RobotOM.IRConcr_PN99_ReinforceData.SLS_MaxCrackingBottom 1719	RobotOM.IRConcrBarSectionData.Hf 1788 RobotOM.IRConcrBarSectionData.Hfb 1789
RobotOM.IRConcr_PN99_ReinforceData.SLS_MaxCrackingBottomEnabled 1720	RobotOM.IRConcrBarSectionData.N 1789 RobotOM.IRConcrBarSectionData.Per 1789 RobotOM.IRConcrBarSectionData.Type 1789
RobotOM.IRConcr_PN99_ReinforceData.SLS_MaxCrackingTop 1720	RobotOM.IRConcrBarSectionGeometryType 1790 RobotOM.IRConcrBarSubtype 1250
RobotOM.IRConcr_PN99_ReinforceData.SLS_MaxCrackingTopEnabled 1720	RobotOM.IRConcrBarType 1250 RobotOM.IRConcrBeam 1124
RobotOM.IRConcr_PN99_ReinforceData.SLS_MaxDeflection 1720	RobotOM.IRConcrBeam.Activate 1131 RobotOM.IRConcrBeam.Calculate 1132
RobotOM.IRConcr_SNIP_ConcreteGrades 1763	RobotOM.IRConcrBeam.CalculationOptions 1126
RobotOM.IRConcr_SNIP_ConcreteParams 1763	RobotOM.IRConcrBeam.Concrete 1126
RobotOM.IRConcr_SNIP_ConcreteParams.ConcretingInLayers 1765	RobotOM.IRConcrBeam.CreateCalculationNoteRtf 1132 RobotOM.IRConcrBeam.CreateFromBars 1132
RobotOM.IRConcr_SNIP_ConcreteParams.CuringMethod 1765	RobotOM.IRConcrBeam.GenerateSpliceBars 1132 RobotOM.IRConcrBeam.Geometry 1127
RobotOM.IRConcr_SNIP_ConcreteParams.HighHumidity 1765	RobotOM.IRConcrBeam.ImportOptions 1127
RobotOM.IRConcr_SNIP_ConcreteParams.Type 1765	RobotOM.IRConcrBeam.IsActive 1127
RobotOM.IRConcr_SNIP_ConcreteTypes 1763	RobotOM.IRConcrBeam.IsSelected 1127
RobotOM.IRConcr_SNIP_CuringMethods 1763	RobotOM.IRConcrBeam.LinearLoad 1128
RobotOM.IRConcr_SNIP_Exposure 1765	RobotOM.IRConcrBeam.LinearLoadsCount 1128
RobotOM.IRConcr_SNIP_ReinforceData 1761	RobotOM.IRConcrBeam.Name 1128
RobotOM.IRConcr_SNIP_ReinforceData.SLS_Cracking 1762	RobotOM.IRConcrBeam.NumberOfElements 1128
RobotOM.IRConcr_SNIP_ReinforceData.SLS_CrackingReinfCorrection 1762	RobotOM.IRConcrBeam.PatternOptions 1129 RobotOM.IRConcrBeam.PointLoad 1129 RobotOM.IRConcrBeam.PointLoadsCount 1129
RobotOM.IRConcr_SNIP_ReinforceData.SLS_Exposure 1762	RobotOM.IRConcrBeam.Reinforcement 1129
RobotOM.IRConcr_SNIP_SteelGrades 1763	RobotOM.IRConcrBeam.Save 1133
RobotOM.IRConcrBarDiameters 1257	RobotOM.IRConcrBeam.Steel 1130
RobotOM.IRConcrBarDiameters.Add 1258	RobotOM.IRConcrBeam.StoryOptions 1130
RobotOM.IRConcrBarDiameters.Count 1258	RobotOM.IRConcrBeam.StructureUserNo 1130
RobotOM.IRConcrBarDiameters.Item 1258	RobotOM.IRConcrBeam.StructureUserNoCount 1130
RobotOM.IRConcrBarDiameters.RemoveAll 1259	RobotOM.IRConcrBeam.UniqueId 1131
RobotOM.IRConcrBarSectionData 1786	RobotOM.IRConcrBeam.Verify 1133

RobotOM.IRConcrBeam.VerifySpan 1133	RobotOM.IRConcrBeamLinearLoad.X3 1146
RobotOM.IRConcrBeamCalcOptions 1133	RobotOM.IRConcrBeamLinearLoad.X4 1146
RobotOM.IRConcrBeamCalcOptions.CalcSpanLengthInAxis 1134	RobotOM.IRConcrBeamLinearLoadType 1159
RobotOM.IRConcrBeamCalcOptions.CoverBottom 1135	RobotOM.IRConcrBeamLoadNatureType 1159
RobotOM.IRConcrBeamCalcOptions.CoverBottomFixed 1135	RobotOM.IRConcrBeamPatternOptions 1147
RobotOM.IRConcrBeamCalcOptions.CoverSide 1135	RobotOM.IRConcrBeamPatternOptions.SpanBySpan 1147
RobotOM.IRConcrBeamCalcOptions.CoverSideFixed 1135	RobotOM.IRConcrBeamPointLoad 1148
RobotOM.IRConcrBeamCalcOptions.CoverTop 1136	RobotOM.IRConcrBeamPointLoad.Case 1148
RobotOM.IRConcrBeamCalcOptions.CoverTopFixed 1136	RobotOM.IRConcrBeamPointLoad.Nature 1149
RobotOM.IRConcrBeamCalcPointDefinitionType 1785	RobotOM.IRConcrBeamPointLoad.RelativeCoordinates 1149
RobotOM.IRConcrBeamCrackingType 1162	RobotOM.IRConcrBeamPointLoad.Spans 1149
RobotOM.IRConcrBeamGeometry 1136	RobotOM.IRConcrBeamPointLoad.Type 1149
RobotOM.IRConcrBeamGeometry.AutoNameSpans 1137	RobotOM.IRConcrBeamPointLoad.Value 1150
RobotOM.IRConcrBeamGeometry.AutoNameSupports 1138	RobotOM.IRConcrBeamPointLoad.X1 1150
RobotOM.IRConcrBeamGeometry.ConcreteVolume 1138	RobotOM.IRConcrBeamPointLoadType 1160
RobotOM.IRConcrBeamGeometry.LeftCantilever 1138	RobotOM.IRConcrBeamSectionDimType 1161
RobotOM.IRConcrBeamGeometry.RightCantilever 1138	RobotOM.IRConcrBeamSectionType 1160
RobotOM.IRConcrBeamGeometry.ShutteringArea 1139	RobotOM.IRConcrBeamSegment 1150
RobotOM.IRConcrBeamGeometry.Span 1139	RobotOM.IRConcrBeamSegment.Dim 1151
RobotOM.IRConcrBeamGeometry.SpanNumber 1139	RobotOM.IRConcrBeamSegment.SectionType 1151
RobotOM.IRConcrBeamGeometry.UpdateInternalGeometryD ata 1140	RobotOM.IRConcrBeamSpan 1151
RobotOM.IRConcrBeamImportOptions 1140	RobotOM.IRConcrBeamSpan.CalculationLength 1152
RobotOM.IRConcrBeamImportOptions.GroupByGeometry 1141	RobotOM.IRConcrBeamSpan.HasOpenings 1153
RobotOM.IRConcrBeamImportOptions.GroupByLevel 1141	RobotOM.IRConcrBeamSpan.LeftSupport 1153
RobotOM.IRConcrBeamImportOptions.ImportCombinations 1141	RobotOM.IRConcrBeamSpan.Length 1153
RobotOM.IRConcrBeamImportOptions.LiveLongCoeff 1142	RobotOM.IRConcrBeamSpan.Name 1153
RobotOM.IRConcrBeamImportOptions.RunCalcAuto 1142	RobotOM.IRConcrBeamSpan.RightSupport 1154
RobotOM.IRConcrBeamImportOptions.ShowDialog 1142	RobotOM.IRConcrBeamSpan.Segment 1154
RobotOM.IRConcrBeamLinearLoad 1142	RobotOM.IRConcrBeamSpan.SegmentNumber 1154
RobotOM.IRConcrBeamLinearLoad.Case 1143	RobotOM.IRConcrBeamSpanNumbers 1154
RobotOM.IRConcrBeamLinearLoad.Nature 1144	RobotOM.IRConcrBeamSpanNumbers.Add 1156
RobotOM.IRConcrBeamLinearLoad.RelativeCoordinates 1144	RobotOM.IRConcrBeamSpanNumbers.Count 1155
RobotOM.IRConcrBeamLinearLoad.Spans 1144	RobotOM.IRConcrBeamSpanNumbers.Item 1155
RobotOM.IRConcrBeamLinearLoad.Type 1145	RobotOM.IRConcrBeamSpanNumbers.RemoveAll 1156
RobotOM.IRConcrBeamLinearLoad.Value1 1145	RobotOM.IRConcrBeamStoryOptions 1156
RobotOM.IRConcrBeamLinearLoad.Value2 1145	RobotOM.IRConcrBeamStoryOptions.Cracking 1157
RobotOM.IRConcrBeamLinearLoad.Value3 1145	RobotOM.IRConcrBeamSupport 1157
RobotOM.IRConcrBeamLinearLoad.X1 1146	RobotOM.IRConcrBeamSupport.MaterialType 1158
RobotOM.IRConcrBeamLinearLoad.X2 1146	RobotOM.IRConcrBeamSupport.Name 1158
	RobotOM.IRConcrBeamSupport.Type 1158
	RobotOM.IRConcrBeamSupport.Width 1159
	RobotOM.IRConcrBeamSupportMaterialType 1162

RobotOM.IRConcrBeamSupportType 1162	RobotOM.IRConcrColumn.Uniqueld 1185
RobotOM.IRConcrCalcEngine 1790	RobotOM.IRConcrColumn.UpperColumn 1186
RobotOM.IRConcrCalcEngine.MemberRequiredReinf 1791	RobotOM.IRConcrColumn.UpperColumnDY 1186
RobotOM.IRConcrCalcEngine.SlabRequiredReinf 1791	RobotOM.IRConcrColumn.UpperColumnDZ 1186
RobotOM.IRConcrCodeBeam 1796	RobotOM.IRConcrColumn.Verify 1188
RobotOM.IRConcrCodeBeam.IsEnabled 1797	RobotOM.IRConcrColumnBucklingModel 1196
RobotOM.IRConcrCodeBeam.Verification 1797	RobotOM.IRConcrColumnBucklingModel.IsDirectionYOff 1197
RobotOM.IRConcrCodeBeamCommand 1798	RobotOM.IRConcrColumnBucklingModel.IsDirectionZOff 1197
RobotOM.IRConcrCodeColumn 1793	RobotOM.IRConcrColumnBucklingModel.IsSwayY 1197
RobotOM.IRConcrCodeColumn.IsEnabled 1794	RobotOM.IRConcrColumnBucklingModel.IsSwayZ 1198
RobotOM.IRConcrCodeColumn.Verification 1794	RobotOM.IRConcrColumnBucklingModel.IsTotalStructureHeight 1198
RobotOM.IRConcrCodeColumnCommand 1795	RobotOM.IRConcrColumnBucklingModel.ky 1198
RobotOM.IRConcrCodeReport 1795	RobotOM.IRConcrColumnBucklingModel.kz 1198
RobotOM.IRConcrCodeReport.AddError 1796	RobotOM.IRConcrColumnBucklingModel.Ly 1199
RobotOM.IRConcrCodeReport.AddMessage 1796	RobotOM.IRConcrColumnBucklingModel.Lz 1199
RobotOM.IRConcrCodeReport.AddWarning 1796	RobotOM.IRConcrColumnBucklingModel.NumberOfStories 1199
RobotOM.IRConcrCodeService 1792	RobotOM.IRConcrColumnBucklingModel.TotalStructureHeight 1199
RobotOM.IRConcrCodeService.Beam 1792	RobotOM.IRConcrColumnCalcOptions 1188
RobotOM.IRConcrCodeService.Column 1793	RobotOM.IRConcrColumnCalcOptions.Cover 1189
RobotOM.IRConcrCodeService.IsConcrComponentServed 1793	RobotOM.IRConcrColumnDimensionType 1195
RobotOM.IRConcrColumn 1179	RobotOM.IRConcrColumnGeometry 1189
RobotOM.IRConcrColumn.Activate 1187	RobotOM.IRConcrColumnGeometry.ConcreteVolume 1190
RobotOM.IRConcrColumn.BucklingModel 1181	RobotOM.IRConcrColumnGeometry.Dim 1190
RobotOM.IRConcrColumn.Calculate 1187	RobotOM.IRConcrColumnGeometry.DimIsFixed 1190
RobotOM.IRConcrColumn.CalculationOptions 1182	RobotOM.IRConcrColumnGeometry.Section 1191
RobotOM.IRConcrColumn.Concrete 1182	RobotOM.IRConcrColumnGeometry.SectionName 1191
RobotOM.IRConcrColumn.CreateCalculationNoteRtf 1187	RobotOM.IRConcrColumnGeometry.ShutteringArea 1191
RobotOM.IRConcrColumn.CreateFromBars 1187	RobotOM.IRConcrColumnImportOptions 1191
RobotOM.IRConcrColumn.Geometry 1182	RobotOM.IRConcrColumnImportOptions.GroupByGeometry 1192
RobotOM.IRConcrColumn.HasUpperColumn 1182	RobotOM.IRConcrColumnImportOptions.GroupByLevel 1193
RobotOM.IRConcrColumn.ImportOptions 1183	RobotOM.IRConcrColumnImportOptions.ImportCombinations 1193
RobotOM.IRConcrColumn.IsActive 1183	RobotOM.IRConcrColumnImportOptions.RunCalcAuto 1193
RobotOM.IRConcrColumn.IsSelected 1183	RobotOM.IRConcrColumnImportOptions.ShowDialog 1193
RobotOM.IRConcrColumn.Loads 1183	RobotOM.IRConcrColumnLoad 1202
RobotOM.IRConcrColumn.Name 1184	RobotOM.IRConcrColumnLoad.CaseName 1203
RobotOM.IRConcrColumn.NumberOfElements 1184	RobotOM.IRConcrColumnLoad.CaseType 1203
RobotOM.IRConcrColumn.PatternOptions 1184	RobotOM.IRConcrColumnLoad.Fy 1203
RobotOM.IRConcrColumn.Reinforcement 1184	RobotOM.IRConcrColumnLoad.Fz 1204
RobotOM.IRConcrColumn.Save 1188	
RobotOM.IRConcrColumn.Steel 1185	
RobotOM.IRConcrColumn.StructureUserNo 1185	
RobotOM.IRConcrColumn.StructureUserNoCount 1185	

RobotOM.IRConcrColumnLoad.Gamma 1204	RobotOM.IRConcrContinuousFooting.NumberOfElements 1167
RobotOM.IRConcrColumnLoad.MnsY 1204	RobotOM.IRConcrContinuousFooting.PatternOptions 1167
RobotOM.IRConcrColumnLoad.MnsZ 1204	RobotOM.IRConcrContinuousFooting.Reinforcement 1167
RobotOM.IRConcrColumnLoad.MyA 1205	RobotOM.IRConcrContinuousFooting.Save 1169
RobotOM.IRConcrColumnLoad.MyB 1205	RobotOM.IRConcrContinuousFooting.Steel 1167
RobotOM.IRConcrColumnLoad.MyC 1205	RobotOM.IRConcrContinuousFooting.StructureUserNo 1168
RobotOM.IRConcrColumnLoad.MzA 1205	RobotOM.IRConcrContinuousFooting.StructureUserNoCount 1168
RobotOM.IRConcrColumnLoad.MzB 1206	RobotOM.IRConcrContinuousFooting.Uniquelid 1168
RobotOM.IRConcrColumnLoad.MzC 1206	RobotOM.IRConcrContinuousFooting.Verify 1170
RobotOM.IRConcrColumnLoad.N 1206	RobotOM.IRConcrContinuousFootingGeometry 1170
RobotOM.IRConcrColumnLoad.NdN 1206	RobotOM.IRConcrContinuousFootingGeometry.AutoNameSpans 1171
RobotOM.IRConcrColumnLoadCaseType 1207	RobotOM.IRConcrContinuousFootingGeometry.AutoNameSupports 1171
RobotOM.IRConcrColumnLoads 1200	RobotOM.IRConcrContinuousFootingGeometry.ConcreteVolume 1172
RobotOM.IRConcrColumnLoads.Add 1201	RobotOM.IRConcrContinuousFootingGeometry.LeftCantilever 1172
RobotOM.IRConcrColumnLoads.Count 1200	RobotOM.IRConcrContinuousFootingGeometry.RightCantilever 1172
RobotOM.IRConcrColumnLoads.Item 1201	RobotOM.IRConcrContinuousFootingGeometry.ShutteringArea 1172
RobotOM.IRConcrColumnLoads.RemoveAll 1201	RobotOM.IRConcrContinuousFootingGeometry.Span 1173
RobotOM.IRConcrColumnPatternOptions 1194	RobotOM.IRConcrContinuousFootingGeometry.SpanNumber 1173
RobotOM.IRConcrColumnPatternOptions.TiesToBeam 1194	RobotOM.IRConcrContinuousFootingGeometry.UpdateInternalIGeometryData 1173
RobotOM.IRConcrColumnSectionType 1195	RobotOM.IRConcrContinuousFootingSectionDimType 1178
RobotOM.IRConcrConcrete 1255	RobotOM.IRConcrContinuousFootingSectionType 1177
RobotOM.IRConcrConcrete.CharacteristicStrength 1256	RobotOM.IRConcrContinuousFootingSegment 1173
RobotOM.IRConcrConcreteParams 1700	RobotOM.IRConcrContinuousFootingSegment.Dim 1174
RobotOM.IRConcrConcreteParams.Ec 1701	RobotOM.IRConcrContinuousFootingSegment.SectionType 1174
RobotOM.IRConcrConcreteParams.Fc 1701	RobotOM.IRConcrContinuousFootingSpan 1175
RobotOM.IRConcrConcreteParams.Fc_calc 1702	RobotOM.IRConcrContinuousFootingSpan.CalculationLength 1176
RobotOM.IRConcrConcreteParams.Fcj 1702	RobotOM.IRConcrContinuousFootingSpan.LeftSupport 1176
RobotOM.IRConcrConcreteParams.Ft 1702	RobotOM.IRConcrContinuousFootingSpan.Length 1176
RobotOM.IRConcrConcreteParams.Grade 1702	RobotOM.IRConcrContinuousFootingSpan.Name 1176
RobotOM.IRConcrContinuousFooting 1163	RobotOM.IRConcrContinuousFootingSpan.RightSupport 1177
RobotOM.IRConcrContinuousFooting.Activate 1169	
RobotOM.IRConcrContinuousFooting.Calculate 1169	
RobotOM.IRConcrContinuousFooting.CalculationOptions 1165	
RobotOM.IRConcrContinuousFooting.Concrete 1165	
RobotOM.IRConcrContinuousFooting.CreateCalculationNoteIf 1169	
RobotOM.IRConcrContinuousFooting.Geometry 1165	
RobotOM.IRConcrContinuousFooting.ImportOptions 1166	
RobotOM.IRConcrContinuousFooting.IsActive 1166	
RobotOM.IRConcrContinuousFooting.IsSelected 1166	
RobotOM.IRConcrContinuousFooting.Name 1166	

RobotOM.IRConcrContinuousFootingSpan.Segment 1177	RobotOM.IRConcrFooting.Reinforcement 1106
RobotOM.IRConcrContinuousFootingSpan.SegmentNumber 1177	RobotOM.IRConcrFooting.Results 1106
RobotOM.IRConcrContinuousFootingSupportMaterialType 1178	RobotOM.IRConcrFooting.Save 1110
RobotOM.IRConcrDeepBeam 1221	RobotOM.IRConcrFooting.Steel 1107
RobotOM.IRConcrDeepBeam.Activate 1224	RobotOM.IRConcrFooting.StructureUserNo 1107
RobotOM.IRConcrDeepBeam.Concrete 1222	RobotOM.IRConcrFooting.StructureUserNoCount 1107
RobotOM.IRConcrDeepBeam.CreateCalculationNoteRtf 1224	RobotOM.IRConcrFooting.Uniqueld 1107
RobotOM.IRConcrDeepBeam.CreateFromObjects 1225	RobotOM.IRConcrFooting.Verify 1110
RobotOM.IRConcrDeepBeam.Geometry 1222	RobotOM.IRConcrFootingCalculationOptions 1118
RobotOM.IRConcrDeepBeam.IsActive 1222	RobotOM.IRConcrFootingCalculationOptions.Cover 1119
RobotOM.IRConcrDeepBeam.IsSelected 1223	RobotOM.IRConcrFootingCalculationOptions.NominalCover 1119
RobotOM.IRConcrDeepBeam.Name 1223	RobotOM.IRConcrFootingCalculationOptions.NominalPierCover 1119
RobotOM.IRConcrDeepBeam.Reinforcement 1223	RobotOM.IRConcrFootingCalculationOptions.PierCover 1120
RobotOM.IRConcrDeepBeam.Save 1225	RobotOM.IRConcrFootingDimType 1113
RobotOM.IRConcrDeepBeam.Steel 1223	RobotOM.IRConcrFootingGeometry 1110
RobotOM.IRConcrDeepBeam.Uniqueld 1224	RobotOM.IRConcrFootingGeometry.ConcreteVolume 1111
RobotOM.IRConcrDeepBeamGeometry 1225	RobotOM.IRConcrFootingGeometry.GetDim 1113
RobotOM.IRConcrDrawing 1259	RobotOM.IRConcrFootingGeometry.PierType 1111
RobotOM.IRConcrDrawing.Group 1260	RobotOM.IRConcrFootingGeometry.SetDim 1113
RobotOM.IRConcrDrawing.GroupCount 1260	RobotOM.IRConcrFootingGeometry.Shape 1112
RobotOM.IRConcrDrawingBarGroup 1260	RobotOM.IRConcrFootingGeometry.ShutteringArea 1112
RobotOM.IRConcrDrawingBarGroup.Bar 1261	RobotOM.IRConcrFootingGeometry.Type 1112
RobotOM.IRConcrDrawingBarGroup.BarCount 1261	RobotOM.IRConcrFootingGround 1115
RobotOM.IRConcrFooting 1101	RobotOM.IRConcrFootingGround.ColumnPierLevel 1116
RobotOM.IRConcrFooting.Activate 1108	RobotOM.IRConcrFootingGround.SoilLevel 1116
RobotOM.IRConcrFooting.Calculate 1108	RobotOM.IRConcrFootingLoads 1116
RobotOM.IRConcrFooting.CalculationOptions 1103	RobotOM.IRConcrFootingPattern 1120
RobotOM.IRConcrFooting.Concrete 1103	RobotOM.IRConcrFootingPattern.Diameter1 1121
RobotOM.IRConcrFooting.CreateCalculationNoteRtf 1109	RobotOM.IRConcrFootingPattern.Diameter2 1121
RobotOM.IRConcrFooting.CreateFromNodes 1109	RobotOM.IRConcrFootingPattern.Spacing1Max 1121
RobotOM.IRConcrFooting.CreateReinforcement 1109	RobotOM.IRConcrFootingPattern.Spacing1Min 1122
RobotOM.IRConcrFooting.DisplayDrawing 1109	RobotOM.IRConcrFootingPattern.Spacing2Max 1122
RobotOM.IRConcrFooting.FootRotation 1104	RobotOM.IRConcrFootingPattern.Spacing2Min 1122
RobotOM.IRConcrFooting.Geometry 1104	RobotOM.IRConcrFootingPierType 1123
RobotOM.IRConcrFooting.Ground 1104	RobotOM.IRConcrFootingResults 1117
RobotOM.IRConcrFooting.IsActive 1105	RobotOM.IRConcrFootingResults.Ax 1117
RobotOM.IRConcrFooting.IsSelected 1105	RobotOM.IRConcrFootingResults.Ay 1118
RobotOM.IRConcrFooting.Loads 1105	RobotOM.IRConcrFootingResults.Val 1118
RobotOM.IRConcrFooting.Name 1105	RobotOM.IRConcrFootingShapeType 1114
RobotOM.IRConcrFooting.NumberOfElements 1106	RobotOM.IRConcrFootingType 1114
RobotOM.IRConcrFooting.Pattern 1106	

RobotOM.IRConcrFootResultType 1122	RobotOM.IRConcrReinforceData.Cover_Bot 1697
RobotOM.IRConcrMemberRequiredReinfCalcParams 1782	RobotOM.IRConcrReinforceData.Cover_Up 1698
RobotOM.IRConcrMemberRequiredReinfCalcParams.BeamPointsType 1783	RobotOM.IRConcrReinforceData.GetMainDirection 1698
RobotOM.IRConcrMemberRequiredReinfCalcParams.BeamPointsValue 1783	RobotOM.IRConcrReinforceData.ReinforcingSteel 1698
RobotOM.IRConcrMemberRequiredReinfCalcParams.CasesA LS 1784	RobotOM.IRConcrReinforceData.SetMainDirection 1699
RobotOM.IRConcrMemberRequiredReinfCalcParams.CasesS LS 1784	RobotOM.IRConcrReinforceData2 1757
RobotOM.IRConcrMemberRequiredReinfCalcParams.CasesU LS 1784	RobotOM.IRConcrReinforceData2.Main 1758
RobotOM.IRConcrMemberRequiredReinfCalcParams.CombA LS 1784	RobotOM.IRConcrReinforceDataMain 1758
RobotOM.IRConcrMemberRequiredReinfCalcParams.CombS LS 1785	RobotOM.IRConcrReinforceDataMain.MembraneReinfInOneLayer 1760
RobotOM.IRConcrMemberRequiredReinfCalcParams.CombU LS 1785	RobotOM.IRConcrReinforceDataMain.MinimumReinf 1760
RobotOM.IRConcrMemberRequiredReinfCalcParams.Member s 1785	RobotOM.IRConcrReinforceDataMain.ReinfCalcType 1760
RobotOM.IRConcrMemberRequiredReinfEngine 1780	RobotOM.IRConcrReinforceDataMain.UnidirReinf 1760
RobotOM.IRConcrMemberRequiredReinfEngine.Calculate 1781	RobotOM.IRConcrReinforcement 1238
RobotOM.IRConcrMemberRequiredReinfEngine.GetCalculate dMembers 1781	RobotOM.IRConcrReinforcement.Bars 1239
RobotOM.IRConcrMemberRequiredReinfEngine.Params 1781	RobotOM.IRConcrReinforcement.Comment 1239
RobotOM.IRConcrMinimumReinforcementType 1771	RobotOM.IRConcrReinforcement.Freeze 1241
RobotOM.IRConcrPlateCodeService2 1778	RobotOM.IRConcrReinforcement.GroupGlobally 1240
RobotOM.IRConcrPlateCodeService2.RobotVersion 1779	RobotOM.IRConcrReinforcement.GroupingMethod 1240
RobotOM.IRConcrPlateCodeService2.ShowLongTermCrackin g 1779	RobotOM.IRConcrReinforcement.IsFrozen 1240
RobotOM.IRConcrReinforceBarType 1743	RobotOM.IRConcrReinforcement.LockRefresh 1241
RobotOM.IRConcrReinforceCalcType 1757	RobotOM.IRConcrReinforcement.RemoveAll 1241
RobotOM.IRConcrReinforceData 1694	RobotOM.IRConcrReinforcement.RemoveParametric 1242
RobotOM.IRConcrReinforceData.BarDim_D1_Bot 1696	RobotOM.IRConcrReinforcingBar 1242
RobotOM.IRConcrReinforceData.BarDim_D1_Up 1696	RobotOM.IRConcrReinforcingBar.CreateCopy 1247
RobotOM.IRConcrReinforceData.BarDim_D2_Bot 1696	RobotOM.IRConcrReinforcingBar.DirectionX 1243
RobotOM.IRConcrReinforceData.BarDim_D2_Up 1697	RobotOM.IRConcrReinforcingBar.DirectionY 1244
RobotOM.IRConcrReinforceData.CodeName 1697	RobotOM.IRConcrReinforcingBar.DirectionZ 1244
RobotOM.IRConcrReinforceData.Concrete 1697	RobotOM.IRConcrReinforcingBar.GetROBarEditor 1247

RobotOM.IRConcrReinforcingBars 1248	RobotOM.IRConcrSlabRequiredReinfCalcParams.Method 1754
RobotOM.IRConcrReinforcingBars.Add 1249	RobotOM.IRConcrSlabRequiredReinfCalcParams.Panels 1755
RobotOM.IRConcrReinforcingBars.Count 1249	RobotOM.IRConcrSlabRequiredReinfEngine 1755
RobotOM.IRConcrReinforcingBars.Item 1249	RobotOM.IRConcrSlabRequiredReinfEngine.Calculate 1756
RobotOM.IRConcrSlab 1208	RobotOM.IRConcrSlabRequiredReinfEngine.GetCalculatedPanels 1757
RobotOM.IRConcrSlab.Activate 1211	RobotOM.IRConcrSlabRequiredReinfEngine.Params 1756
RobotOM.IRConcrSlab.CalculationOptions 1209	RobotOM.IRConcrSlabRnfSegmentType 1220
RobotOM.IRConcrSlab.Concrete 1209	RobotOM.IRConcrSlabRnfType 1219
RobotOM.IRConcrSlab.CreateCalculationNoteRtf 1212	RobotOM.IRConcrSlabSupportType 1220
RobotOM.IRConcrSlab.CreateFromObjects 1212	RobotOM.IRConcrSlabWizard 1216
RobotOM.IRConcrSlab.Geometry 1209	RobotOM.IRConcrSlabWizard.AddColumnSupport 1218
RobotOM.IRConcrSlab.IsSelected 1210	RobotOM.IRConcrSlabWizard.AddOpening 1218
RobotOM.IRConcrSlab.Name 1210	RobotOM.IRConcrSlabWizard.AddOpeningNode 1218
RobotOM.IRConcrSlab.PatternOptions 1210	RobotOM.IRConcrSlabWizard.AddSlabNode 1219
RobotOM.IRConcrSlab.Reinforcement 1210	RobotOM.IRConcrSlabWizard.AddWallSupport 1219
RobotOM.IRConcrSlab.Save 1212	RobotOM.IRConcrSlabWizard.Finish 1219
RobotOM.IRConcrSlab.StartWizard 1213	RobotOM.IRConcrSlabWizard.Thickness 1217
RobotOM.IRConcrSlab.Steel 1211	RobotOM.IRConcrSteel 1253
RobotOM.IRConcrSlab.UniqueId 1211	RobotOM.IRConcrSteel.AvailableNames 1254
RobotOM.IRConcrSlabCalculationOptions 1213	RobotOM.IRConcrSteel.AvailableStrengths 1254
RobotOM.IRConcrSlabCalculationOptions.Cover 1213	RobotOM.IRConcrSteel.BarDiameters 1254
RobotOM.IRConcrSlabCalculationOptions.CoverLateral 1214	RobotOM.IRConcrSteel.BarsDatabasePath 1255
RobotOM.IRConcrSlabGeometry 1214	RobotOM.IRConcrSteel.CharacteristicStrength 1255
RobotOM.IRConcrSlabGeometry.ConcreteVolume 1215	RobotOM.IRConcrSteel.SelectedName 1255
RobotOM.IRConcrSlabGeometry.ShutteringArea 1215	RobotOM.IRConcrSteel.SurfaceType 1255
RobotOM.IRConcrSlabPatternOptions 1215	RobotOM.IRConcrSteelNames 1251
RobotOM.IRConcrSlabPatternOptions.ReinforcementSegment 1216	RobotOM.IRConcrSteelNames.Count 1252
RobotOM.IRConcrSlabPatternOptions.ReinforcementType 1216	RobotOM.IRConcrSteelNames.Item 1252
RobotOM.IRConcrSlabRequiredReinfCalcParams 1752	RobotOM.IRConcrSteelParams 1703
RobotOM.IRConcrSlabRequiredReinfCalcParams.CasesACC 1753	RobotOM.IRConcrSteelParams.Ey 1704
RobotOM.IRConcrSlabRequiredReinfCalcParams.CasesSLS 1753	RobotOM.IRConcrSteelParams.Ft 1704
RobotOM.IRConcrSlabRequiredReinfCalcParams.CasesULS 1753	RobotOM.IRConcrSteelParams.Ft_calc 1704
RobotOM.IRConcrSlabRequiredReinfCalcParams.DisplayErrors 1754	RobotOM.IRConcrSteelParams.Fy 1704
RobotOM.IRConcrSlabRequiredReinfCalcParams.ForceReduction 1754	RobotOM.IRConcrSteelParams.Fy_calc 1705
RobotOM.IRConcrSlabRequiredReinfCalcParams.GloballyAvgDesginForces 1754	RobotOM.IRConcrSteelParams.Grade 1705
	RobotOM.IRConcrSteelStrengths 1257
	RobotOM.IRConcrSteelSurfaceType 1256
	RobotOM.IRConcrSteelType 1252
	RobotOM.IRConcrWall 1226
	RobotOM.IRConcrWall.Activate 1229

RobotOM.IRConcrWall.Calculate 1229	RobotOM.IRDimCalcStateFlagType 1906
RobotOM.IRConcrWall.Concrete 1227	RobotOM.IRDimCalcStateParamType 1907
RobotOM.IRConcrWall.CreateCalculationNoteRtf 1229	RobotOM.IRDimCalcStateParamValue 1907
RobotOM.IRConcrWall.CreateFromObjects 1230	RobotOM.IRDimCalcStateValueType 1907
RobotOM.IRConcrWall.Geometry 1227	RobotOM.IRDimClient 1925
RobotOM.IRConcrWall.IsSelected 1227	RobotOM.IRDimClient.GetRDimCodeService 1925
RobotOM.IRConcrWall.Name 1228	RobotOM.IRDimCodeResCB71 1855
RobotOM.IRConcrWall.Reinforcement 1228	RobotOM.IRDimCodeResCB71.BadWidthBfUnderFire 1860
RobotOM.IRConcrWall.Save 1230	RobotOM.IRDimCodeResCB71.BuckCoefKMax 1861
RobotOM.IRConcrWall.Steel 1228	RobotOM.IRDimCodeResCB71.BuckCoefKy 1861
RobotOM.IRConcrWall.Uniqueld 1228	RobotOM.IRDimCodeResCB71.BuckCoefKz 1861
RobotOM.IRConcrWall.Verify 1230	RobotOM.IRDimCodeResCB71.BuckLengthComposBeamLey 1861
RobotOM.IRConcrWallGeometry 1230	RobotOM.IRDimCodeResCB71.BuckLengthComposBeamLez 1862
RobotOM.IRConcrWallGeometry.Height 1231	RobotOM.IRDimCodeResCB71.BuckLengthLfy 1862
RobotOM.IRConcrWallGeometry.LeftBoundaryLength 1232	RobotOM.IRDimCodeResCB71.BuckLengthLfz 1862
RobotOM.IRConcrWallGeometry.LeftBoundaryThickness 1232	RobotOM.IRDimCodeResCB71.BuckSlendComposBeamLay m 1862
RobotOM.IRConcrWallGeometry.Length 1232	RobotOM.IRDimCodeResCB71.BuckSlendComposBeamLaz m 1863
RobotOM.IRConcrWallGeometry.Name 1233	RobotOM.IRDimCodeResCB71.BuckSlendLay 1863
RobotOM.IRConcrWallGeometry.Openings 1233	RobotOM.IRDimCodeResCB71.BuckSlendLaz 1863
RobotOM.IRConcrWallGeometry.RightBoundaryLength 1233	RobotOM.IRDimCodeResCB71.CoefDependOnAngle 1863
RobotOM.IRConcrWallGeometry.RightBoundaryThickness 1233	RobotOM.IRDimCodeResCB71.CoefDependOnHeight 1864
RobotOM.IRConcrWallGeometry.Thickness 1234	RobotOM.IRDimCodeResCB71.CoefDependOnHumAndBend 1864
RobotOM.IRConcrWallOpening 1234	RobotOM.IRDimCodeResCB71.CoefDependOnHumAndCom p 1864
RobotOM.IRConcrWallOpening.Lx 1235	RobotOM.IRDimCodeResCB71.CoefDependOnWidth 1864
RobotOM.IRConcrWallOpening.Lz 1235	RobotOM.IRDimCodeResCB71.EfficiencyRatio 1864
RobotOM.IRConcrWallOpening.Name 1235	RobotOM.IRDimCodeResCB71.FireHeightInMidSpanH 1865
RobotOM.IRConcrWallOpening.PosX 1235	RobotOM.IRDimCodeResCB71.FireInerMomInMidSpanly 1865
RobotOM.IRConcrWallOpening.PosZ 1236	RobotOM.IRDimCodeResCB71.FireInerMomInMidSpanlz 1865
RobotOM.IRConcrWallOpenings 1236	RobotOM.IRDimCodeResCB71.FireInerMomlx 1865
RobotOM.IRConcrWallOpenings.Add 1237	RobotOM.IRDimCodeResCB71.FireInerMomy 1866
RobotOM.IRConcrWallOpenings.Count 1237	RobotOM.IRDimCodeResCB71.FireInerMomly 1866
RobotOM.IRConcrWallOpenings.Item 1237	RobotOM.IRDimCodeResCB71.FireInerMomlx 1866
RobotOM.IRDimBuckDiagramCB71 1878	RobotOM.IRDimCodeResCB71.FireLftEdgeDistVpy 1866
RobotOM.IRDimBuckDiagramEC3 1810	RobotOM.IRDimCodeResCB71.FireLowEdgeDistVpz 1866
RobotOM.IRDimCalcState 1907	
RobotOM.IRDimCalcState.GetParam 1908	
RobotOM.IRDimCalcState.GetParamValue 1908	
RobotOM.IRDimCalcState.IsFlagSet 1909	
RobotOM.IRDimCalcState.SetFlag 1909	
RobotOM.IRDimCalcState.SetParam 1909	
RobotOM.IRDimCalcState.SetParamValue 1909	

RobotOM.IRDimCodeResCB71.FireProtectCoefK2 1867	RobotOM.IRDimCodeResCB71.StrsRgtEdgeMZ 1877
RobotOM.IRDimCodeResCB71.FireProtection 1867	RobotOM.IRDimCodeResCB71.StrsShearY 1877
RobotOM.IRDimCodeResCB71.FireProtectionPosition 1867	RobotOM.IRDimCodeResCB71.StrsShearZ 1877
RobotOM.IRDimCodeResCB71.FireRedCoefK1b 1867	RobotOM.IRDimCodeResCB71.StrsTens 1877
RobotOM.IRDimCodeResCB71.FireRedCoefK1h 1868	RobotOM.IRDimCodeResCB71.StrsUprEdgeMY 1878
RobotOM.IRDimCodeResCB71.FireResistance 1868	RobotOM.IRDimCodeResCB71.TimberType 1878
RobotOM.IRDimCodeResCB71.FireRgtEdgeDistVy 1868	RobotOM.IRDimCodeResEC3 1810
RobotOM.IRDimCodeResCB71.FireSecArealnMidSpanS 1868	RobotOM.IRDimCodeResEC3.AxForceExcentrEny 1818
RobotOM.IRDimCodeResCB71.FireSecAreaS 1869	RobotOM.IRDimCodeResEC3.AxForcExcentrEnz 1819
RobotOM.IRDimCodeResCB71.FireSeModulWy 1869	RobotOM.IRDimCodeResEC3.BendParamBetaMltY 1819
RobotOM.IRDimCodeResCB71.FireSeModulWZ 1869	RobotOM.IRDimCodeResEC3.BendParamBetaMltZ 1819
RobotOM.IRDimCodeResCB71.FireShearAreaSY 1869	RobotOM.IRDimCodeResEC3.BendParamBetaMz 1819
RobotOM.IRDimCodeResCB71.FireShearAreaSZ 1870	RobotOM.IRDimCodeResEC3.BendParamBetMy 1820
RobotOM.IRDimCodeResCB71.FireUprEdgeDistVz 1870	RobotOM.IRDimCodeResEC3.BuckCoeffMinXi 1820
RobotOM.IRDimCodeResCB71.FireWidthAtMembEndB 1870	RobotOM.IRDimCodeResEC3.BuckCoeffXy 1820
RobotOM.IRDimCodeResCB71.FireWidthAtMembEndH 1870	RobotOM.IRDimCodeResEC3.BuckCoeffXz 1820
RobotOM.IRDimCodeResCB71.FireWidthB 1871	RobotOM.IRDimCodeResEC3.BuckCurveCoeffAlfy 1821
RobotOM.IRDimCodeResCB71.FireWidthH 1871	RobotOM.IRDimCodeResEC3.BuckCurveCoeffAlfz 1821
RobotOM.IRDimCodeResCB71.FireWidthInMidSpanB 1871	RobotOM.IRDimCodeResEC3.BuckCurveNumbY 1821
RobotOM.IRDimCodeResCB71.IsBuckY 1871	RobotOM.IRDimCodeResEC3.BuckCurveNumbZ 1821
RobotOM.IRDimCodeResCB71.IsBuckZ 1872	RobotOM.IRDimCodeResEC3.BuckParamFiy 1822
RobotOM.IRDimCodeResCB71.LatBuckCoefKinst 1872	RobotOM.IRDimCodeResEC3.BuckParamFiz 1822
RobotOM.IRDimCodeResCB71.LatBuckLengthCoef 1872	RobotOM.IRDimCodeResEC3.BuckParamKy 1822
RobotOM.IRDimCodeResCB71.LatBuckLengthLd 1872	RobotOM.IRDimCodeResEC3.BuckParamKz 1822
RobotOM.IRDimCodeResCB71.LatBuckRelSlendLam 1873	RobotOM.IRDimCodeResEC3.BuckRelSlendLaby 1823
RobotOM.IRDimCodeResCB71.LatBuckStrsCrit 1873	RobotOM.IRDimCodeResEC3.BuckRelSlendLabz 1823
RobotOM.IRDimCodeResCB71.MatAxCompResist 1873	RobotOM.IRDimCodeResEC3.BuckSLendLamy 1823
RobotOM.IRDimCodeResCB71.MatAxTensResist 1873	RobotOM.IRDimCodeResEC3.BuckStrenNbyrd 1823
RobotOM.IRDimCodeResCB71.MatBendResist 1874	RobotOM.IRDimCodeResEC3.BuckStrenNbzrd 1824
RobotOM.IRDimCodeResCB71.MatShearResist 1874	RobotOM.IRDimCodeResEC3.BuclSlendLamz 1824
RobotOM.IRDimCodeResCB71.MatTrCompResist 1874	RobotOM.IRDimCodeResEC3.ClassOfSect 1824
RobotOM.IRDimCodeResCB71.MatTrTensResist 1874	RobotOM.IRDimCodeResEC3.ClassOfSectElem1 1824
RobotOM.IRDimCodeResCB71.StrsBend 1875	RobotOM.IRDimCodeResEC3.ClassOfSectElem2 1825
RobotOM.IRDimCodeResCB71.StrsBendInCompEdgeMY 1875	RobotOM.IRDimCodeResEC3.ClassOfSectElem3 1825
RobotOM.IRDimCodeResCB71.StrsBendInCompEdgeMZ 1875	RobotOM.IRDimCodeResEC3.ClassOfSectElem4 1825
RobotOM.IRDimCodeResCB71.StrsBendInCurveBeam 1875	RobotOM.IRDimCodeResEC3.CompParamBetaA 1825
RobotOM.IRDimCodeResCB71.StrsComp 1876	RobotOM.IRDimCodeResEC3.CritMomenMcr 1826
RobotOM.IRDimCodeResCB71.StrsFinal 1876	RobotOM.IRDimCodeResEC3.EffectiveMomenMeff 1826
RobotOM.IRDimCodeResCB71.StrsLftEdgeMZ 1876	RobotOM.IRDimCodeResEC3.EffSectAreaSeff 1826
RobotOM.IRDimCodeResCB71.StrsLowEdgeMY 1876	RobotOM.IRDimCodeResEC3.EffSectModulWyeff 1826
	RobotOM.IRDimCodeResEC3.EffSectModulWzeff 1827
	RobotOM.IRDimCodeResEC3.ElastMomStrenMelyrd 1827

RobotOM.IRDimCodeResEC3.ElastMomStrenMelzrd 1827	RobotOM.IRDimCodeResEC3.PlastSectModulWyp1 1838
RobotOM.IRDimCodeResEC3.ElastSectModulWyel 1827	RobotOM.IRDimCodeResEC3.PlastSectModulWzpl 1838
RobotOM.IRDimCodeResEC3.ElastSectModulWzel 1828	RobotOM.IRDimCodeResEC3.PlastTensStrenNtrd 1838
RobotOM.IRDimCodeResEC3.FlangeAreaAf 1828	RobotOM.IRDimCodeResEC3.ReducMomStrenMcyrd 1838
RobotOM.IRDimCodeResEC3.FlangeAreaAw 1828	RobotOM.IRDimCodeResEC3.ReducMomStrenMczrd 1839
RobotOM.IRDimCodeResEC3.InteractParamAlfa 1828	RobotOM.IRDimCodeResEC3.ReducMomStrenMnyrd 1839
RobotOM.IRDimCodeResEC3.InteractParamBeta 1829	RobotOM.IRDimCodeResEC3.ReducMomStrenMnzrd 1839
RobotOM.IRDimCodeResEC3.InteractParamMilt 1829	RobotOM.IRDimCodeResEC3.ReducMomStrenMvyrd 1839
RobotOM.IRDimCodeResEC3.InteractParamMiy 1829	RobotOM.IRDimCodeResEC3.ReducMomStrenMvzrd 1840
RobotOM.IRDimCodeResEC3.InteractParamMiz 1829	RobotOM.IRDimCodeResEC3.ShearStrenTplyrd 1840
RobotOM.IRDimCodeResEC3.IsBuckY 1830	RobotOM.IRDimCodeResEC3.ShearStrenTplzrd 1840
RobotOM.IRDimCodeResEC3.IsBuckZ 1830	RobotOM.IRDimCodeResEC3.StrsComp 1840
RobotOM.IRDimCodeResEC3.LaoLevel 1830	RobotOM.IRDimCodeResEC3.StrsLftEdgeMZ 1841
RobotOM.IRDimCodeResEC3.LatBuckCoeffXlt 1830	RobotOM.IRDimCodeResEC3.StrsLowEdgeMY 1841
RobotOM.IRDimCodeResEC3.LatBuckLengthLd 1831	RobotOM.IRDimCodeResEC3.StrsRgtEdgeMZ 1841
RobotOM.IRDimCodeResEC3.LatBuckMomStrenmbrd 1831	RobotOM.IRDimCodeResEC3.StrsShearY 1841
RobotOM.IRDimCodeResEC3.LatBuckParamBetaW 1831	RobotOM.IRDimCodeResEC3.StrsShearZ 1842
RobotOM.IRDimCodeResEC3.LatBuckParamC1 1831	RobotOM.IRDimCodeResEC3.StrsTens 1842
RobotOM.IRDimCodeResEC3.LatBuckParamC2 1832	RobotOM.IRDimCodeResEC3.StrsUprEdgeMY 1842
RobotOM.IRDimCodeResEC3.LatBuckParamFilt 1832	RobotOM.IRDimCodeResEC3.TensStrenNurd 1842
RobotOM.IRDimCodeResEC3.LatBuckParamKlt 1832	RobotOM.IRDimCodeResEC3.TorsMomInertIt 1843
RobotOM.IRDimCodeResEC3.LatBuckParamLd 1832	RobotOM.IRDimCodeResEC3.UppFlanSlend 1843
RobotOM.IRDimCodeResEC3.LatBuckSlendLamIt 1833	RobotOM.IRDimCodeResEC3.WarpingConstantlw 1843
RobotOM.IRDimCodeResEC3.LowFlanSlend1 1833	RobotOM.IRDimCodeResEC3.WebSlend 1843
RobotOM.IRDimCodeResEC3.LowFlanSlend2 1833	RobotOM.IRDimCodeResEC3.WebSlend1 1844
RobotOM.IRDimCodeResEC3.MaEffRatio 1833	RobotOM.IRDimCodeService 1923
RobotOM.IRDimCodeResEC3.MaterCoeffGamma0 1834	RobotOM.IRDimCodeService.EditMembDef 1924
RobotOM.IRDimCodeResEC3.MaterCoeffGamma1 1834	RobotOM.IRDimCodeService.GetDefaultMembDef 1924
RobotOM.IRDimCodeResEC3.MaterCoeffGamma2 1834	RobotOM.IRDimCodeService.GetMembCalc 1925
RobotOM.IRDimCodeResEC3.MaterialCapacityFy 1834	RobotOM.IRDimEffDef 1890
RobotOM.IRDimCodeResEC3.MaxBuckSlend 1835	RobotOM.IRDimEffDef.Clear 1894
RobotOM.IRDimCodeResEC3.MomStrengthMpzEuro 1835	RobotOM.IRDimEffDef.MX 1892
RobotOM.IRDimCodeResEC3.OverallBuckStrenNbld 1835	RobotOM.IRDimEffDef.MY 1892
RobotOM.IRDimCodeResEC3.PartEffRatio1 1835	RobotOM.IRDimEffDef.MZ 1892
RobotOM.IRDimCodeResEC3.PartEffRatio2 1836	RobotOM.IRDimEffDef.N 1892
RobotOM.IRDimCodeResEC3.PartEffRatio3 1836	RobotOM.IRDimEffDef.QY 1893
RobotOM.IRDimCodeResEC3.PartEffRatio4 1836	RobotOM.IRDimEffDef.QZ 1893
RobotOM.IRDimCodeResEC3.PartEffRatio5 1836	RobotOM.IRDimEffDef.Read_IntPsEff_M_1P4L 1894
RobotOM.IRDimCodeResEC3.PlastAxForcStrenNplrd 1837	RobotOM.IRDimEffDef.Read_IntPsEff_M_3P4L 1894
RobotOM.IRDimCodeResEC3.PlastCompStrenNcrd 1837	RobotOM.IRDimEffDef.Read_IntPsEff_M_MID 1895
RobotOM.IRDimCodeResEC3.PlastMomStrenMplyrd 1837	RobotOM.IRDimEffDef.Read_IntPsEff_M1 1895
RobotOM.IRDimCodeResEC3.PlastMomStrenMplzrd 1837	RobotOM.IRDimEffDef.Read_IntPsEff_M12 1895

RobotOM.IRDimEffDef.Read_IntPsEff_M2 1895	RobotOM.IRDimMatDefLongExValType 1900
RobotOM.IRDimEffDef.Read_IntPsEff_MN_MAX 1895	RobotOM.IRDimMatDefType 1899
RobotOM.IRDimEffDef.Read_IntPsEff_MP_MAX 1896	RobotOM.IRDimMatDefValType 1899
RobotOM.IRDimEffDef.Read_M_1P4L 1896	RobotOM.IRDimMembCalc 1919
RobotOM.IRDimEffDef.Read_M_3P4L 1896	RobotOM.IRDimMembCalc.CalculBuckling 1922
RobotOM.IRDimEffDef.Read_M_MID 1896	RobotOM.IRDimMembCalc.CalculMember 1922
RobotOM.IRDimEffDef.Read_M1 1896	RobotOM.IRDimMembCalc.GetRatio 1922
RobotOM.IRDimEffDef.Read_M12 1897	RobotOM.IRDimMembCalc.GetResultsInterface 1922
RobotOM.IRDimEffDef.Read_M2 1897	RobotOM.IRDimMembCalc.IsExtraMomentsSet1 1920
RobotOM.IRDimEffDef.Read_MN_MAX 1897	RobotOM.IRDimMembCalc.IsExtraMomentsSet2 1921
RobotOM.IRDimEffDef.Read_MP_MAX 1897	RobotOM.IRDimMembCalc.IsIntPointsMomentsSet1 1921
RobotOM.IRDimEffDef.ReadParam 1897	RobotOM.IRDimMembCalc.IsIntPointsMomentsSet2 1921
RobotOM.IRDimEffDef.WriteForces 1898	RobotOM.IRDimMembCalc.SetCalcState 1922
RobotOM.IRDimEffDef.WriteLinePsEffSet1 1898	RobotOM.IRDimMembCalc.SetEffDef 1923
RobotOM.IRDimEffDef.WriteLinePsEffSet2 1898	RobotOM.IRDimMembCalc.SetMatDef 1923
RobotOM.IRDimEffDef.WriteParam 1898	RobotOM.IRDimMembCalc.SetMembDef 1923
RobotOM.IRDimEffDef.WriteValuesSet1 1899	RobotOM.IRDimMembCalc.SetProfDef 1923
RobotOM.IRDimEffDef.WriteValuesSet2 1899	RobotOM.IRDimMembCalcBuckType 1911
RobotOM.IRDimEffDefDirType 1890	RobotOM.IRDimMembCalcRetValue 1910
RobotOM.IRDimEffDefIntPsType 1890	RobotOM.IRDimMembDef 1885
RobotOM.IRDimEffDefParamType 1889	RobotOM.IRDimMembDef.ClientID 1886
RobotOM.IRDimFireMemberPositionsCB71 1880	RobotOM.IRDimMembDef.DefYZRelLimit 1888
RobotOM.IRDimFireProtectionTimesCB71 1880	RobotOM.IRDimMembDef.DisplXYRelLimit 1888
RobotOM.IRDimLatBuckCoeffDiagramEC3 1809	RobotOM.IRDimMembDef.IsBuckCoefConst 1888
RobotOM.IRDimLaterBuckTypeCB71 1879	RobotOM.IRDimMembDef.IsDeflectionYZ 1888
RobotOM.IRDimLaterBuckTypeEC3 1809	RobotOM.IRDimMembDef.IsDisplacementXY 1888
RobotOM.IRDimLoadLevelCB71 1879	RobotOM.IRDimMembDef.Length 1886
RobotOM.IRDimLoadLevelEC3 1809	RobotOM.IRDimMembDef.LengthYZUV 1889
RobotOM.IRDimLoadTypeCB71 1879	RobotOM.IRDimMembDef.MatType 1887
RobotOM.IRDimLoadTypeEC3 1808	RobotOM.IRDimMembDef.Name 1887
RobotOM.IRDimMatDef 1900	RobotOM.IRDimMembDef.Retrieve 1889
RobotOM.IRDimMatDef.Name 1901	RobotOM.IRDimMembDef.Store 1889
RobotOM.IRDimMatDef.ReadDoubleExtraValue 1902	RobotOM.IRDimMembDef.Type 1887
RobotOM.IRDimMatDef.ReadLongExtraValue 1902	RobotOM.IRDimMembDefBucklingDataType 1885
RobotOM.IRDimMatDef.ReadValue 1902	RobotOM.IRDimMembDefDeflDataType 1885
RobotOM.IRDimMatDef.SecondName 1901	RobotOM.IRDimMembDefDispDataType 1885
RobotOM.IRDimMatDef.SetNames 1903	RobotOM.IRDimMembDefLengthDataType 1884
RobotOM.IRDimMatDef.Type 1901	RobotOM.IRDimMembDefMatType 1884
RobotOM.IRDimMatDef.WriteDoubleExtraValue 1903	RobotOM.IRDimMembDefType 1884
RobotOM.IRDimMatDef.WriteLineExtraValue 1903	RobotOM.IRDimMembParamsCB71 1845
RobotOM.IRDimMatDef.writeValue 1903	RobotOM.IRDimMembParamsCB71.ArcBeamCheck 1847
RobotOM.IRDimMatDefDblExValType 1900	RobotOM.IRDimMembParamsCB71.ArcRadius 1848

RobotOM.IRDimMembParamsCB71.BuckComposBeamCoefCey 1848	RobotOM.IRDimMembParamsEC3.LatCoeffLowerFlangeValue 1804
RobotOM.IRDimMembParamsCB71.BuckComposBeamCoefCez 1848	RobotOM.IRDimMembParamsEC3.LatCoeffUpperFlangeValue 1804
RobotOM.IRDimMembParamsCB71.BuckIsComposedY 1848	RobotOM.IRDimMembParamsEC3.LatCoeffUpperFlangeValue 1805
RobotOM.IRDimMembParamsCB71.BuckIsComposedZ 1849	RobotOM.IRDimMembParamsEC3.LoadLevel 1805
RobotOM.IRDimMembParamsCB71.BuckLengthCoeffZ 1849	RobotOM.IRDimMembParamsEC3.LoadLevelValue 1805
RobotOM.IRDimMembParamsCB71.BucklingDiagramY 1849	RobotOM.IRDimMembParamsEC3.LoadTypeY 1805
RobotOM.IRDimMembParamsCB71.BucklingDiagramZ 1849	RobotOM.IRDimMembParamsEC3.LoadTypeZ 1806
RobotOM.IRDimMembParamsCB71.BuckLengthCoeffY 1850	RobotOM.IRDimMembParamsEC3.MaterCoeffGamma0 1806
RobotOM.IRDimMembParamsCB71.FireContinProtForCompo sBeam 1850	RobotOM.IRDimMembParamsEC3.MaterCoeffGamma1 1806
RobotOM.IRDimMembParamsCB71.FireLftSideProt 1850	RobotOM.IRDimMembParamsEC3.RelLimitDeflUy 1806
RobotOM.IRDimMembParamsCB71.FireLowSideProt 1850	RobotOM.IRDimMembParamsEC3.RelLimitdeflUz 1807
RobotOM.IRDimMembParamsCB71.FireMembPosition 1851	RobotOM.IRDimMembParamsEC3.TensAreaNetGros 1807
RobotOM.IRDimMembParamsCB71.FireProtectionTime 1851	RobotOM.IRDimMembParamsEC3.TubeControl 1807
RobotOM.IRDimMembParamsCB71.FireRequiredResist 1851	RobotOM.IRDimMembParamsEC3.YieldStrengthType 1808
RobotOM.IRDimMembParamsCB71.FireRgtSideProt 1851	RobotOM.IRDimMembParamsEC3.YieldStrengthValue 1808
RobotOM.IRDimMembParamsCB71.FireUprSideProt 1852	RobotOM.IRDimMembRes 1911
RobotOM.IRDimMembParamsCB71.ForceDistFromBeginDist X 1852	RobotOM.IRDimMembRes.BlockCount 1915
RobotOM.IRDimMembParamsCB71.HumChangeDeltaH 1852	RobotOM.IRDimMembRes.CreateResWnd 1916
RobotOM.IRDimMembParamsCB71.InflDeflCoefTheta 1853	RobotOM.IRDimMembRes.EffRatio 1913
RobotOM.IRDimMembParamsCB71.LatBuckType 1853	RobotOM.IRDimMembRes.GetEffDefAccess 1916
RobotOM.IRDimMembParamsCB71.LatCoeffLowerFlangeVal ue 1853	RobotOM.IRDimMembRes.GetLineComponent 1916
RobotOM.IRDimMembParamsCB71.LatCoeffUpperFlangeVal ue 1853	RobotOM.IRDimMembRes.GetLineType 1916
RobotOM.IRDimMembParamsCB71.LoadLevel 1854	RobotOM.IRDimMembRes.GetMatDefAccess 1916
RobotOM.IRDimMembParamsCB71.LoadTypeY 1854	RobotOM.IRDimMembRes.GetMaxLineNo 1917
RobotOM.IRDimMembParamsCB71.TensAreaNetGros 1854	RobotOM.IRDimMembRes.GetMembDefAccess 1917
RobotOM.IRDimMembParamsCB71.TubeControl 1854	RobotOM.IRDimMembRes.GetProfDefAccess 1917
RobotOM.IRDimMembParamsEC3 1800	RobotOM.IRDimMembRes.IsLineActive 1917
RobotOM.IRDimMembParamsEC3.BuckLengthCoeffY 1802	RobotOM.IRDimMembRes.IsStatement 1917
RobotOM.IRDimMembParamsEC3.BuckLengthCoeffZ 1803	RobotOM.IRDimMembRes.Language 1913
RobotOM.IRDimMembParamsEC3.BucklingDiagramY 1803	RobotOM.IRDimMembRes.Ratio 1913
RobotOM.IRDimMembParamsEC3.BucklingDiagramZ 1803	RobotOM.IRDimMembRes.RecognizedPQ 1918
RobotOM.IRDimMembParamsEC3.HotRolledPipes 1803	RobotOM.IRDimMembRes.RefreshUnits 1918
RobotOM.IRDimMembParamsEC3.LatBuckType 1804	RobotOM.IRDimMembRes.ReplaceMark 1918
RobotOM.IRDimMembParamsEC3.LatCoeffLowerFlange 1804	RobotOM.IRDimMembRes.ResOfCalc 1914
	RobotOM.IRDimMembRes.Retrieve 1918
	RobotOM.IRDimMembRes.RtfFileName 1914
	RobotOM.IRDimMembRes.SetEffDef 1918
	RobotOM.IRDimMembRes.SetMatDef 1919
	RobotOM.IRDimMembRes.SetMembDef 1919

RobotOM.IRDimMembRes.SetProfDef 1919	RobotOM.IRJointAnchor.Tige 1995
RobotOM.IRDimMembRes.SlendY 1914	RobotOM.IRJointAnchor.Type 1995
RobotOM.IRDimMembRes.SlendZ 1914	RobotOM.IRJointAnchorBolt 1991
RobotOM.IRDimMembRes.Units 1914	RobotOM.IRJointAnchorBolt.Length1 1991
RobotOM.IRDimMembResTableComp 1911	RobotOM.IRJointAnchorBolt.Length2 1992
RobotOM.IRDimMembResTableLineType 1911	RobotOM.IRJointAnchorBolt.Length3 1992
RobotOM.IRDimMembSrv 1909	RobotOM.IRJointAnchorBolt.Length4 1992
RobotOM.IRDimMembSrv.CheckLabelName 1910	RobotOM.IRJointAnchorPlate 1992
RobotOM.IRDimMembSrv.Save 1910	RobotOM.IRJointAnchorPlate.Type 1993
RobotOM.IRDimProfDef 1904	RobotOM.IRJointAnchorPlateType 2009
RobotOM.IRDimProfDef.Clear 1906	RobotOM.IRJointAnchorType 1993
RobotOM.IRDimProfDef.IsVar 1905	RobotOM.IRJointAngle 2023
RobotOM.IRDimProfDef.Name 1905	RobotOM.IRJointAngle.DistFromUpperBeamEdge 2024
RobotOM.IRDimProfDef.ReadValue 1906	RobotOM.IRJointAngle.Length 2024
RobotOM.IRDimProfDef.Type 1905	RobotOM.IRJointAngle.Profile 2024
RobotOM.IRDimProfDef.writeValue 1906	RobotOM.IRJointAngle.ProfilePosition 2025
RobotOM.IRDimProfDefItem.Type 1904	RobotOM.IRJointAngle.Type 2025
RobotOM.IRDimProfDefType 1903	RobotOM.IRJointAngleBolts 2020
RobotOM.IRDimProfDefVal.Type 1904	RobotOM.IRJointAngleBolts.Area 2021
RobotOM.IRDimServer 1928	RobotOM.IRJointAngleBolts.ClassName 2021
RobotOM.IRDimStream 1881	RobotOM.IRJointAngleBolts.Diameter 2021
RobotOM.IRDimStream.Clear 1882	RobotOM.IRJointAngleBolts.DiameterName 2021
RobotOM.IRDimStream.ReadDouble 1882	RobotOM.IRJointAngleBolts.DistFromUpperEdge 2022
RobotOM.IRDimStream.ReadLong 1882	RobotOM.IRJointAngleBolts.DistFromWeb 2022
RobotOM.IRDimStream.ReadText 1883	RobotOM.IRJointAngleBolts.Friction 2022
RobotOM.IRDimStream.SeekSet 1883	RobotOM.IRJointAngleBolts.Rows 2022
RobotOM.IRDimStream.Size 1883	RobotOM.IRJointAngleBolts.Spacing 2023
RobotOM.IRDimStream.WriteDouble 1883	RobotOM.IRJointAngleLoad 2030
RobotOM.IRDimStream.WriteLine 1883	RobotOM.IRJointAngleLoad.Fz 2031
RobotOM.IRDimStream.WriteString 1884	RobotOM.IRJointAngleProfilePosition 2030
RobotOM.IRDimStreamType 1881	RobotOM.IRJointAngleType 2025
RobotOM.IRDimUnits 1926	RobotOM.IRJointAngleUnit 2167
RobotOM.IRDimUnits.AreRobotUnits 1927	RobotOM.IRJointBeamCut 2025
RobotOM.IRDimUnits.Format 1927	RobotOM.IRJointBeamCut.HeightLower 2026
RobotOM.IRDimUnits.ReadToUserCoef 1927	RobotOM.IRJointBeamCut.HeightUpper 2026
RobotOM.IRDimUnits.ReadUserName 1928	RobotOM.IRJointBeamCut.Length 2027
RobotOM.IRDimUnits.Refresh 1928	RobotOM.IRJointBeamGirder 2127
RobotOM.IRDimUnitType 1926	RobotOM.IRJointBeamGirder.AngleLeft 2129
RobotOM.IRDimYieldStrengthTypeEC3 1844	RobotOM.IRJointBeamGirder.AngleRight 2130
RobotOM.IRJointAnchor 1994	RobotOM.IRJointBeamGirder.BeamLeft 2130
RobotOM.IRJointAnchor.AnchorPlate 1995	RobotOM.IRJointBeamGirder.BeamRight 2130
RobotOM.IRJointAnchor.Bolts 1995	RobotOM.IRJointBeamGirder.DistGirderFlangeToBeamFlangeLeft

2130	RobotOM.IRJointBeamGirderPlate 2120
RobotOM.IRJointBeamGirder.DistGirderFlangeToBeamFlangeRight 2131	RobotOM.IRJointBeamGirderPlate.BoltsBeam 2121
RobotOM.IRJointBeamGirder.DistGirderToBeamLeft 2131	RobotOM.IRJointBeamGirderPlate.ConnectorsType 2122
RobotOM.IRJointBeamGirder.DistGirderToBeamRight 2131	RobotOM.IRJointBeamGirderPlate.DistFromUpperBeamEdge 2122
RobotOM.IRJointBeamGirder.ElementType 2131	RobotOM.IRJointBeamGirderPlate.Length 2122
RobotOM.IRJointBeamGirder.Girder 2132	RobotOM.IRJointBeamGirderPlate.Material 2123
RobotOM.IRJointBeamGirder.PlateLeft 2132	RobotOM.IRJointBeamGirderPlate.Thick 2123
RobotOM.IRJointBeamGirder.PlateRight 2132	RobotOM.IRJointBeamGirderPlate.WeldsBeam 2123
RobotOM.IRJointBeamGirder.SeatLeftDown 2133	RobotOM.IRJointBeamGirderPlate.WeldsGirder 2124
RobotOM.IRJointBeamGirder.SeatLeftUp 2133	RobotOM.IRJointBeamGirderPlate.Width 2124
RobotOM.IRJointBeamGirder.SeatRightDown 2133	RobotOM.IRJointBeamGirderSeat 2124
RobotOM.IRJointBeamGirder.SeatRightUp 2133	RobotOM.IRJointBeamGirderSeat.BoltsBeam 2125
RobotOM.IRJointBeamGirder.StiffenerLeft 2134	RobotOM.IRJointBeamGirderSeat.BoltsGirder 2126
RobotOM.IRJointBeamGirder.StiffenerRight 2134	RobotOM.IRJointBeamGirderSeat.ConnectorsToWebType 2126
RobotOM.IRJointBeamGirderAngle 2141	RobotOM.IRJointBeamGirderSeat.Length 2126
RobotOM.IRJointBeamGirderAngle.BoltsBeam 2142	RobotOM.IRJointBeamGirderSeat.Material 2126
RobotOM.IRJointBeamGirderAngle.BoltsGirder 2142	RobotOM.IRJointBeamGirderSeat.Section 2127
RobotOM.IRJointBeamGirderAngle.Element 2142	RobotOM.IRJointBeamGirderSeat.WeldsGirder 2127
RobotOM.IRJointBeamGirderBeam 2143	RobotOM.IRJointBeamGirderSeatBolts 2137
RobotOM.IRJointBeamGirderBeam.CutEnd 2143	RobotOM.IRJointBeamGirderSeatBolts.Area 2138
RobotOM.IRJointBeamGirderBeam.Profile 2144	RobotOM.IRJointBeamGirderSeatBolts.ClassName 2138
RobotOM.IRJointBeamGirderBolts 2112	RobotOM.IRJointBeamGirderSeatBolts.Cols 2138
RobotOM.IRJointBeamGirderBolts.Area 2113	RobotOM.IRJointBeamGirderSeatBolts.Diameter 2139
RobotOM.IRJointBeamGirderBolts.ClassName 2113	RobotOM.IRJointBeamGirderSeatBolts.DiameterName 2139
RobotOM.IRJointBeamGirderBolts.Cols 2114	RobotOM.IRJointBeamGirderSeatBolts.DistFromPerpendicular ArmEdge 2139
RobotOM.IRJointBeamGirderBolts.Diameter 2114	RobotOM.IRJointBeamGirderSeatBolts.DistFromVertEdge 2139
RobotOM.IRJointBeamGirderBolts.DiameterName 2114	RobotOM.IRJointBeamGirderSeatBolts.Friction 2140
RobotOM.IRJointBeamGirderBolts.DistFromUpperBeamEdge 2115	RobotOM.IRJointBeamGirderStiffener 2116
RobotOM.IRJointBeamGirderBolts.DistFromVertBeamEdge 2115	RobotOM.IRJointBeamGirderStiffener.BoltsBeam 2118
RobotOM.IRJointBeamGirderBolts.Friction 2115	RobotOM.IRJointBeamGirderStiffener.ConnectorsType 2118
RobotOM.IRJointBeamGirderBolts.Rows 2115	RobotOM.IRJointBeamGirderStiffener.CutHeightDown 2118
RobotOM.IRJointBeamGirderBolts.SpacingH 2116	RobotOM.IRJointBeamGirderStiffener.CutHeightUp 2118
RobotOM.IRJointBeamGirderBolts.SpacingV 2116	RobotOM.IRJointBeamGirderStiffener.CutLength 2119
RobotOM.IRJointBeamGirderLoad 2134	RobotOM.IRJointBeamGirderStiffener.Length 2119
RobotOM.IRJointBeamGirderLoad.LFx 2135	RobotOM.IRJointBeamGirderStiffener.Material 2119
RobotOM.IRJointBeamGirderLoad.LFz 2135	RobotOM.IRJointBeamGirderStiffener.Thick 2120
RobotOM.IRJointBeamGirderLoad.LMy 2136	RobotOM.IRJointBeamGirderStiffener.WeldsBeam 2120
RobotOM.IRJointBeamGirderLoad.RFx 2136	RobotOM.IRJointBearingPlate 2003
RobotOM.IRJointBeamGirderLoad.RFz 2136	RobotOM.IRJointBearingPlate.ThickBearingBar 2004
RobotOM.IRJointBeamGirderLoad.RMy 2137	

RobotOM.IRJointBolts 2153	RobotOM.IRJointConcreteColumnFoundation.par_o1 1987
RobotOM.IRJointBolts.Area 2154	RobotOM.IRJointConcreteColumnFoundation.par_o2 1988
RobotOM.IRJointBolts.ClassName 2155	RobotOM.IRJointConcreteColumnLoad 1988
RobotOM.IRJointBolts.Cols 2155	RobotOM.IRJointConcreteColumnLoad.Fx 1989
RobotOM.IRJointBolts.Diameter 2155	RobotOM.IRJointConcreteColumnLoad.FxAssemb 1989
RobotOM.IRJointBolts.DiameterName 2155	RobotOM.IRJointConcreteColumnLoad.Fy 1990
RobotOM.IRJointBolts.Friction 2156	RobotOM.IRJointConcreteColumnLoad.Fz 1990
RobotOM.IRJointBolts.Height1 2156	RobotOM.IRJointConcreteColumnLoad.My 1990
RobotOM.IRJointBolts.Rows 2156	RobotOM.IRJointConcreteColumnLoad.Mz 1990
RobotOM.IRJointBoltType 2157	RobotOM.IRJointConcreteMaterials 1979
RobotOM.IRJointColumnBasePlateCalcModel 1988	RobotOM.IRJointConcreteMaterials.CoeffBA 1980
RobotOM.IRJointColumnBracket 2006	RobotOM.IRJointConcreteMaterials.Dosage 1980
RobotOM.IRJointColumnBracket.Exist 2009	RobotOM.IRJointConcreteMaterials.Sigma 1980
RobotOM.IRJointColumnBracket.Height 2007	RobotOM.IRJointConnection 2147
RobotOM.IRJointColumnBracket.Length 2008	RobotOM.IRJointConnection.GetFromRobot 2149
RobotOM.IRJointColumnBracket.Thick 2008	RobotOM.IRJointConnection.SetToRobot 2149
RobotOM.IRJointColumnBracket.VThick 2008	RobotOM.IRJointConnection.Type 2148
RobotOM.IRJointColumnBracket.Width 2008	RobotOM.IRJointConnection.WFRelPos 2148
RobotOM.IRJointColumnSquare 2014	RobotOM.IRJointConnectionDefType 2144
RobotOM.IRJointColumnSquare.Diameter 2014	RobotOM.IRJointConnectionInfo 2145
RobotOM.IRJointColumnSquare.Length 2015	RobotOM.IRJointConnectionInfo.DefType 2146
RobotOM.IRJointColumnSquare.Profile 2015	RobotOM.IRJointConnectionInfo.Elements 2146
RobotOM.IRJointComponentType 2168	RobotOM.IRJointConnectionInfo.Node 2146
RobotOM.IRJointConcreteColumn 1981	RobotOM.IRJointConnectionInfo.Number 2146
RobotOM.IRJointConcreteColumn.Base 1982	RobotOM.IRJointConnectionInfo.Type 2147
RobotOM.IRJointConcreteColumn.CalcModel 1982	RobotOM.IRJointConnectionInfo.UniqueId 2147
RobotOM.IRJointConcreteColumn.Depth 1982	RobotOM.IRJointConnectionServer 2157
RobotOM.IRJointConcreteColumn.Foundation 1982	RobotOM.IRJointConnectionServer.Calculate 2159
RobotOM.IRJointConcreteColumn.Materials 1983	RobotOM.IRJointConnectionServer.CalculateNote 2159
RobotOM.IRJointConcreteColumn.Profile 1983	RobotOM.IRJointConnectionServer.Count 2158
RobotOM.IRJointConcreteColumn.SpreadFootingType 1983	RobotOM.IRJointConnectionServer.Create 2160
RobotOM.IRJointConcreteColumnFoundation 1984	RobotOM.IRJointConnectionServer.CreateInfo 2160
RobotOM.IRJointConcreteColumnFoundation.par_a1 1985	RobotOM.IRJointConnectionServer.Delete 2160
RobotOM.IRJointConcreteColumnFoundation.par_b 1985	RobotOM.IRJointConnectionServer.Exist 2160
RobotOM.IRJointConcreteColumnFoundation.par_b1 1985	RobotOM.IRJointConnectionServer.Find 2161
RobotOM.IRJointConcreteColumnFoundation.par_b2 1986	RobotOM.IRJointConnectionServer.FindWithId 2161
RobotOM.IRJointConcreteColumnFoundation.par_h 1986	RobotOM.IRJointConnectionServer.Get 2161
RobotOM.IRJointConcreteColumnFoundation.par_h1 1986	RobotOM.IRJointConnectionServer.GetAllNumbers 2162
RobotOM.IRJointConcreteColumnFoundation.par_h2 1986	RobotOM.IRJointConnectionServer.GetInfo 2162
RobotOM.IRJointConcreteColumnFoundation.par_l 1987	RobotOM.IRJointConnectionType 2144
RobotOM.IRJointConcreteColumnFoundation.par_l1 1987	RobotOM.IRJointConnectorsType 2140
RobotOM.IRJointConcreteColumnFoundation.par_l2 1987	RobotOM.IRJointElementType 2140

RobotOM.IRJointExtType 2167	RobotOM.IRJointFootMaterials.PlateSigma 2013
RobotOM.IRJointFixedColumnBase 1972	RobotOM.IRJointFootMaterials.PlateYoung 2013
RobotOM.IRJointFixedColumnBase.Anchor 1973	RobotOM.IRJointFootPlate 1999
RobotOM.IRJointFixedColumnBase.Base 1974	RobotOM.IRJointFootPlate.Diameter 2000
RobotOM.IRJointFixedColumnBase.BasePlateMaterial 1974	RobotOM.IRJointFootPlate.Type 2000
RobotOM.IRJointFixedColumnBase.Bolts 1974	RobotOM.IRJointFootPlateType 1999
RobotOM.IRJointFixedColumnBase.ComplexStiff 1974	RobotOM.IRJointFootStiffenerComplex 2017
RobotOM.IRJointFixedColumnBase.FootPlate 1975	RobotOM.IRJointFootStiffenerComplex.Height 2018
RobotOM.IRJointFixedColumnBase.Materials 1975	RobotOM.IRJointFootStiffenerComplex.Length 2018
RobotOM.IRJointFixedColumnBase.NodeNumber 1975	RobotOM.IRJointFootStiffenerComplex.ThickPlateHor 2018
RobotOM.IRJointFixedColumnBase.Profile 1975	RobotOM.IRJointFootStiffenerComplex.ThickStiff 2019
RobotOM.IRJointFixedColumnBase.SimpleStiff 1976	RobotOM.IRJointFootStiffenerHoriz 2011
RobotOM.IRJointFixedColumnBase.StiffType 1976	RobotOM.IRJointFootStiffenerHoriz.Thick 2011
RobotOM.IRJointFixedColumnBase.Washer 1976	RobotOM.IRJointFootStiffenerSimple 2015
RobotOM.IRJointFixedColumnBase.Wedge 1976	RobotOM.IRJointFootStiffenerSimple.Height 2016
RobotOM.IRJointFixedColumnBase.Welds 1977	RobotOM.IRJointFootStiffenerSimple.Length 2016
RobotOM.IRJointFixedColumnBaseStiffType 1979	RobotOM.IRJointFootStiffenerSimple.Thick 2016
RobotOM.IRJointFixedFootWelds 1977	RobotOM.IRJointFootStiffenerSimple.Type 2017
RobotOM.IRJointFixedFootWelds.FootPlate 1978	RobotOM.IRJointFootStiffenerSimple.Width 2017
RobotOM.IRJointFixedFootWelds.Stiff 1978	RobotOM.IRJointFootStiffenerVert 2009
RobotOM.IRJointFixedFootWelds.Washer 1978	RobotOM.IRJointFootStiffenerVert.Length 2010
RobotOM.IRJointFixedFootWelds.Wedge 1978	RobotOM.IRJointFootStiffenerVert.Thick 2010
RobotOM.IRJointFixedLoad 1970	RobotOM.IRJointFootStiffenerVert.WidthSpacing 2011
RobotOM.IRJointFixedLoad.Fx 1971	RobotOM.IRJointFootStiffType 2019
RobotOM.IRJointFixedLoad.Fy 1971	RobotOM.IRJointFootWelds 2004
RobotOM.IRJointFixedLoad.Fz 1971	RobotOM.IRJointFootWelds.Bearing 2005
RobotOM.IRJointFixedLoad.My 1972	RobotOM.IRJointFootWelds.FootPlate 2005
RobotOM.IRJointFixedLoad.Mz 1972	RobotOM.IRJointFootWelds.Stiff 2006
RobotOM.IRJointFootBolts 1996	RobotOM.IRJointFootWelds.Washer 2006
RobotOM.IRJointFootBolts.Area 1997	RobotOM.IRJointFootWelds.Wedge 2006
RobotOM.IRJointFootBolts.ClassName 1997	RobotOM.IRJointGussetBoltsDiag 2049
RobotOM.IRJointFootBolts.Diameter 1997	RobotOM.IRJointGussetBoltsDiag.BoltAxisShift 2050
RobotOM.IRJointFootBolts.DiameterName 1997	RobotOM.IRJointGussetBoltsDiag.ClassName 2050
RobotOM.IRJointFootBolts.Distance 1998	RobotOM.IRJointGussetBoltsDiag.Diameter 2051
RobotOM.IRJointFootBolts.Friction 1998	RobotOM.IRJointGussetBoltsDiag.DistanceH1 2051
RobotOM.IRJointFootBolts.Rows 1998	RobotOM.IRJointGussetBoltsDiag.Friction 2051
RobotOM.IRJointFootBolts.SpacingH 1998	RobotOM.IRJointGussetBoltsDiag.Rows 2052
RobotOM.IRJointFootBolts.SpacingV 1999	RobotOM.IRJointGussetBoltsDiag.Spacing 2052
RobotOM.IRJointFootMaterials 2012	RobotOM.IRJointGussetCornersType 2067
RobotOM.IRJointFootMaterials.CementAmount 2012	RobotOM.IRJointGussetCross 2084
RobotOM.IRJointFootMaterials.ConcrClass 2013	RobotOM.IRJointGussetCross.BoltsClassVector 2086
RobotOM.IRJointFootMaterials.ConcrSteelCoeff 2013	RobotOM.IRJointGussetCross.BoltsDimensionNamesVector 2086

RobotOM.IRJointGussetCross.BoltsDimensionsVector 2087	RobotOM.IRJointGussetCrossPlate.VOffset 2084
RobotOM.IRJointGussetCross.DiagLeftLower 2087	RobotOM.IRJointGussetCrossProfileCutting 2075
RobotOM.IRJointGussetCross.DiagLeftUpper 2087	RobotOM.IRJointGussetDiagonale 2054
RobotOM.IRJointGussetCross.DiagRightLower 2087	RobotOM.IRJointGussetDiagonale.BarNumber 2055
RobotOM.IRJointGussetCross.DiagRightUpper 2088	RobotOM.IRJointGussetDiagonale.BoltsDiag 2055
RobotOM.IRJointGussetCross.FixType_LeftL_RightU 2088	RobotOM.IRJointGussetDiagonale.DistanceEC 2055
RobotOM.IRJointGussetCross.FixType_LeftLower 2088	RobotOM.IRJointGussetDiagonale.Exist 2056
RobotOM.IRJointGussetCross.FixType_LeftU_RightL 2088	RobotOM.IRJointGussetDiagonale.Position 2056
RobotOM.IRJointGussetCross.FixType_LeftUpper 2089	RobotOM.IRJointGussetDiagonale.Profile 2056
RobotOM.IRJointGussetCross.FixType_RightLower 2089	RobotOM.IRJointGussetDiagonale.WeldsDiag 2056
RobotOM.IRJointGussetCross.FixType_RightUpper 2089	RobotOM.IRJointGussetDiagonalePositionType 2111
RobotOM.IRJointGussetCross.GussetPlate 2090	RobotOM.IRJointGussetFixType 2057
RobotOM.IRJointGussetCross.ProfileCutting 2090	RobotOM.IRJointGussetFlange 2100
RobotOM.IRJointGussetCrossLoad 2107	RobotOM.IRJointGussetFlange.BoltsClassVector 2102
RobotOM.IRJointGussetCrossLoad.Fx_LeftLower 2107	RobotOM.IRJointGussetFlange.BoltsDimensionNamesVector 2102
RobotOM.IRJointGussetCrossLoad.Fx_LeftUpper 2108	RobotOM.IRJointGussetFlange.BoltsDimensionsVector 2102
RobotOM.IRJointGussetCrossLoad.Fx_RightLower 2108	RobotOM.IRJointGussetFlange.DiagonalLeft 2103
RobotOM.IRJointGussetCrossLoad.Fx_RightUpper 2108	RobotOM.IRJointGussetFlange.DiagonalRight 2103
RobotOM.IRJointGussetCrossPlate 2076	RobotOM.IRJointGussetFlange.DiagonalUp 2103
RobotOM.IRJointGussetCrossPlate.B1 2077	RobotOM.IRJointGussetFlange.FixType_DiagonalLeft 2103
RobotOM.IRJointGussetCrossPlate.B2 2077	RobotOM.IRJointGussetFlange.FixType_DiagonalRight 2104
RobotOM.IRJointGussetCrossPlate.B3 2078	RobotOM.IRJointGussetFlange.FixType_DiagonalUp 2104
RobotOM.IRJointGussetCrossPlate.B4 2078	RobotOM.IRJointGussetFlange.FixType_FlangeLeft 2104
RobotOM.IRJointGussetCrossPlate.CornersType 2078	RobotOM.IRJointGussetFlange.FixType_FlangeRight 2104
RobotOM.IRJointGussetCrossPlate.CutH1 2078	RobotOM.IRJointGussetFlange.FlangeLeft 2105
RobotOM.IRJointGussetCrossPlate.CutH2 2079	RobotOM.IRJointGussetFlange.FlangeRight 2105
RobotOM.IRJointGussetCrossPlate.CutH3 2079	RobotOM.IRJointGussetFlange.GussetPlate 2105
RobotOM.IRJointGussetCrossPlate.CutH4 2079	RobotOM.IRJointGussetFlange.ProfileCutting 2105
RobotOM.IRJointGussetCrossPlate.CutV1 2080	RobotOM.IRJointGussetFlangeLoad 2108
RobotOM.IRJointGussetCrossPlate.CutV2 2080	RobotOM.IRJointGussetFlangeLoad.Fx_FlangeLeft 2109
RobotOM.IRJointGussetCrossPlate.CutV3 2080	RobotOM.IRJointGussetFlangeLoad.Fx_FlangeRight 2110
RobotOM.IRJointGussetCrossPlate.CutV4 2081	RobotOM.IRJointGussetFlangeLoad.Fx_LeftUpper 2110
RobotOM.IRJointGussetCrossPlate.H1 2081	RobotOM.IRJointGussetFlangeLoad.Fx_RightUpper 2110
RobotOM.IRJointGussetCrossPlate.H2 2081	RobotOM.IRJointGussetFlangeLoad.Fx_Upper 2110
RobotOM.IRJointGussetCrossPlate.H3 2082	RobotOM.IRJointGussetFlangePlate 2090
RobotOM.IRJointGussetCrossPlate.H4 2082	RobotOM.IRJointGussetFlangePlate.B1 2092
RobotOM.IRJointGussetCrossPlate.Height 2082	RobotOM.IRJointGussetFlangePlate.B2 2092
RobotOM.IRJointGussetCrossPlate.HOffset 2082	RobotOM.IRJointGussetFlangePlate.B3 2093
RobotOM.IRJointGussetCrossPlate.HorizontalOffset 2083	RobotOM.IRJointGussetFlangePlate.B4 2093
RobotOM.IRJointGussetCrossPlate.Length 2083	RobotOM.IRJointGussetFlangePlate.CornersType 2093
RobotOM.IRJointGussetCrossPlate.Material 2083	RobotOM.IRJointGussetFlangePlate.CutH1 2093
RobotOM.IRJointGussetCrossPlate.Thick 2084	RobotOM.IRJointGussetFlangePlate.CutH2 2094

RobotOM.IRJointGussetFlangePlate.CutH3 2094	2072
RobotOM.IRJointGussetFlangePlate.CutH4 2094	RobotOM.IRJointGussetSimpleAttachBoltsHorizontal.Rows 2072
RobotOM.IRJointGussetFlangePlate.CutV1 2095	RobotOM.IRJointGussetSimpleAttachBoltsHorizontal.Spacing 2072
RobotOM.IRJointGussetFlangePlate.CutV2 2095	RobotOM.IRJointGussetSimpleAttachBoltsVertical 2072
RobotOM.IRJointGussetFlangePlate.CutV3 2095	RobotOM.IRJointGussetSimpleAttachBoltsVertical.DistanceE 2073
RobotOM.IRJointGussetFlangePlate.CutV4 2096	RobotOM.IRJointGussetSimpleAttachBoltsVertical.DistanceH 2073
RobotOM.IRJointGussetFlangePlate.EH 2096	RobotOM.IRJointGussetSimpleAttachBoltsVertical.Rows 2074
RobotOM.IRJointGussetFlangePlate.EV 2096	RobotOM.IRJointGussetSimpleAttachBoltsVertical.Spacing 2074
RobotOM.IRJointGussetFlangePlate.Gheight 2097	RobotOM.IRJointGussetSimpleAttachment 2067
RobotOM.IRJointGussetFlangePlate.H1 2097	RobotOM.IRJointGussetSimpleAttachment.BoltsAttach 2068
RobotOM.IRJointGussetFlangePlate.H2 2097	RobotOM.IRJointGussetSimpleAttachment.FixType 2068
RobotOM.IRJointGussetFlangePlate.H3 2097	RobotOM.IRJointGussetSimpleAttachment.WeldsAttach 2068
RobotOM.IRJointGussetFlangePlate.H4 2098	RobotOM.IRJointGussetSimpleAttachment.WeldType 2068
RobotOM.IRJointGussetFlangePlate.HOffset 2098	RobotOM.IRJointGussetSimpleAttachWelds 2074
RobotOM.IRJointGussetFlangePlate.Length 2098	RobotOM.IRJointGussetSimpleAttachWelds.ThickEdgeA 2075
RobotOM.IRJointGussetFlangePlate.Material 2099	RobotOM.IRJointGussetSimpleAttachWelds.ThickEdgeB 2075
RobotOM.IRJointGussetFlangePlate.NewPlateDefinition 2099	RobotOM.IRJointGussetSimpleLoad 2106
RobotOM.IRJointGussetFlangePlate.Thick 2099	RobotOM.IRJointGussetSimpleLoad.Fx 2106
RobotOM.IRJointGussetFlangePlate.Type 2099	RobotOM.IRJointGussetSimplePlate 2061
RobotOM.IRJointGussetFlangePlate.VOffset 2100	RobotOM.IRJointGussetSimplePlate.CornersType 2062
RobotOM.IRJointGussetFlangePlateRegularType 2111	RobotOM.IRJointGussetSimplePlate.DistanceD 2062
RobotOM.IRJointGussetFlangeProfileCutting 2090	RobotOM.IRJointGussetSimplePlate.DistanceEH 2062
RobotOM.IRJointGussetSimple 2057	RobotOM.IRJointGussetSimplePlate.DistanceEV 2063
RobotOM.IRJointGussetSimple.Attachment 2058	RobotOM.IRJointGussetSimplePlate.H1 2063
RobotOM.IRJointGussetSimple.BoltsClassVector 2059	RobotOM.IRJointGussetSimplePlate.H2 2063
RobotOM.IRJointGussetSimple.BoltsDimensionNamesVector 2059	RobotOM.IRJointGussetSimplePlate.H3 2064
RobotOM.IRJointGussetSimple.BoltsDimensionsVector 2059	RobotOM.IRJointGussetSimplePlate.H4 2064
RobotOM.IRJointGussetSimple.Diagonale 2060	RobotOM.IRJointGussetSimplePlate.Height 2064
RobotOM.IRJointGussetSimple.FixType 2060	RobotOM.IRJointGussetSimplePlate.Material 2064
RobotOM.IRJointGussetSimple.GussetPlate 2060	RobotOM.IRJointGussetSimplePlate.Thick 2065
RobotOM.IRJointGussetSimple.ProfilePosition 2060	RobotOM.IRJointGussetSimplePlate.V1 2065
RobotOM.IRJointGussetSimpleAttachBolts 2069	RobotOM.IRJointGussetSimplePlate.V2 2065
RobotOM.IRJointGussetSimpleAttachBolts.ClassName 2069	RobotOM.IRJointGussetSimplePlate.V3 2066
RobotOM.IRJointGussetSimpleAttachBolts.Diameter 2070	RobotOM.IRJointGussetSimplePlate.V4 2066
RobotOM.IRJointGussetSimpleAttachBolts.Friction 2070	RobotOM.IRJointGussetSimplePlate.Width 2066
RobotOM.IRJointGussetSimpleAttachBolts.Horizontal 2070	RobotOM.IRJointGussetSimpleProfilePosition 2057
RobotOM.IRJointGussetSimpleAttachBolts.Vertical 2070	RobotOM.IRJointGussetWeldsDiag 2052
RobotOM.IRJointGussetSimpleAttachBoltsHorizontal 2071	RobotOM.IRJointGussetWeldsDiag.Length1 2053
RobotOM.IRJointGussetSimpleAttachBoltsHorizontal.DistanceEB 2071	
RobotOM.IRJointGussetSimpleAttachBoltsHorizontal.DistanceH1	

RobotOM.IRJointGussetWeldsDiag.Length2 2053	RobotOM.IRJointKnee.WeldWeb 1955
RobotOM.IRJointGussetWeldsDiag.ThickCorner 2053	RobotOM.IRJointKneeBolts 1930
RobotOM.IRJointGussetWeldsDiag.ThickFlange 2054	RobotOM.IRJointKneeBolts.ClassName 1931
RobotOM.IRJointKnee 1942	RobotOM.IRJointKneeBolts.Cols 1932
RobotOM.IRJointKnee.Beam 1945	RobotOM.IRJointKneeBolts.DiameterName 1932
RobotOM.IRJointKnee.Bolts 1945	RobotOM.IRJointKneeBolts.EqualSpac 1932
RobotOM.IRJointKneeBracketLow 1945	RobotOM.IRJointKneeBolts.Height1 1933
RobotOM.IRJointKneeBracketUp 1946	RobotOM.IRJointKneeBolts.Rows 1933
RobotOM.IRJointKnee.Column 1946	RobotOM.IRJointKneeBolts.SpacingH 1933
RobotOM.IRJointKnee.DefComponentMaterial 1946	RobotOM.IRJointKneeBolts.SpacingV 1933
RobotOM.IRJointKnee.FixType 1946	RobotOM.IRJointKneeBolts.Symmetry 1934
RobotOM.IRJointKnee.FlanPlateLower 1947	RobotOM.IRJointKneeBracket 1934
RobotOM.IRJointKnee.FlanPlatePosLower 1947	RobotOM.IRJointKneeBracket.Angle 1935
RobotOM.IRJointKnee.FlanPlatePosUpper 1947	RobotOM.IRJointKneeBracket.AngleExt 1937
RobotOM.IRJointKnee.FlanPlateUpper 1948	RobotOM.IRJointKneeBracket.Exist 1935
RobotOM.IRJointKnee.IsBolted 1948	RobotOM.IRJointKneeBracket.Height 1936
RobotOM.IRJointKnee.IsDefComponentMaterialSet 1948	RobotOM.IRJointKneeBracket.Length 1936
RobotOM.IRJointKnee.KneeType 1948	RobotOM.IRJointKneeBracket.Material 1936
RobotOM.IRJointKnee.Material 1949	RobotOM.IRJointKneeBracket.ThickFlange 1936
RobotOM.IRJointKnee.MaterPlates 1949	RobotOM.IRJointKneeBracket.ThickWeb 1937
RobotOM.IRJointKnee.Plate 1949	RobotOM.IRJointKneeBracket.Width 1937
RobotOM.IRJointKnee.PlatePosition 1949	RobotOM.IRJointKneeDiagonalStiff 1956
RobotOM.IRJointKnee.PlatePositionLow 1950	RobotOM.IRJointKneeDiagonalStiff.Exist 1956
RobotOM.IRJointKnee.StiffDiag 1950	RobotOM.IRJointKneeDiagonalStiff.Material 1957
RobotOM.IRJointKnee.StiffLow 1950	RobotOM.IRJointKneeDiagonalStiff.Thick 1957
RobotOM.IRJointKnee.StiffTypeLow 1951	RobotOM.IRJointKneeDiagonalStiff.Type 1957
RobotOM.IRJointKnee.StiffTypeUp 1951	RobotOM.IRJointKneeDiagonalStiffType 1955
RobotOM.IRJointKnee.StiffUp 1951	RobotOM.IRJointKneeFixType 1941
RobotOM.IRJointKnee.TensionPlateLow 1951	RobotOM.IRJointKneeLoad 1958
RobotOM.IRJointKnee.TensionPlateUp 1952	RobotOM.IRJointKneeLoad.M 1959
RobotOM.IRJointKnee.VStiffLow 1952	RobotOM.IRJointKneeLoad.N 1959
RobotOM.IRJointKnee.VStiffUp 1952	RobotOM.IRJointKneeLoad.Q 1959
RobotOM.IRJointKnee.WebBeamStiffenerVLower 1952	RobotOM.IRJointKneeMaterials 1960
RobotOM.IRJointKnee.WebBeamStiffenerVUpper 1953	RobotOM.IRJointKneeReinfType 1941
RobotOM.IRJointKnee.WebColumnStiffenerHLower 1953	RobotOM.IRJointKneeStiffColumn 1957
RobotOM.IRJointKnee.WebColumnStiffenerHUpper 1953	RobotOM.IRJointKneeStiffColumn.Thick 1958
RobotOM.IRJointKnee.WebPlate 1954	RobotOM.IRJointKneeType 1941
RobotOM.IRJointKnee.WebStiffType 1954	RobotOM.IRJointKneeWebPlate 1938
RobotOM.IRJointKnee.WeldBracketDownFlange 1954	RobotOM.IRJointKneeWebPlate.Bolts 1939
RobotOM.IRJointKnee.WeldBracketUpFlange 1954	RobotOM.IRJointKneeWebPlate.Height 1939
RobotOM.IRJointKnee.WeldFlange 1955	RobotOM.IRJointKneeWebPlate.Material 1939
RobotOM.IRJointKnee.WeldStiff 1955	RobotOM.IRJointKneeWebPlate.Thick 1939

RobotOM.IRJointKneeWebPlate.Type 1940	RobotOM.IRJointProfile 2162
RobotOM.IRJointKneeWebPlate.WeldLong 1940	RobotOM.IRJointProfile.Angle 2163
RobotOM.IRJointKneeWebPlate.WeldTran 1940	RobotOM.IRJointProfile.AngleExt 2164
RobotOM.IRJointKneeWebPlate.Width 1940	RobotOM.IRJointProfile.BarNumber 2163
RobotOM.IRJointKneeWebStiffType 1955	RobotOM.IRJointProfile.Edit 2165
RobotOM.IRJointLoad 2165	RobotOM.IRJointProfile.EditComponent 2165
RobotOM.IRJointLoad.Cases 2166	RobotOM.IRJointProfile.Material 2164
RobotOM.IRJointLoad.Type 2166	RobotOM.IRJointProfile.Section 2164
RobotOM.IRJointLoadType 2167	RobotOM.IRJointSpreadFootingType 1983
RobotOM.IRJointPinnedColumnBase 1962	RobotOM.IRJointTube 2031
RobotOM.IRJointPinnedColumnBase.Anchor 1963	RobotOM.IRJointTube.DiagLeftLower 2033
RobotOM.IRJointPinnedColumnBase.Base 1963	RobotOM.IRJointTube.DiagLeftUpper 2033
RobotOM.IRJointPinnedColumnBase.BasePlateMaterial 1964	RobotOM.IRJointTube.DiagRightUpper 2033
RobotOM.IRJointPinnedColumnBase.Bearing 1964	RobotOM.IRJointTube.Flange 2033
RobotOM.IRJointPinnedColumnBase.BearingPlate 1964	RobotOM.IRJointTube.PostUpper 2034
RobotOM.IRJointPinnedColumnBase.Bolts 1964	RobotOM.IRJointTube.StiffHBracketMaterial 2034
RobotOM.IRJointPinnedColumnBase.FootPlate 1965	RobotOM.IRJointTube.StiffHoriz 2034
RobotOM.IRJointPinnedColumnBase.Materials 1965	RobotOM.IRJointTube.StiffLateral 2035
RobotOM.IRJointPinnedColumnBase.NodeNumber 1965	RobotOM.IRJointTube.TubeType 2035
RobotOM.IRJointPinnedColumnBase.Profile 1965	RobotOM.IRJointTube.WeldDiag 2035
RobotOM.IRJointPinnedColumnBase.Square 1966	RobotOM.IRJointTube.WeldStiff 2035
RobotOM.IRJointPinnedColumnBase.StiffHoriz 1966	RobotOM.IRJointTubeDiagProfile 2039
RobotOM.IRJointPinnedColumnBase.StiffType 1966	RobotOM.IRJointTubeDiagProfile.Angle 2040
RobotOM.IRJointPinnedColumnBase.StiffVert 1966	RobotOM.IRJointTubeDiagProfile.BarNumber 2040
RobotOM.IRJointPinnedColumnBase.Washer 1967	RobotOM.IRJointTubeDiagProfile.Exist 2040
RobotOM.IRJointPinnedColumnBase.Wedge 1967	RobotOM.IRJointTubeDiagProfile.Gap 2041
RobotOM.IRJointPinnedColumnBase.Welds 1967	RobotOM.IRJointTubeDiagProfile.Length 2041
RobotOM.IRJointPinnedColumnBaseStiffType 1961	RobotOM.IRJointTubeDiagProfile.Material 2041
RobotOM.IRJointPinnedLoad 1967	RobotOM.IRJointTubeDiagProfile.Overlap 2041
RobotOM.IRJointPinnedLoad.Nc 1968	RobotOM.IRJointTubeDiagProfile.Profile 2042
RobotOM.IRJointPinnedLoad.Nt 1968	RobotOM.IRJointTubeDiagProfile.Section 2042
RobotOM.IRJointPinnedLoad.NTy 1969	RobotOM.IRJointTubeFlangeProfile 2036
RobotOM.IRJointPinnedLoad.NTz 1969	RobotOM.IRJointTubeFlangeProfile.BarNumber 2037
RobotOM.IRJointPinnedLoad.Ty 1969	RobotOM.IRJointTubeFlangeProfile.Excentr 2037
RobotOM.IRJointPinnedLoad.Tz 1969	RobotOM.IRJointTubeFlangeProfile.Exist 2038
RobotOM.IRJointPlate 2151	RobotOM.IRJointTubeFlangeProfile.Material 2038
RobotOM.IRJointPlate.AngleExt 2153	RobotOM.IRJointTubeFlangeProfile.Profile 2038
RobotOM.IRJointPlate.Exist 2152	RobotOM.IRJointTubeFlangeProfile.Section 2039
RobotOM.IRJointPlate.Length 2152	RobotOM.IRJointTubeLoad 2045
RobotOM.IRJointPlate.Material 2152	RobotOM.IRJointTubeLoad.DiagLeftLowerM 2046
RobotOM.IRJointPlate.Thick 2152	RobotOM.IRJointTubeLoad.DiagLeftLowerN 2046
RobotOM.IRJointPlate.Width 2153	RobotOM.IRJointTubeLoad.DiagLeftUpperM 2046

RobotOM.IRJointTubeLoad.DiagLeftUpperN 2046	RobotOM.IRobotAddIn.GetExpectedVersion 1543
RobotOM.IRJointTubeLoad.DiagRightUpperM 2047	RobotOM.IRobotAddIn.InstallCommands 1544
RobotOM.IRJointTubeLoad.DiagRightUpperN 2047	RobotOM.IRobotAddInMngr 1544
RobotOM.IRJointTubeLoad.FlangeM 2047	RobotOM.IRobotAddInMngr.InstallCommand 1545
RobotOM.IRJointTubeLoad.FlangeN 2048	RobotOM.IRobotAddInRegistrar 1549
RobotOM.IRJointTubeLoad.FlangeQ 2048	RobotOM.IRobotAddInRegistrar.Guid 1550
RobotOM.IRJointTubeLoad.PostUpperN 2048	RobotOM.IRobotAddInRegistrar.InstallMenu 1551
RobotOM.IRJointTubePostProfile 2042	RobotOM.IRobotAddInRegistrar.ProductName 1550
RobotOM.IRJointTubePostProfile.BarNumber 2043	RobotOM.IRobotAddInRegistrar.ProgId 1550
RobotOM.IRJointTubePostProfile.Exist 2043	RobotOM.IRobotAddInRegistrar.ProviderName 1551
RobotOM.IRJointTubePostProfile.Length 2043	RobotOM.IRobotAddInRegistrar.Register 1552
RobotOM.IRJointTubePostProfile.Material 2044	RobotOM.IRobotAddInRegistrar.Unregister 1552
RobotOM.IRJointTubePostProfile.Profile 2044	RobotOM.IRobotAdvancedResultServer 919
RobotOM.IRJointTubePostProfile.Section 2044	RobotOM.IRobotAdvancedResultServer.Eigenvalues 920
RobotOM.IRJointTubeProfileType 2036	RobotOM.IRobotAdvancedResultServer.Eigenvectors 920
RobotOM.IRJointTubeType 2036	RobotOM.IRobotAdvancedResultServer.FootfallValue 922
RobotOM.IRJointWebFlangeRelativePos 2147	RobotOM.IRobotAdvancedResultServer.FRF 920
RobotOM.IRJointWebType 2157	RobotOM.IRobotAdvancedResultServer.MassSum 921
RobotOM.IRJointWedge 2001	RobotOM.IRobotAdvancedResultServer.SpectralCoeffs 921
RobotOM.IRJointWedge.Length 2001	RobotOM.IRobotAdvancedResultServer.TimeHistory 921
RobotOM.IRJointWedge.Profile 2002	RobotOM.IRobotAdvancedSupportData 513
RobotOM.IRJointWedge.Thick 2002	RobotOM.IRobotAdvancedSupportData.B 514
RobotOM.IRJointWedge.Type 2002	RobotOM.IRobotAdvancedSupportData.D 514
RobotOM.IRJointWedge.Width 2002	RobotOM.IRobotAdvancedSupportData.EquivalentElasticity 515
RobotOM.IRJointWedge.XTypeMaterial 2003	RobotOM.IRobotAdvancedSupportData.H 515
RobotOM.IRJointWedgeType 2003	RobotOM.IRobotAdvancedSupportData.IsEquivalentElasticity 515
RobotOM.IRJointWeld 2149	RobotOM.IRobotAdvancedSupportData.Type 515
RobotOM.IRJointWeld.Strength 2150	RobotOM.IRobotAdvancedSupportType 513
RobotOM.IRJointWeld.Thick 2150	RobotOM.IRobotApplication 1528
RobotOM.IRJointWithAngles 2027	RobotOM.IRobotApplication.CmpntFactory 1529
RobotOM.IRJointWithAngles.Angle 2028	RobotOM.IRobotApplication.Interactive 1530
RobotOM.IRJointWithAngles.BeamBolts 2028	RobotOM.IRobotApplication.Ils360 1530
RobotOM.IRJointWithAngles.BeamCut 2029	RobotOM.IRobotApplication.Kernel 1530
RobotOM.IRJointWithAngles.BeamProfile 2029	RobotOM.IRobotApplication.LicenseCheckEntitlement 1533
RobotOM.IRJointWithAngles.ColumnBolts 2029	RobotOM.IRobotApplication.Preferences 1530
RobotOM.IRJointWithAngles.ColumnProfile 2029	RobotOM.IRobotApplication.ProgramVersion 1531
RobotOM.IRJointWithAngles.Distance 2030	RobotOM.IRobotApplication.Project 1531
RobotOM.IRobotActiveModelType 1351	RobotOM.IRobotApplication.Quit 1533
RobotOM.IRobotAddIn 1541	RobotOM.IRobotApplication.UserControl 1531
RobotOM.IRobotAddIn.Connect 1542	RobotOM.IRobotApplication.Version 1532
RobotOM.IRobotAddIn.Disconnect 1543	RobotOM.IRobotApplication.Visible 1532
RobotOM.IRobotAddIn.DoCommand 1543	

RobotOM.IRobotApplication.Window 1532	RobotOM.IRobotBar.ChangeOrientation 628
RobotOM.IRobotBackground 1325	RobotOM.IRobotBar.Elements 621
RobotOM.IRobotBackground.Color 1326	RobotOM.IRobotBar.End 622
RobotOM.IRobotBackground.FilePath 1327	RobotOM.IRobotBar.EndNode 622
RobotOM.IRobotBackground.InsertParams 1327	RobotOM.IRobotBar.Gamma 622
RobotOM.IRobotBackground.IsOn 1327	RobotOM.IRobotBar.GetElemsData 628
RobotOM.IRobotBackground.Name 1327	RobotOM.IRobotBar.GetLCS 629
RobotOM.IRobotBackground.Number 1328	RobotOM.IRobotBar.GetSimpleBars 629
RobotOM.IRobotBackground.Save 1328	RobotOM.IRobotBar.InactiveBar 623
RobotOM.IRobotBackground.VisibilityRange 1328	RobotOM.IRobotBar.IsSuperBar 623
RobotOM.IRobotBackgroundInsertParams 1321	RobotOM.IRobotBar.Length 623
RobotOM.IRobotBackgroundInsertParams.InsertPoint 1323	RobotOM.IRobotBar.Name 623
RobotOM.IRobotBackgroundInsertParams.Layers 1323	RobotOM.IRobotBar.NameTemplate 624
RobotOM.IRobotBackgroundInsertParams.Mirror 1323	RobotOM.IRobotBar.ReversedOffset 624
RobotOM.IRobotBackgroundInsertParams.Plane 1324	RobotOM.IRobotBar.ReversedRelease 624
RobotOM.IRobotBackgroundInsertParams.PositionOnNormalAxis 1324	RobotOM.IRobotBar.ReversedSection 625
RobotOM.IRobotBackgroundInsertParams.ReferencePoint 1324	RobotOM.IRobotBar.SetOffset 629
RobotOM.IRobotBackgroundInsertParams.RotationAngle 1324	RobotOM.IRobotBar.SetOrientation 630
RobotOM.IRobotBackgroundInsertParams.ScalingFactor 1325	RobotOM.IRobotBar.SetSection 630
RobotOM.IRobotBackgroundInsertParams.Units 1325	RobotOM.IRobotBar.ShearForces 625
RobotOM.IRobotBackgroundLayers 1319	RobotOM.IRobotBar.Start 625
RobotOM.IRobotBackgroundLayers.Count 1320	RobotOM.IRobotBar.StartNode 626
RobotOM.IRobotBackgroundLayers.FindName 1320	RobotOM.IRobotBar.StructuralType 626
RobotOM.IRobotBackgroundLayers.Get 1321	RobotOM.IRobotBar.TensionCompression 626
RobotOM.IRobotBackgroundLayers.IsImported 1321	RobotOM.IRobotBar.TrussBar 627
RobotOM.IRobotBackgroundLayers.SetImported 1321	RobotOM.IRobotBar.UniqueId 627
RobotOM.IRobotBackgroundServer 1331	RobotOM.IRobotBarBucklingData 892
RobotOM.IRobotBackgroundServer.Create 1332	RobotOM.IRobotBarBucklingData.BuckLengthY 893
RobotOM.IRobotBackgroundServer.Get 1332	RobotOM.IRobotBarBucklingData.BuckLengthZ 893
RobotOM.IRobotBackgroundServer.GetAllNumbers 1332	RobotOM.IRobotBarBucklingData.CriticalCoef 893
RobotOM.IRobotBackgroundServer.Remove 1333	RobotOM.IRobotBarBucklingData.CriticalForce 894
RobotOM.IRobotBackgroundServer.RemoveAll 1333	RobotOM.IRobotBarBucklingData.SlendY 894
RobotOM.IRobotBackgroundVisibilityRange 1329	RobotOM.IRobotBarBucklingData.SlendZ 894
RobotOM.IRobotBackgroundVisibilityRange.FromPos 1330	RobotOM.IRobotBarBucklingServer 876
RobotOM.IRobotBackgroundVisibilityRange.ToPos 1330	RobotOM.IRobotBarBucklingServer.CriticalCoef 877
RobotOM.IRobotBackgroundVisibilityRange.Type 1330	RobotOM.IRobotBarBucklingServer.EigenValue 877
RobotOM.IRobotBackgroundVisibilityRangeType 1329	RobotOM.IRobotBarCableAssemblingParamType 537
RobotOM.IRobotBar 618	RobotOM.IRobotBarCableData 535
RobotOM.IRobotBar.AnalyzeTTMethod 621	RobotOM.IRobotBarCableData.AssemblingParam 535
RobotOM.IRobotBar.CalcGamma 628	RobotOM.IRobotBarCableData.AssemblingParamValue 536
	RobotOM.IRobotBarCableData.MaterialName 536
	RobotOM.IRobotBarCableData.SectionAX 536

RobotOM.IRobotBarDeadRecordValues 43	RobotOM.IRobotBarElementDataSet.GetBarNumber 653
RobotOM.IRobotBarDeflectionData 890	RobotOM.IRobotBarElementDataSet.GetElem 653
RobotOM.IRobotBarDeflectionData.PosUX 891	RobotOM.IRobotBarElementDataSet.GetElemCountForBar 653
RobotOM.IRobotBarDeflectionData.PosUY 891	RobotOM.IRobotBarElementDataSet.GetElemForBar 654
RobotOM.IRobotBarDeflectionData.PosUZ 891	RobotOM.IRobotBarEnd 642
RobotOM.IRobotBarDeflectionData.UX 891	RobotOM.IRobotBarEnd.GetLabel 644
RobotOM.IRobotBarDeflectionData.UY 891	RobotOM.IRobotBarEnd.GetLabelName 644
RobotOM.IRobotBarDeflectionData.UZ 892	RobotOM.IRobotBarEnd.GetLabels 644
RobotOM.IRobotBarDeflectionServer 870	RobotOM.IRobotBarEnd.GetOffsetValue 645
RobotOM.IRobotBarDeflectionServer.DynCombMaxValue 872	RobotOM.IRobotBarEnd.HasLabel 645
RobotOM.IRobotBarDeflectionServer.DynCombValue 872	RobotOM.IRobotBarEnd.Node 643
RobotOM.IRobotBarDeflectionServer.DynMaxValue 872	RobotOM.IRobotBarEnd.RemoveLabel 645
RobotOM.IRobotBarDeflectionServer.DynValue 872	RobotOM.IRobotBarEnd.SetLabel 646
RobotOM.IRobotBarDeflectionServer.MaxValue 873	RobotOM.IRobotBarEndBracketData 615
RobotOM.IRobotBarDeflectionServer.MaxValueEx 873	RobotOM.IRobotBarEndBracketData.GetValue 617
RobotOM.IRobotBarDeflectionServer.Value 873	RobotOM.IRobotBarEndBracketData.IsValueRelative 617
RobotOM.IRobotBarDeflectionServer.ValueEx 874	RobotOM.IRobotBarEndBracketData.SetValue 617
RobotOM.IRobotBarDilatationRecordValues 40	RobotOM.IRobotBarEndBracketData.Type 616
RobotOM.IRobotBarDisplacementServer 877	RobotOM.IRobotBarEndBracketDataValue 615
RobotOM.IRobotBarDisplacementServer.DynCombValue 878	RobotOM.IRobotBarEndBracketType 615
RobotOM.IRobotBarDisplacementServer.DynValue 878	RobotOM.IRobotBarEndOffsetData 613
RobotOM.IRobotBarDisplacementServer.Value 879	RobotOM.IRobotBarEndOffsetData.MemberLength 614
RobotOM.IRobotBarElasticGroundData 538	RobotOM.IRobotBarEndOffsetData.UX 614
RobotOM.IRobotBarElasticGroundData.GetOneDir 540	RobotOM.IRobotBarEndOffsetData.UY 615
RobotOM.IRobotBarElasticGroundData.HX 538	RobotOM.IRobotBarEndOffsetData.UZ 615
RobotOM.IRobotBarElasticGroundData.KY 539	RobotOM.IRobotBarEndReleaseData 577
RobotOM.IRobotBarElasticGroundData.KZ 539	RobotOM.IRobotBarEndReleaseData.AX 578
RobotOM.IRobotBarElasticGroundData.SetOneDir 540	RobotOM.IRobotBarEndReleaseData.AY 579
RobotOM.IRobotBarElement 646	RobotOM.IRobotBarEndReleaseData.AZ 579
RobotOM.IRobotBarElement.EndNode 647	RobotOM.IRobotBarEndReleaseData.BX 579
RobotOM.IRobotBarElement.Inactive 647	RobotOM.IRobotBarEndReleaseData.BY 580
RobotOM.IRobotBarElement.IsCalc 648	RobotOM.IRobotBarEndReleaseData.BZ 580
RobotOM.IRobotBarElement.Number 648	RobotOM.IRobotBarEndReleaseData.HX 580
RobotOM.IRobotBarElement.StartNode 648	RobotOM.IRobotBarEndReleaseData.HY 580
RobotOM.IRobotBarElementData 649	RobotOM.IRobotBarEndReleaseData.HZ 581
RobotOM.IRobotBarElementData.Bar 649	RobotOM.IRobotBarEndReleaseData.KX 581
RobotOM.IRobotBarElementData.GetEndNode 650	RobotOM.IRobotBarEndReleaseData.KY 581
RobotOM.IRobotBarElementData.GetStartNode 651	RobotOM.IRobotBarEndReleaseData.KZ 581
RobotOM.IRobotBarElementData.Number 650	RobotOM.IRobotBarEndReleaseData.NonlinearModel 582
RobotOM.IRobotBarElementDataSet 651	RobotOM.IRobotBarEndReleaseData.RX 582
RobotOM.IRobotBarElementDataSet.BarCount 652	RobotOM.IRobotBarEndReleaseData.RY 582
RobotOM.IRobotBarElementDataSet.ElemCount 652	RobotOM.IRobotBarEndReleaseData.RZ 582

RobotOM.IRobotBarEndReleaseData.UX 583	RobotOM.IRobotBarOffsetData.ObjectNumber 613
RobotOM.IRobotBarEndReleaseData.UY 583	RobotOM.IRobotBarOffsetData.Position 613
RobotOM.IRobotBarEndReleaseData.UZ 583	RobotOM.IRobotBarOffsetData.Start 613
RobotOM.IRobotBarEndReleaseValue 575	RobotOM.IRobotBarOffsetMemberLength 618
RobotOM.IRobotBarForceConcentrateRecordValues 41	RobotOM.IRobotBarReleaseData 576
RobotOM.IRobotBarForceData 884	RobotOM.IRobotBarReleaseData.EndNode 576
RobotOM.IRobotBarForceData.FX 885	RobotOM.IRobotBarReleaseData.StartNode 577
RobotOM.IRobotBarForceData.FY 885	RobotOM.IRobotBarResultServer 864
RobotOM.IRobotBarForceData.FZ 885	RobotOM.IRobotBarResultServer.Buckling 865
RobotOM.IRobotBarForceData.KY 885	RobotOM.IRobotBarResultServer.Deflections 865
RobotOM.IRobotBarForceData.KYAvailable 885	RobotOM.IRobotBarResultServer.Displacements 866
RobotOM.IRobotBarForceData.KZ 886	RobotOM.IRobotBarResultServer.Forces 866
RobotOM.IRobotBarForceData.KZAvailable 886	RobotOM.IRobotBarResultServer.GeoImperfections 867
RobotOM.IRobotBarForceData.MX 886	RobotOM.IRobotBarResultServer.Stresses 866
RobotOM.IRobotBarForceData.MY 886	RobotOM.IRobotBarSectionComplexData 568
RobotOM.IRobotBarForceData.MZ 887	RobotOM.IRobotBarSectionComplexData.B 569
RobotOM.IRobotBarForceServer 867	RobotOM.IRobotBarSectionComplexData.Count 569
RobotOM.IRobotBarForceServer.DynCombValue 868	RobotOM.IRobotBarSectionComplexData.D 569
RobotOM.IRobotBarForceServer.DynCombValueByNPoints 868	RobotOM.IRobotBarSectionComplexData.Get 570
RobotOM.IRobotBarForceServer.DynValue 869	RobotOM.IRobotBarSectionComplexData.GetShape 570
RobotOM.IRobotBarForceServer.DynValueByNPoints 869	RobotOM.IRobotBarSectionComplexData.GetValue 570
RobotOM.IRobotBarForceServer.Value 869	RobotOM.IRobotBarSectionComplexData.Set 571
RobotOM.IRobotBarForceServer.ValueByNPoints 870	RobotOM.IRobotBarSectionComponentShape 571
RobotOM.IRobotBarForceServer.ValueByNPointsEx 870	RobotOM.IRobotBarSectionConcreteCutsPosition 547
RobotOM.IRobotBarForceServer.ValueEx 870	RobotOM.IRobotBarSectionConcreteData 560
RobotOM.IRobotBarGeoImperfectionsAxis 587	RobotOM.IRobotBarSectionConcreteData.BeamCutsPosition 561
RobotOM.IRobotBarGeoImperfectionsData 583	RobotOM.IRobotBarSectionConcreteData.CalcGeometry 562
RobotOM.IRobotBarGeoImperfectionsData.GetBucklingCoeff 585	RobotOM.IRobotBarSectionConcreteData.GetReduction 563
RobotOM.IRobotBarGeoImperfectionsData.GetUser 585	RobotOM.IRobotBarSectionConcreteData.GetTapered 563
RobotOM.IRobotBarGeoImperfectionsData.IsAutomatic 585	RobotOM.IRobotBarSectionConcreteData.GetValue 563
RobotOM.IRobotBarGeoImperfectionsData.IsMinus 586	RobotOM.IRobotBarSectionConcreteData.IsBeam 562
RobotOM.IRobotBarGeoImperfectionsData.SetAutomatic 586	RobotOM.IRobotBarSectionConcreteData.IsColumn 562
RobotOM.IRobotBarGeoImperfectionsData.SetMinus 586	RobotOM.IRobotBarSectionConcreteData.SetReduction 564
RobotOM.IRobotBarGeoImperfectionsData.SetUser 587	RobotOM.IRobotBarSectionConcreteData.SetTapered 564
RobotOM.IRobotBarIntersectRelationship 2209	RobotOM.IRobotBarSectionConcreteData.SetValue 564
RobotOM.IRobotBarMomentDistributedRecordValues 41	RobotOM.IRobotBarSectionConcreteDataValue 546
RobotOM.IRobotBarOffsetAutoPosition 618	RobotOM.IRobotBarSectionData 548
RobotOM.IRobotBarOffsetData 611	RobotOM.IRobotBarSectionData.AliasCount 550
RobotOM.IRobotBarOffsetData.AxisOffset 612	RobotOM.IRobotBarSectionData.CalcNonstdGeometry 555
RobotOM.IRobotBarOffsetData.CoordinateSystem 612	RobotOM.IRobotBarSectionData.Concrete 550
RobotOM.IRobotBarOffsetData.End 612	RobotOM.IRobotBarSectionData.CreateNonstd 555
	RobotOM.IRobotBarSectionData.DrawSymbol 555

RobotOM.IRobotBarSectionData.ElasticParams 551	RobotOM.IRobotBarSectionQuantitySurvey.GetVolume 1079
RobotOM.IRobotBarSectionData.FindAlias 556	RobotOM.IRobotBarSectionQuantitySurvey.GetWeight 1080
RobotOM.IRobotBarSectionData.FindNonstd 556	RobotOM.IRobotBarSectionShapeType 542
RobotOM.IRobotBarSectionData.GetAlias 556	RobotOM.IRobotBarSectionSpecialData 572
RobotOM.IRobotBarSectionData.GetAliasEx 556	RobotOM.IRobotBarSectionSpecialData.DbFullName 572
RobotOM.IRobotBarSectionData.GetNonstd 557	RobotOM.IRobotBarSectionSpecialData.DbName 573
RobotOM.IRobotBarSectionData.GetValue 557	RobotOM.IRobotBarSectionSpecialData.GetValue 574
RobotOM.IRobotBarSectionData.IsConcrete 551	RobotOM.IRobotBarSectionSpecialData.Section 573
RobotOM.IRobotBarSectionData.IsJoist 551	RobotOM.IRobotBarSectionSpecialData.SetValue 574
RobotOM.IRobotBarSectionData.IsSpecial 551	RobotOM.IRobotBarSectionSpecialDataValue 574
RobotOM.IRobotBarSectionData.LoadFromDBase 557	RobotOM.IRobotBarSectionType 545
RobotOM.IRobotBarSectionData.LoadFromDBase2 558	RobotOM.IRobotBarServer 630
RobotOM.IRobotBarSectionData.MaterialName 552	RobotOM.IRobotBarServer.BeginMultiOperation 634
RobotOM.IRobotBarSectionData.Members 552	RobotOM.IRobotBarServer.CalcGamma 635
RobotOM.IRobotBarSectionData.Name 552	RobotOM.IRobotBarServer.ChangeOrientation 635
RobotOM.IRobotBarSectionData.NonstdCount 553	RobotOM.IRobotBarServer.Create 635
RobotOM.IRobotBarSectionData.RemoveNonstd 558	RobotOM.IRobotBarServer.CreateSuperBar 636
RobotOM.IRobotBarSectionData.SetValue 558	RobotOM.IRobotBarServer.EndMultiOperation 636
RobotOM.IRobotBarSectionData.ShapeType 553	RobotOM.IRobotBarServer.FindWithId 636
RobotOM.IRobotBarSectionData.Special 553	RobotOM.IRobotBarServer.FreeNumber 632
RobotOM.IRobotBarSectionData.Type 554	RobotOM.IRobotBarServer.GetAnalyzeTTMethodEnabled 636
RobotOM.IRobotBarSectionDataValue 544	RobotOM.IRobotBarServer.GetElemsData 637
RobotOM.IRobotBarSectionElasticParams 565	RobotOM.IRobotBarServer.GetLCS 637
RobotOM.IRobotBarSectionElasticParams.Active 566	RobotOM.IRobotBarServer.GetName 637
RobotOM.IRobotBarSectionElasticParams.L1 566	RobotOM.IRobotBarServer.GetNameTemplate 638
RobotOM.IRobotBarSectionElasticParams.L2 566	RobotOM.IRobotBarServer.GetStructuralType 638
RobotOM.IRobotBarSectionElasticParams.MaterialModel 566	RobotOM.IRobotBarServer.GetUniquelD 638
RobotOM.IRobotBarSectionElasticParams.N 567	RobotOM.IRobotBarServer.IsNotNullActive 639
RobotOM.IRobotBarSectionElasticParams.N1 567	RobotOM.IRobotBarServer.IsTrussBar 639
RobotOM.IRobotBarSectionElasticParams.N2 567	RobotOM.IRobotBarServer.NonlinearHingeModels 633
RobotOM.IRobotBarSectionNonstdData 559	RobotOM.IRobotBarServer.NonlinearHinges 633
RobotOM.IRobotBarSectionNonstdData.GetValue 560	RobotOM.IRobotBarServer.SetAnalyzeTTMethod 639
RobotOM.IRobotBarSectionNonstdData.Position 559	RobotOM.IRobotBarServer.SetBoolInactive 640
RobotOM.IRobotBarSectionNonstdData.SetValue 560	RobotOM.IRobotBarServer.SetLabelExt 640
RobotOM.IRobotBarSectionNonstdDataValue 546	RobotOM.IRobotBarServerSetNameTemplate 640
RobotOM.IRobotBarSectionQuantitySurvey 1077	RobotOM.IRobotBarServer.SetOrientation 641
RobotOM.IRobotBarSectionQuantitySurvey.Count 1078	RobotOM.IRobotBarServer.SetShearForces 641
RobotOM.IRobotBarSectionQuantitySurvey.GetLength 1078	RobotOM.IRobotBarServer.SetStructuralType 641
RobotOM.IRobotBarSectionQuantitySurvey.GetName 1078	RobotOM.IRobotBarServer.SetTensionCompression 642
RobotOM.IRobotBarSectionQuantitySurvey.GetPaintArea 1079	RobotOM.IRobotBarServer.SetTrussBar 642
RobotOM.IRobotBarSectionQuantitySurvey.GetUnitWeight 1079	RobotOM.IRobotBarServer.Update 642
	RobotOM.IRobotBarStressData 887

RobotOM.IRobotBarStressData.FXSX 888	RobotOM.IRobotCalcEngine.AnalysisParams 1555
RobotOM.IRobotBarStressData.ShearY 888	RobotOM.IRobotCalcEngine.AutoFreezeResults 1555
RobotOM.IRobotBarStressData.ShearZ 888	RobotOM.IRobotCalcEngine.AutoGenerateModel 1555
RobotOM.IRobotBarStressData.Smax 888	RobotOM.IRobotCalcEngine.BucklingDeformation 1556
RobotOM.IRobotBarStressData.SmaxMY 889	RobotOM.IRobotCalcEngine.Calculate 1558
RobotOM.IRobotBarStressData.SmaxMZ 889	RobotOM.IRobotCalcEngine.CalculateEx 1558
RobotOM.IRobotBarStressData.Smin 889	RobotOM.IRobotCalcEngine.DAM 1556
RobotOM.IRobotBarStressData.SminMY 889	RobotOM.IRobotCalcEngine.GenerateModel 1559
RobotOM.IRobotBarStressData.SminMZ 889	RobotOM.IRobotCalcEngine.GenerationParams 1556
RobotOM.IRobotBarStressData.Torsion 889	RobotOM.IRobotCalcEngine.SaveResultsInExternalFile 1556
RobotOM.IRobotBarStressServer 874	RobotOM.IRobotCalcEngine.SeismicResultsSaveParams 1557
RobotOM.IRobotBarStressServer.DynCombValue 875	RobotOM.IRobotCalcEngine.StatusWindowParent 1557
RobotOM.IRobotBarStressServer.DynValue 875	RobotOM.IRobotCalcEngine.StopCalculation 1559
RobotOM.IRobotBarStressServer.Value 875	RobotOM.IRobotCalcEngine.UseStatusWindow 1557
RobotOM.IRobotBarStressServer.ValueEx 876	RobotOM.IRobotCalcEngineEvents 1584
RobotOM.IRobotBarTensionCompression 646	RobotOM.IRobotCalcEngineEvents.CalcMessage 1584
RobotOM.IRobotBarThermalRecordValues 43	RobotOM.IRobotCalcMessageSeverityLevel 1585
RobotOM.IRobotBarTrapezoidalRecordValues 42	RobotOM.IRobotCalcMessageSource 1585
RobotOM.IRobotBarUniformRecordValues 41	RobotOM.IRobotCalculationMode 1586
RobotOM.IRobotBucklingAnalysisMethod 360	RobotOM.IRobotCalculationModelCoherence 1318
RobotOM.IRobotBucklingAnalysisParams 356	RobotOM.IRobotCalculationResume 1028
RobotOM.IRobotBucklingAnalysisParams.Increment 357	RobotOM.IRobotCalculationResume.DiagonalStiffnessMatrixMax 1029
RobotOM.IRobotBucklingAnalysisParams.IsNonlinear 358	RobotOM.IRobotCalculationResume.DiagonalStiffnessMatrixMin 1029
RobotOM.IRobotBucklingAnalysisParams.IterationsCount 358	RobotOM.IRobotCalculationResume.DiagonalStiffnessMatrixPrecision 1029
RobotOM.IRobotBucklingAnalysisParams.Method 358	RobotOM.IRobotCalculationResume.EquationSolvingMethodUsed 1030
RobotOM.IRobotBucklingAnalysisParams.ModesCount 358	RobotOM.IRobotCalculationResume.GetEnergy 1030
RobotOM.IRobotBucklingAnalysisParams.NonlinearParams 359	RobotOM.IRobotCalculationResume.GetEnergyPrecision 1031
RobotOM.IRobotBucklingAnalysisParams.Shift 359	RobotOM.IRobotCalculationStatus 1586
RobotOM.IRobotBucklingAnalysisParams.SturmVerification 359	RobotOM.IRobotCalculationsType 1540
RobotOM.IRobotBucklingAnalysisParams.Tolerance 360	RobotOM.IRobotCase 404
RobotOM.IRobotBucklingDeformationParams 1581	RobotOM.IRobotCase.AnalizeType 405
RobotOM.IRobotBucklingDeformationParams.CaseNumber 1582	RobotOM.IRobotCase.Name 405
RobotOM.IRobotBucklingDeformationParams.GetModeCoeff 1583	RobotOM.IRobotCase.Nature 405
RobotOM.IRobotBucklingDeformationParams.MaxDisplacement 1582	RobotOM.IRobotCase.Number 406
RobotOM.IRobotBucklingDeformationParams.OmitCaseForDeformations 1582	RobotOM.IRobotCase.Type 406
RobotOM.IRobotBucklingDeformationParams.SetModeCoeff 1583	RobotOM.IRobotCaseAnalizeType 407
RobotOM.IRobotCalcEngine 1553	

RobotOM.IRobotCaseAnalysisModesFilter 432	RobotOM.IRobotCaseServer.Get 414
RobotOM.IRobotCaseAnalysisModesFilter.MassPercentage 432	RobotOM.IRobotCaseServer.GetAll 415
RobotOM.IRobotCaseAnalysisModesFilter.Modes 433	RobotOM.IRobotCaseServer.GetCmpntCount 415
RobotOM.IRobotCaseAnalysisModesFilter.Type 433	RobotOM.IRobotCaseServer.GetMany 415
RobotOM.IRobotCaseAnalysisModesFilterType 433	RobotOM.IRobotCaseServer.GetRelatedValue 416
RobotOM.IRobotCaseCollection 431	RobotOM.IRobotCaseServer.GetUniqueId 416
RobotOM.IRobotCaseCombination 422	RobotOM.IRobotCaseServer.IsAuxiliary 416
RobotOM.IRobotCaseCombination.CaseFactors 423	RobotOM.IRobotCaseServer.QCmbTau 410
RobotOM.IRobotCaseCombination.CombinationType 423	RobotOM.IRobotCaseServer.SEFLSeismicEngine 410
RobotOM.IRobotCaseCombination.GetAnalysisParams 426	RobotOM.IRobotCaseServer.SetAuxiliary 417
RobotOM.IRobotCaseCombination.IsAuxiliary 424	RobotOM.IRobotCaseServer.SnowWindEngine 410
RobotOM.IRobotCaseCombination.Label 424	RobotOM.IRobotCaseServer.TimeHistoryFunctions 410
RobotOM.IRobotCaseCombination.NatureName 424	RobotOM.IRobotCaseServer.WindLoadsSimulationEngine 411
RobotOM.IRobotCaseCombination.Quadratic 425	RobotOM.IRobotCaseType 429
RobotOM.IRobotCaseCombination.SeismicType 425	RobotOM.IRobotCladdingData 609
RobotOM.IRobotCaseCombination.SetAnalysisParams 426	RobotOM.IRobotCladdingData.Method 610
RobotOM.IRobotCaseCombination.SetNatureExt 427	RobotOM.IRobotCladdingData.Type 610
RobotOM.IRobotCaseCombination.Signed 425	RobotOM.IRobotCladdingMethod 611
RobotOM.IRobotCaseCombination.UniqueId 425	RobotOM.IRobotCladdingType 610
RobotOM.IRobotCaseFactor 430	RobotOM.IRobotCmdInfo 1547
RobotOM.IRobotCaseFactor.CaseNumber 430	RobotOM.IRobotCmdInfo.Id 1547
RobotOM.IRobotCaseFactor.Factor 431	RobotOM.IRobotCmdInfo.MenuChecked 1548
RobotOM.IRobotCaseFactorMngr 427	RobotOM.IRobotCmdInfo.MenuEnabled 1548
RobotOM.IRobotCaseFactorMngr.Count 428	RobotOM.IRobotCmdInfo.Name 1548
RobotOM.IRobotCaseFactorMngr.Delete 428	RobotOM.IRobotCmdList 1545
RobotOM.IRobotCaseFactorMngr.Get 428	RobotOM.IRobotCmdList.Count 1546
RobotOM.IRobotCaseFactorMngr.New 429	RobotOM.IRobotCmdList.Get 1546
RobotOM.IRobotCaseNature 406	RobotOM.IRobotCmdList.New 1546
RobotOM.IRobotCasePredefinedNumber 474	RobotOM.IRobotCodeCmbActionCoeffType 162
RobotOM.IRobotCaseRelatedValueType 442	RobotOM.IRobotCodeCmbActionServer 158
RobotOM.IRobotCaseServer 407	RobotOM.IRobotCodeCmbActionServer.Count 159
RobotOM.IRobotCaseServer.BeginMultiOperation 412	RobotOM.IRobotCodeCmbActionServer.GetCoeff 160
RobotOM.IRobotCaseServer.CodeCmbEngine 409	RobotOM.IRobotCodeCmbActionServer.GetName 160
RobotOM.IRobotCaseServer.CreateCombination 412	RobotOM.IRobotCodeCmbActionServer.GetNature 160
RobotOM.IRobotCaseServer.CreateMobile 412	RobotOM.IRobotCodeCmbActionServer.New 161
RobotOM.IRobotCaseServer.CreateSimple 413	RobotOM.IRobotCodeCmbActionServer.Remove 161
RobotOM.IRobotCaseServer.Delete 413	RobotOM.IRobotCodeCmbActionServer.SetCoeff 161
RobotOM.IRobotCaseServer.DeleteMany 413	RobotOM.IRobotCodeCmbActionServerSetName 161
RobotOM.IRobotCaseServer.EndMultiOperation 414	RobotOM.IRobotCodeCmbActionServer.SetNature 162
RobotOM.IRobotCaseServer.Exist 414	RobotOM.IRobotCodeCmbActiveCaseInfo 184
RobotOM.IRobotCaseServer.FindWithId 414	RobotOM.IRobotCodeCmbActiveCaseInfo.CaseNature 185
RobotOM.IRobotCaseServer.FreeNumber 409	RobotOM.IRobotCodeCmbActiveCaseInfo.CaseNumber 185

RobotOM.IRobotCodeCmbActiveCaseInfo.Coefficient	186	170
RobotOM.IRobotCodeCmbActiveCaseInfo.GroupNumber	186	RobotOM.IRobotCodeCmbGenerationParams.Relations 170
RobotOM.IRobotCodeCmbActiveCaseInfo.IsSelected	186	RobotOM.IRobotCodeCmbGenerationParams.SelectCombination 172
RobotOM.IRobotCodeCmbCombPartType	165	RobotOM.IRobotCodeCmbGenerationParams.SelectCombinationType 172
RobotOM.IRobotCodeCmbCombs	163	RobotOM.IRobotCodeCmbGenerationParams.SelectDecidingValue 172
RobotOM.IRobotCodeCmbCombs.Count	163	RobotOM.IRobotCodeCmbGenerationType 186
RobotOM.IRobotCodeCmbCombs.Get	164	RobotOM.IRobotCodeCmbGroup 173
RobotOM.IRobotCodeCmbCombs.New	164	RobotOM.IRobotCodeCmbGroup.Add 175
RobotOM.IRobotCodeCmbCombs.Remove	165	RobotOM.IRobotCodeCmbGroup.AddAll 176
RobotOM.IRobotCodeCmbCombs.Set	165	RobotOM.IRobotCodeCmbGroup.Count 174
RobotOM.IRobotCodeCmbCombType	166	RobotOM.IRobotCodeCmbGroup.IsFull 176
RobotOM.IRobotCodeCmbComponent	154	RobotOM.IRobotCodeCmbGroup.Nature 174
RobotOM.IRobotCodeCmbComponent.Count	155	RobotOM.IRobotCodeCmbGroup.Operator 175
RobotOM.IRobotCodeCmbComponent.Get	155	RobotOM.IRobotCodeCmbGroupRelation 179
RobotOM.IRobotCodeCmbComponentMngr	152	RobotOM.IRobotCodeCmbGroupRelation.AddGroup 181
RobotOM.IRobotCodeCmbComponentMngr.Count	152	RobotOM.IRobotCodeCmbGroupRelation.Clear 181
RobotOM.IRobotCodeCmbComponentMngr.Get	153	RobotOM.IRobotCodeCmbGroupRelation.GetGroup 181
RobotOM.IRobotCodeCmbDecidingValueType	173	RobotOM.IRobotCodeCmbGroupRelation.GetGroupCount 182
RobotOM.IRobotCodeCmbFactor	153	RobotOM.IRobotCodeCmbGroupRelation.GetOperator 182
RobotOM.IRobotCodeCmbFactor.CaseNumber	154	RobotOM.IRobotCodeCmbGroupRelation.Nature 180
RobotOM.IRobotCodeCmbFactor.Factor	154	RobotOM.IRobotCodeCmbGroupRelation.NewRow 182
RobotOM.IRobotCodeCmbFlag	155	RobotOM.IRobotCodeCmbGroupRelation.RemoveRow 182
RobotOM.IRobotCodeCmbGenerationParams	166	RobotOM.IRobotCodeCmbGroupRelation.RowCount 180
RobotOM.IRobotCodeCmbGenerationParams.ActiveCases	168	RobotOM.IRobotCodeCmbGroupRelation.SetGroup 183
RobotOM.IRobotCodeCmbGenerationParams.AllBars	168	RobotOM.IRobotCodeCmbGroupRelation.SetOperator 183
RobotOM.IRobotCodeCmbGenerationParams.AllNodes	168	RobotOM.IRobotCodeCmbGroupRelationServer 183
RobotOM.IRobotCodeCmbGenerationParams.BarSel	168	RobotOM.IRobotCodeCmbGroupRelationServer.Get 184
RobotOM.IRobotCodeCmbGenerationParams.ExtremalSnowFactor	169	RobotOM.IRobotCodeCmbGroupRelationServer.Set 184
RobotOM.IRobotCodeCmbGenerationParams.GenType	169	RobotOM.IRobotCodeCmbGroupServer 176
RobotOM.IRobotCodeCmbGenerationParams.Groups	169	RobotOM.IRobotCodeCmbGroupServer.FindByCase 177
RobotOM.IRobotCodeCmbGenerationParams.IsCombinationSelected	171	RobotOM.IRobotCodeCmbGroupServer.Get 178
RobotOM.IRobotCodeCmbGenerationParams.IsCombinationTypeSelected	171	RobotOM.IRobotCodeCmbGroupServer.GetCount 178
RobotOM.IRobotCodeCmbGenerationParams.IsDecidingValueSelected	171	RobotOM.IRobotCodeCmbGroupServer.New 178
RobotOM.IRobotCodeCmbGenerationParams.NodeSel	169	RobotOM.IRobotCodeCmbOperator 176
RobotOM.IRobotCodeCmbGenerationParams.PointsOnBar	170	RobotOM.IRobotCodeCmbRegulations 156
RobotOM.IRobotCodeCmbGenerationParams.Regulations		RobotOM.IRobotCodeCmbRegulations.Actions 157
		RobotOM.IRobotCodeCmbRegulations.CodeName 157
		RobotOM.IRobotCodeCmbRegulations.Combinations 157

RobotOM.IRobotCodeCmbRegulations.IsEuroCode 157	RobotOM.IRobotDAMNotionalLoads.Coefficient 1592
RobotOM.IRobotCodeCmbRegulations.MaterialType 158	RobotOM.IRobotDAMNotionalLoads.GravityLoadCombEnable d 1593
RobotOM.IRobotCodeCmbRegulations.Version 158	RobotOM.IRobotDAMNotionalLoads.LateralLoadCombEnable d 1593
RobotOM.IRobotCodeCombination 149	RobotOM.IRobotDAMNotionalLoads.LateralLoadCombType 1593
RobotOM.IRobotCodeCombination.CombinationType 150	RobotOM.IRobotDAMPParams 1597
RobotOM.IRobotCodeCombination.Components 150	RobotOM.IRobotDAMPParams.Analysis 1598
RobotOM.IRobotCodeCombination.FindByFlag 151	RobotOM.IRobotDAMPParams.GetNLPDParams 1600
RobotOM.IRobotCodeCombination.Flag 151	RobotOM.IRobotDAMPParams.NotionalLoads 1599
RobotOM.IRobotCodeCombination.UniqueId 151	RobotOM.IRobotDAMPParams.ReducedStiffness 1599
RobotOM.IRobotCodeCombinationEngine 148	RobotOM.IRobotDAMPParams.SetNLPDParams 1600
RobotOM.IRobotCodeCombinationEngine.Generate 149	RobotOM.IRobotDAMPParams.Update 1600
RobotOM.IRobotCodeCombinationEngine.Params 149	RobotOM.IRobotDAMReducedStiffness 1594
RobotOM.IRobotCodeRegistrar 1637	RobotOM.IRobotDAMReducedStiffness.RCBeamsValue 1595
RobotOM.IRobotCodeRegistrar.CodeName 1639	RobotOM.IRobotDAMReducedStiffness.RCColumnsAndWalls Value 1595
RobotOM.IRobotCodeRegistrar.CodeType 1639	RobotOM.IRobotDAMReducedStiffness.RCSlabsValue 1596
RobotOM.IRobotCodeRegistrar.Country 1639	RobotOM.IRobotDAMReducedStiffness.SteelMembersReducti onType 1596
RobotOM.IRobotCodeRegistrar.Guid 1640	RobotOM.IRobotDAMReducedStiffness.SteelMembersTauBV alue 1596
RobotOM.IRobotCodeRegistrar.Manufacturer 1640	RobotOM.IRobotDAMReducedStiffness.SteelMembersValue 1597
RobotOM.IRobotCodeRegistrar.ProgId 1640	RobotOM.IRobotDAMSteelMembersReductionType 1597
RobotOM.IRobotCodeRegistrar.Register 1641	RobotOM.IRobotDatabaseType 1317
RobotOM.IRobotCodeRegistrar.Unregister 1641	RobotOM.IRobotDataObject 29
RobotOM.IRobotCodeType 1303	RobotOM.IRobotDataObject.GetLabel 31
RobotOM.IRobotCollection 9	RobotOM.IRobotDataObject.GetLabelName 31
RobotOM.IRobotCollection.Count 10	RobotOM.IRobotDataObject.GetLabels 31
RobotOM.IRobotCollection.Get 10	RobotOM.IRobotDataObject.HasLabel 32
RobotOM.IRobotCombinationType 429	RobotOM.IRobotDataObject.Number 30
RobotOM.IRobotComponentFactory 1645	RobotOM.IRobotDataObject.RemoveLabel 32
RobotOM.IRobotComponentFactory.Create 1646	RobotOM.IRobotDataObject.SetLabel 32
RobotOM.IRobotComponentFactory.CreateExt 1646	RobotOM.IRobotDataObjectServer 33
RobotOM.IRobotComponentType 1647	RobotOM.IRobotDataObjectServer.Delete 34
RobotOM.IRobotDAMAnalysisType 1589	RobotOM.IRobotDataObjectServer.DeleteMany 34
RobotOM.IRobotDAMCalcModule 1587	RobotOM.IRobotDataObjectServer.Exist 34
RobotOM.IRobotDAMCalcModule.DeleteModel 1589	RobotOM.IRobotDataObjectServer.Get 34
RobotOM.IRobotDAMCalcModule.IsActive 1588	RobotOM.IRobotDataObjectServer.GetAll 35
RobotOM.IRobotDAMCalcModule.Params 1588	RobotOM.IRobotDataObjectServer.GetMany 35
RobotOM.IRobotDAMCalcModule.Run 1589	
RobotOM.IRobotDAMLateralLoadCombType 1594	
RobotOM.IRobotDAMNotionalLoads 1590	
RobotOM.IRobotDAMNotionalLoads.ActiveXN 1591	
RobotOM.IRobotDAMNotionalLoads.ActiveXP 1591	
RobotOM.IRobotDAMNotionalLoads.ActiveYN 1592	
RobotOM.IRobotDAMNotionalLoads.ActiveYP 1592	

RobotOM.IRobotDataObjectServer.RemoveLabel 35	437
RobotOM.IRobotDataObjectServer.SetLabel 36	RobotOM.IRobotDynamicAnalysisExcitationDirection.Quadrati cActive
RobotOM.IRobotDeadRecordValues 51	437
RobotOM.IRobotDegreeOfFreedom 1648	RobotOM.IRobotDynamicAnalysisExcitationDirection.Quadrati cSigned
RobotOM.IRobotDialogId 1469	437
RobotOM.IRobotDirectory 1536	RobotOM.IRobotDynamicAnalysisExcitationDirection.Resoluti onActive
RobotOM.IRobotDirectoryExtension 1538	437
RobotOM.IRobotDisplacementData 882	RobotOM.IRobotDynamicAnalysisExcitationDirection.Rx 438
RobotOM.IRobotDisplacementData.RX 882	RobotOM.IRobotDynamicAnalysisExcitationDirection.Ry 438
RobotOM.IRobotDisplacementData.RY 883	RobotOM.IRobotDynamicAnalysisExcitationDirection.Rz 438
RobotOM.IRobotDisplacementData.RZ 883	RobotOM.IRobotDynamicAnalysisExcitationDirection.UseNor malized
RobotOM.IRobotDisplacementData.UX 883	438
RobotOM.IRobotDisplacementData.UY 883	RobotOM.IRobotDynamicAnalysisExcitationDirection.X 438
RobotOM.IRobotDisplacementData.UZ 883	RobotOM.IRobotDynamicAnalysisExcitationDirection.Y 439
RobotOM.IRobotDynamicAnalysisDamping 439	RobotOM.IRobotDynamicAnalysisExcitationDirection.Z 439
RobotOM.IRobotDynamicAnalysisDamping.ConstValue 440	RobotOM.IRobotEigenvalues 894
RobotOM.IRobotDynamicAnalysisDamping.GetModeDamping 441	RobotOM.IRobotEigenvalues.AvPartCoeff 895
RobotOM.IRobotDynamicAnalysisDamping.GetModes 441	RobotOM.IRobotEigenvalues.Damping 896
RobotOM.IRobotDynamicAnalysisDamping.RemoveModeDa mping 441	RobotOM.IRobotEigenvalues.EigenValue 896
RobotOM.IRobotDynamicAnalysisDamping.SetModeDamping 442	RobotOM.IRobotEigenvalues.Energy 896
RobotOM.IRobotDynamicAnalysisDamping.Type 440	RobotOM.IRobotEigenvalues.Frequence 896
RobotOM.IRobotDynamicAnalysisDampingType 442	RobotOM.IRobotEigenvalues.Period 897
RobotOM.IRobotDynamicAnalysisExcitationDirection 434	RobotOM.IRobotEigenvalues.Precision 897
RobotOM.IRobotDynamicAnalysisExcitationDirection.CombTy pe 435	RobotOM.IRobotEigenvalues.Pulsation 897
RobotOM.IRobotDynamicAnalysisExcitationDirection.Group1 435	RobotOM.IRobotEigenvaluesServer 922
RobotOM.IRobotDynamicAnalysisExcitationDirection.Group2 436	RobotOM.IRobotEigenvaluesServer.CombValue 923
RobotOM.IRobotDynamicAnalysisExcitationDirection.Group3 436	RobotOM.IRobotEigenvaluesServer.Value 923
RobotOM.IRobotDynamicAnalysisExcitationDirection.Lambda 436	RobotOM.IRobotEigenvectorsServer 927
RobotOM.IRobotDynamicAnalysisExcitationDirection.Mi 436	RobotOM.IRobotEigenvectorsServer.CombValue 928
RobotOM.IRobotDynamicAnalysisExcitationDirection.Normaliz edX 436	RobotOM.IRobotEigenvectorsServer.Value 928
RobotOM.IRobotDynamicAnalysisExcitationDirection.Normaliz edY 437	RobotOM.IRobotEmitter 511
RobotOM.IRobotDynamicAnalysisExcitationDirection.Normaliz edZ	RobotOM.IRobotEmitter.EstimatedElemNumber 511
	RobotOM.IRobotEmitter.H0 512
	RobotOM.IRobotEmitter.R1 512
	RobotOM.IRobotEmitter.R2 512
	RobotOM.IRobotEmitter.VariableMeshDensityIncrement 512
	RobotOM.IRobotEquationSolvingMethod 1559
	RobotOM.IRobotEurocodeFactors 1316
	RobotOM.IRobotEurocodeFactors.SteelConnections 1317
	RobotOM.IRobotEurocodeFactors.SteelDesign 1317
	RobotOM.IRobotEurocodeSteelConnectionFactors 1311

RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma0 1312	RobotOM.IRobotExtremeValue.Bar 906
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma1 1312	RobotOM.IRobotExtremeValue.Case 906
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma2 1313	RobotOM.IRobotExtremeValue.CaseCmpnt 906
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma3 1313	RobotOM.IRobotExtremeValue.IsAvailable 907
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma3S er 1313	RobotOM.IRobotExtremeValue.ModeCmb 907
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma4 1314	RobotOM.IRobotExtremeValue.Node 907
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma5 1314	RobotOM.IRobotExtremeValue.Position 908
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma6 1314	RobotOM.IRobotExtremeValue.Value 908
RobotOM.IRobotEurocodeSteelConnectionFactors.Gamma7 1315	RobotOM.IRobotExtremeValueType 910
RobotOM.IRobotEurocodeSteelConnectionFactors.GammaC 1315	RobotOM.IRobotFeExtremeParams 981
RobotOM.IRobotEurocodeSteelConnectionFactors.LoadFrom Code 1315	RobotOM.IRobotFeExtremeParams.CaseCmpnt 983
RobotOM.IRobotEurocodeSteelConnectionFactors.LoadFrom CodeNumber 1316	RobotOM.IRobotFeExtremeParams.CaseSel 983
RobotOM.IRobotEurocodeSteelDesignFactors 1308	RobotOM.IRobotFeExtremeParams.ElementSel 984
RobotOM.IRobotEurocodeSteelDesignFactors.Gamma0 1309	RobotOM.IRobotFeExtremeParams.GetDirX 987
RobotOM.IRobotEurocodeSteelDesignFactors.Gamma1 1309	RobotOM.IRobotFeExtremeParams.Layer 984
RobotOM.IRobotEurocodeSteelDesignFactors.Gamma2 1309	RobotOM.IRobotFeExtremeParams.LayerArbitraryValue 984
RobotOM.IRobotEurocodeSteelDesignFactors.GammaFire 1310	RobotOM.IRobotFeExtremeParams.ModeCmb 985
RobotOM.IRobotEurocodeSteelDesignFactors.LoadFromCod e 1310	RobotOM.IRobotFeExtremeParams.NodeSel 985
RobotOM.IRobotEurocodeSteelDesignFactors.LoadFromCod eNumber 1310	RobotOM.IRobotFeExtremeParams.PanelSel 985
RobotOM.IRobotExternalFileFormat 1266	RobotOM.IRobotFeExtremeParams.ReducedCutPos 986
RobotOM.IRobotExternalPreviewFormat 1510	RobotOM.IRobotFeExtremeParams.Reset 987
RobotOM.IRobotExtremeParams 908	RobotOM.IRobotFeExtremeParams.ResultId 986
RobotOM.IRobotExtremeParams.BarDivision 909	RobotOM.IRobotFeExtremeParams.SetDirX 987
RobotOM.IRobotExtremeParams.Selection 909	RobotOM.IRobotFeExtremeParams.Smoothing 986
RobotOM.IRobotExtremeParams.ValueType 910	RobotOM.IRobotFeExtremeValue 988
RobotOM.IRobotExtremeResultServer 1012	RobotOM.IRobotFeExtremeValue.Case 989
RobotOM.IRobotExtremeResultServer.MaxValue 1012	RobotOM.IRobotFeExtremeValue.CaseCmpnt 990
RobotOM.IRobotExtremeResultServer.MinValue 1013	RobotOM.IRobotFeExtremeValue.Element 990
RobotOM.IRobotExtremeValue 905	RobotOM.IRobotFeExtremeValue.GetDirX 993
	RobotOM.IRobotFeExtremeValue.IsAvailable 990
	RobotOM.IRobotFeExtremeValue.Layer 990
	RobotOM.IRobotFeExtremeValue.LayerArbitraryValue 991
	RobotOM.IRobotFeExtremeValue.ModeCmb 991
	RobotOM.IRobotFeExtremeValue.Node 991
	RobotOM.IRobotFeExtremeValue.Panel 991
	RobotOM.IRobotFeExtremeValue.ReducedCutPos 992
	RobotOM.IRobotFeExtremeValue.ResultId 992
	RobotOM.IRobotFeExtremeValue.Smoothing 992
	RobotOM.IRobotFeExtremeValue.Value 992
	RobotOM.IRobotFeLayerType 932
	RobotOM.IRobotFeMultiExtremeValue 995
	RobotOM.IRobotFeMultiExtremeValue.Count 996

RobotOM.IRobotFeMultiExtremeValue.Get 996	RobotOM.IRobotFeResultParams.GetDirX 937
RobotOM.IRobotFeMultiExtremeValue.GetByResType 997	RobotOM.IRobotFeResultParams.Layer 935
RobotOM.IRobotFeMultiResultType 993	RobotOM.IRobotFeResultParams.LayerArbitraryValue 935
RobotOM.IRobotFeMultiResultType.Add 994	RobotOM.IRobotFeResultParams.ModeCmb 936
RobotOM.IRobotFeMultiResultType.Count 994	RobotOM.IRobotFeResultParams.Node 936
RobotOM.IRobotFeMultiResultType.Get 995	RobotOM.IRobotFeResultParams.Panel 936
RobotOM.IRobotFeMultiResultType.Remove 995	RobotOM.IRobotFeResultParams.SetDirX 937
RobotOM.IRobotFeResultComplex 958	RobotOM.IRobotFeResultPrincipal 951
RobotOM.IRobotFeResultComplex.M_MISES 959	RobotOM.IRobotFeResultPrincipal.M1 953
RobotOM.IRobotFeResultComplex.MXX_BOTTOM 959	RobotOM.IRobotFeResultPrincipal.M1_2 953
RobotOM.IRobotFeResultComplex.MXX_TOP 960	RobotOM.IRobotFeResultPrincipal.M2 953
RobotOM.IRobotFeResultComplex.MYY_BOTTOM 960	RobotOM.IRobotFeResultPrincipal.MAL 953
RobotOM.IRobotFeResultComplex.MYY_TOP 960	RobotOM.IRobotFeResultPrincipal.N1 954
RobotOM.IRobotFeResultComplex.N_MISES 961	RobotOM.IRobotFeResultPrincipal.N1_2 954
RobotOM.IRobotFeResultComplex.S_MISES 961	RobotOM.IRobotFeResultPrincipal.N2 954
RobotOM.IRobotFeResultDetailed 938	RobotOM.IRobotFeResultPrincipal.NAL 955
RobotOM.IRobotFeResultDetailed.MXX 939	RobotOM.IRobotFeResultPrincipal.Q1_2 955
RobotOM.IRobotFeResultDetailed.MXY 940	RobotOM.IRobotFeResultPrincipal.S1 955
RobotOM.IRobotFeResultDetailed.MYY 940	RobotOM.IRobotFeResultPrincipal.S1_2 956
RobotOM.IRobotFeResultDetailed.NXX 940	RobotOM.IRobotFeResultPrincipal.S2 956
RobotOM.IRobotFeResultDetailed.NXY 940	RobotOM.IRobotFeResultPrincipal.SAL 956
RobotOM.IRobotFeResultDetailed.NYY 941	RobotOM.IRobotFeResultPrincipal.T1_2 956
RobotOM.IRobotFeResultDetailed.PNorm 941	RobotOM.IRobotFeResultPrincipal.U 957
RobotOM.IRobotFeResultDetailed.QXX 941	RobotOM.IRobotFeResultPrincipal.UGX 957
RobotOM.IRobotFeResultDetailed.QYY 942	RobotOM.IRobotFeResultPrincipal.UGY 957
RobotOM.IRobotFeResultDetailed.RNorm 942	RobotOM.IRobotFeResultPrincipal.UGZ 958
RobotOM.IRobotFeResultDetailed.RXX 942	RobotOM.IRobotFeResultReduced 966
RobotOM.IRobotFeResultDetailed.RYY 943	RobotOM.IRobotFeResultReduced.CutPos 968
RobotOM.IRobotFeResultDetailed.SXX 943	RobotOM.IRobotFeResultReduced.Height 968
RobotOM.IRobotFeResultDetailed.SXY 943	RobotOM.IRobotFeResultReduced.Length 968
RobotOM.IRobotFeResultDetailed.SYY 943	RobotOM.IRobotFeResultReduced.MY 969
RobotOM.IRobotFeResultDetailed.TXX 944	RobotOM.IRobotFeResultReduced.MZ 969
RobotOM.IRobotFeResultDetailed.TYY 944	RobotOM.IRobotFeResultReduced.NodeLeftBottom 969
RobotOM.IRobotFeResultDetailed.UXX 944	RobotOM.IRobotFeResultReduced.NodeLeftTop 969
RobotOM.IRobotFeResultDetailed.UYY 945	RobotOM.IRobotFeResultReduced.NodeRightBottom 970
RobotOM.IRobotFeResultDetailed.WNorm 945	RobotOM.IRobotFeResultReduced.NodeRightTop 970
RobotOM.IRobotFeResultKind 981	RobotOM.IRobotFeResultReduced.NX 970
RobotOM.IRobotFeResultParams 932	RobotOM.IRobotFeResultReduced.SE 970
RobotOM.IRobotFeResultParams.CalcMethod 934	RobotOM.IRobotFeResultReduced.SO 971
RobotOM.IRobotFeResultParams.Case 934	RobotOM.IRobotFeResultReduced.T 971
RobotOM.IRobotFeResultParams.CaseCmpnt 934	RobotOM.IRobotFeResultReduced.TY 971
RobotOM.IRobotFeResultParams.Element 935	RobotOM.IRobotFeResultReduced.TZ 972

RobotOM.IRobotFeResultReducedCutPosition 972	RobotOM.IRobotFiniteElement.ObjectPartIdx 830
RobotOM.IRobotFeResultReinforcement 961	RobotOM.IRobotFiniteElement.UniqueId 830
RobotOM.IRobotFeResultReinforcement.A_MIN 962	RobotOM.IRobotFiniteElementData 837
RobotOM.IRobotFeResultReinforcement.AX 963	RobotOM.IRobotFiniteElementData.GetNode 839
RobotOM.IRobotFeResultReinforcement.AX_BOTTOM 963	RobotOM.IRobotFiniteElementData.NodeCount 838
RobotOM.IRobotFeResultReinforcement.AX_TOP 963	RobotOM.IRobotFiniteElementData.Number 839
RobotOM.IRobotFeResultReinforcement.AY 964	RobotOM.IRobotFiniteElementData.Object 839
RobotOM.IRobotFeResultReinforcement.AY_BOTTOM 964	RobotOM.IRobotFiniteElementDataSet 840
RobotOM.IRobotFeResultReinforcement.AY_TOP 964	RobotOM.IRobotFiniteElementDataSet.ElemCount 841
RobotOM.IRobotFeResultReinforcement.CalcError 965	RobotOM.IRobotFiniteElementDataSet.GetElem 842
RobotOM.IRobotFeResultReinforcement.E_AX_BOTTOM 965	RobotOM.IRobotFiniteElementDataSet.GetElemCountForObjec ct 842
RobotOM.IRobotFeResultReinforcement.E_AX_TOP 965	RobotOM.IRobotFiniteElementDataSet.GetElemForObject 842
RobotOM.IRobotFeResultReinforcement.E_AY_BOTTOM 965	RobotOM.IRobotFiniteElementDataSet.GetObjectNumber 843
RobotOM.IRobotFeResultReinforcement.E_AY_TOP 966	RobotOM.IRobotFiniteElementDataSet.ObjectCount 841
RobotOM.IRobotFeResultReinforcement.F 966	RobotOM.IRobotFiniteElementNodes 831
RobotOM.IRobotFeResultServer 945	RobotOM.IRobotFiniteElementNodes.Count 832
RobotOM.IRobotFeResultServer.Complex 947	RobotOM.IRobotFiniteElementNodes.Get 833
RobotOM.IRobotFeResultServer.Detailed 948	RobotOM.IRobotFiniteElementNodes.GetAll 833
RobotOM.IRobotFeResultServer.MaxValue 948	RobotOM.IRobotFiniteElementNodes.Set 833
RobotOM.IRobotFeResultServer.MinValue 948	RobotOM.IRobotFiniteElementNodes.SetAll 834
RobotOM.IRobotFeResultServer.MultiMaxValue 949	RobotOM.IRobotFiniteElementServer 834
RobotOM.IRobotFeResultServer.MultiMinValue 949	RobotOM.IRobotFiniteElementServer.CalcArea 836
RobotOM.IRobotFeResultServer.PanelCuts 946	RobotOM.IRobotFiniteElementServer.Create 836
RobotOM.IRobotFeResultServer.Principal 949	RobotOM.IRobotFiniteElementServer.FreeNumber 835
RobotOM.IRobotFeResultServer.Reduced 950	RobotOM.IRobotFiniteElementServer.MeshConcentrate 836
RobotOM.IRobotFeResultServer.ReducedEx 950	RobotOM.IRobotFiniteElementServer.MeshConsolidate 837
RobotOM.IRobotFeResultServer.Reinforcement 950	RobotOM.IRobotFiniteElementServer.Update 837
RobotOM.IRobotFeResultSmoothing 980	RobotOM.IRobotFiniteElementType 831
RobotOM.IRobotFeResultType 979	RobotOM.IRobotFootfallAnalysisExcitationForces 395
RobotOM.IRobotFileInsertParams 1346	RobotOM.IRobotFootfallAnalysisExcitationMethod 395
RobotOM.IRobotFileInsertParams.AsObject 1347	RobotOM.IRobotFootfallAnalysisModalParams 398
RobotOM.IRobotFileInsertParams.GetInsertionPoint 1348	RobotOM.IRobotFootfallAnalysisModalParams.FrequencyLimi t 398
RobotOM.IRobotFileInsertParams.GetRotation 1348	RobotOM.IRobotFootfallAnalysisModalParams.IgnoreDensity 399
RobotOM.IRobotFileInsertParams.ReferenceNode 1347	RobotOM.IRobotFootfallAnalysisModalParams.IncludeMassF orDirX 399
RobotOM.IRobotFileInsertParams.ScaleFactor 1347	RobotOM.IRobotFootfallAnalysisModalParams.IncludeMassF orDirY 399
RobotOM.IRobotFileInsertParams.SetInsertionPoint 1348	RobotOM.IRobotFootfallAnalysisModalParams.IncludeMassF
RobotOM.IRobotFileInsertParams.SetRotation 1349	
RobotOM.IRobotFiniteElement 828	
RobotOM.IRobotFiniteElement.CalcArea 830	
RobotOM.IRobotFiniteElement.FeType 829	
RobotOM.IRobotFiniteElement.Nodes 829	
RobotOM.IRobotFiniteElement.ObjectNumber 829	

orDirZ 400	RobotOM.IRobotFRFAnalysisParams.FinalFrequency 393
RobotOM.IRobotFootfallAnalysisNodeSelection 396	RobotOM.IRobotFRFAnalysisParams.FrequencyDivision 394
RobotOM.IRobotFootfallAnalysisNodeSelection.SelectedNodes 396	RobotOM.IRobotFRFAnalysisParams.IncludeEigenfrequencies 394
RobotOM.IRobotFootfallAnalysisNodeSelection.SelectedPanels 397	RobotOM.IRobotFRFAnalysisParams.InitialFrequency 394
RobotOM.IRobotFootfallAnalysisNodeSelection.Type 397	RobotOM.IRobotFRFResults 913
RobotOM.IRobotFootfallAnalysisNodeSelectionType 397	RobotOM.IRobotFRFResults.Frequency 914
RobotOM.IRobotFootfallAnalysisParams 400	RobotOM.IRobotFRFResults.VX 914
RobotOM.IRobotFootfallAnalysisParams.Damping 401	RobotOM.IRobotFRFResults.VY 914
RobotOM.IRobotFootfallAnalysisParams.ExcitationForces 401	RobotOM.IRobotFRFResults.VZ 915
RobotOM.IRobotFootfallAnalysisParams.ExcitationMethod 402	RobotOM.IRobotFRFResultServer 929
RobotOM.IRobotFootfallAnalysisParams.ExcitationNodes 402	RobotOM.IRobotFRFResultServer.Value 930
RobotOM.IRobotFootfallAnalysisParams.FootstepsNumber 402	RobotOM.IRobotGeoArc 1620
RobotOM.IRobotFootfallAnalysisParams.MaxWalkingFrequency 402	RobotOM.IRobotGeoArc.P1 1621
RobotOM.IRobotFootfallAnalysisParams.MinWalkingFrequency 403	RobotOM.IRobotGeoArc.P2 1621
RobotOM.IRobotFootfallAnalysisParams.ModalParams 403	RobotOM.IRobotGeoArc.P3 1621
RobotOM.IRobotFootfallAnalysisParams.ResponseNodes 403	RobotOM.IRobotGeoArcDefinitionMethod 1626
RobotOM.IRobotFootfallAnalysisParams.WalkersWeight 404	RobotOM.IRobotGeoCircle 1621
RobotOM.IRobotFootfallResults 915	RobotOM.IRobotGeoCircle.P1 1622
RobotOM.IRobotFootfallResults.A 916	RobotOM.IRobotGeoCircle.P2 1622
RobotOM.IRobotFootfallResults.ExcitationNode 916	RobotOM.IRobotGeoCircle.P3 1623
RobotOM.IRobotFootfallResults.Frequency 917	RobotOM.IRobotGeoCircle.PC 1623
RobotOM.IRobotFootfallResults.RF_Overall 917	RobotOM.IRobotGeoContour 1618
RobotOM.IRobotFootfallResults.RF_Resonant 917	RobotOM.IRobotGeoContour.Add 1619
RobotOM.IRobotFootfallResults.RF_Transient 918	RobotOM.IRobotGeoContour.Clear 1619
RobotOM.IRobotFootfallResults.VRMQ 918	RobotOM.IRobotGeoContour.Segments 1619
RobotOM.IRobotFootfallResults.VRMS 918	RobotOM.IRobotGeoCoordinateAxis 1606
RobotOM.IRobotForcesData 897	RobotOM.IRobotGeoCoordinateAxisSense 1626
RobotOM.IRobotForcesData.FX 898	RobotOM.IRobotGeoCoordinateSystem 1607
RobotOM.IRobotForcesData.FY 898	RobotOM.IRobotGeoCurveDiv 1608
RobotOM.IRobotForcesData.FZ 899	RobotOM.IRobotGeoCurveDiv.IsAnalytical 1609
RobotOM.IRobotForcesData.MX 899	RobotOM.IRobotGeoCurveDiv.Length 1609
RobotOM.IRobotForcesData.MY 899	RobotOM.IRobotGeoCurveDiv.Mode 1609
RobotOM.IRobotForcesData.MZ 899	RobotOM.IRobotGeoCurveDiv.N 1610
RobotOM.IRobotFRFAnalysisParams 392	RobotOM.IRobotGeoLayer 1624
RobotOM.IRobotFRFAnalysisParams.Damping 393	RobotOM.IRobotGeoLayer.IsThickDefined 1624
	RobotOM.IRobotGeoLayer.P1 1625
	RobotOM.IRobotGeoLayer.P2 1625
	RobotOM.IRobotGeoLayer.P3 1625
	RobotOM.IRobotGeoLayer.PDir 1625
	RobotOM.IRobotGeoLayer.Thickness 1626
	RobotOM.IRobotGeoObject 1615

RobotOM.IRobotGeoObject.Initialize 1616	RobotOM.IRobotGroup.Store 1039
RobotOM.IRobotGeoObject.Type 1615	RobotOM.IRobotGroupObjectServer 843
RobotOM.IRobotGeoObjectType 1608	RobotOM.IRobotGroupObjectServer.Explode 844
RobotOM.IRobotGeoPlane 1627	RobotOM.IRobotGroupObjectServer.FindFirst 844
RobotOM.IRobotGeoPoint2D 1602	RobotOM.IRobotGroupObjectServer.FindNext 845
RobotOM.IRobotGeoPoint2D.Get 1604	RobotOM.IRobotGroupObjectServer.GetContents 845
RobotOM.IRobotGeoPoint2D.Set 1604	RobotOM.IRobotGroupObjectServer.GroupGiven 845
RobotOM.IRobotGeoPoint2D.X 1603	RobotOM.IRobotGroupObjectServer.GroupSelected 846
RobotOM.IRobotGeoPoint2D.Y 1603	RobotOM.IRobotGroupServer 1040
RobotOM.IRobotGeoPoint3D 1604	RobotOM.IRobotGroupServer.Create 1041
RobotOM.IRobotGeoPoint3D.Get 1606	RobotOM.IRobotGroupServer.Delete 1041
RobotOM.IRobotGeoPoint3D.Set 1606	RobotOM.IRobotGroupServer.Find 1041
RobotOM.IRobotGeoPoint3D.X 1605	RobotOM.IRobotGroupServer.Get 1042
RobotOM.IRobotGeoPoint3D.Y 1605	RobotOM.IRobotGroupServer.GetCount 1042
RobotOM.IRobotGeoPoint3D.Z 1605	RobotOM.IRobotHarmonicAnalysisParams 387
RobotOM.IRobotGeoPoint3DCollection 1623	RobotOM.IRobotHarmonicAnalysisParams.Excitation 387
RobotOM.IRobotGeoPolyline 1616	RobotOM.IRobotHarmonicAnalysisParams.MassMatrix 388
RobotOM.IRobotGeoPolyline.Add 1617	RobotOM.IRobotHtmlView 1460
RobotOM.IRobotGeoPolyline.Clear 1617	RobotOM.IRobotHtmlView.LoadFromFile 1462
RobotOM.IRobotGeoPolyline.Segments 1617	RobotOM.IRobotHtmlView.Printable 1461
RobotOM.IRobotGeoSegment 1610	RobotOM.IRobotHtmlView.SaveToFile 1462
RobotOM.IRobotGeoSegment.P1 1611	RobotOM.IRobotIn3PointsRecordValues 44
RobotOM.IRobotGeoSegment.Type 1611	RobotOM.IRobotInContourRecordValues 48
RobotOM.IRobotGeoSegmentArc 1613	RobotOM.IRobotIterativePreconditionerType 1566
RobotOM.IRobotGeoSegmentArc.Div 1614	RobotOM.IRobotIterativeSolverMemoryUsage 1571
RobotOM.IRobotGeoSegmentArc.P2 1614	RobotOM.IRobotIterativeSolverMethod 1570
RobotOM.IRobotGeoSegmentArc.Set 1614	RobotOM.IRobotIterativeSolverParams 1566
RobotOM.IRobotGeoSegmentCollection 1618	RobotOM.IRobotIterativeSolverParams.AggregationLevelsCount 1567
RobotOM.IRobotGeoSegmentLine 1611	RobotOM.IRobotIterativeSolverParams.AnalyseDiagonale 1568
RobotOM.IRobotGeoSegmentLine.Div 1612	RobotOM.IRobotIterativeSolverParams.CalcKMatrix 1568
RobotOM.IRobotGeoSegmentLine.NDiv 1612	RobotOM.IRobotIterativeSolverParams.InternalIterationsCount 1568
RobotOM.IRobotGeoSegmentLine.Radius 1612	RobotOM.IRobotIterativeSolverParams.MemoryUsage 1569
RobotOM.IRobotGeoSegmentLine.Round 1613	RobotOM.IRobotIterativeSolverParams.Method 1569
RobotOM.IRobotGeoSegmentType 1607	RobotOM.IRobotIterativeSolverParams.Multilevel 1569
RobotOM.IRobotGroup 1036	RobotOM.IRobotIterativeSolverParams.PreconditionerType 1570
RobotOM.IRobotGroup.Color 1037	RobotOM.IRobotIterativeSolverParams.Tolerance 1570
RobotOM.IRobotGroup.CreateCollection 1038	RobotOM.IRobotKernel 2195
RobotOM.IRobotGroup.CreateSelection 1039	RobotOM.IRobotKernel.CalcEngine 2197
RobotOM.IRobotGroup.Name 1037	RobotOM.IRobotKernel.CmpntFactory 2197
RobotOM.IRobotGroup.ObjectType 1038	
RobotOM.IRobotGroup.Read 1039	
RobotOM.IRobotGroup.SeLList 1038	

RobotOM.IRobotKernel.ConcrReinfEngine	2197	RobotOM.IRobotLabelServer.IsPredefinedName	27
RobotOM.IRobotKernel.GetExtension	2201	RobotOM.IRobotLabelServer.IsUsed	27
RobotOM.IRobotKernel.Preferences	2197	RobotOM.IRobotLabelServer.SetDefault	27
RobotOM.IRobotKernel.ProgramName	2198	RobotOM.IRobotLabelServer.Store	28
RobotOM.IRobotKernel.ProgramPath	2198	RobotOM.IRobotLabelServer.StoreWithname	28
RobotOM.IRobotKernel.ProgramVersion	2198	RobotOM.IRobotLabelText	17
RobotOM.IRobotKernel.ProjectActiveModel	2199	RobotOM.IRobotLanguage	1537
RobotOM.IRobotKernel.ProjectNew	2201	RobotOM.IRobotLayoutId	1468
RobotOM.IRobotKernel.ProjectNewFromTemplate	2201	RobotOM.IRobotLicenseEntitlement	1539
RobotOM.IRobotKernel.ProjectOpen	2202	RobotOM.IRobotLicenseEntitlementStatus	1539
RobotOM.IRobotKernel.ProjectPreferences	2199	RobotOM.IRobotLimitState	431
RobotOM.IRobotKernel.ProjectSave	2202	RobotOM.IRobotLinear3DRecordValues	46
RobotOM.IRobotKernel.ProjectSaveAs	2202	RobotOM.IRobotLinearOnEdgesRecordValues	50
RobotOM.IRobotKernel.ProjectUniqueId	2199	RobotOM.IRobotLinearRecordValues	45
RobotOM.IRobotKernel.Structure	2200	RobotOM.IRobotLinearReleaseData	698
RobotOM.IRobotKernel.Version	2200	RobotOM.IRobotLinearReleaseData.HX	699
RobotOM.IRobotKernelPreferences	2203	RobotOM.IRobotLinearReleaseData.KX	699
RobotOM.IRobotKernelPreferences.GetDirectory	2204	RobotOM.IRobotLinearReleaseData.KY	699
RobotOM.IRobotKernelPreferences.GetDirectoryExt	2205	RobotOM.IRobotLinearReleaseData.KZ	700
RobotOM.IRobotKernelPreferences.GetLanguage	2205	RobotOM.IRobotLinearReleaseData.RX	700
RobotOM.IRobotKernelPreferences.Multiprocessing	2204	RobotOM.IRobotLinearReleaseData.UX	700
RobotOM.IRobotKernelPreferences.SetDirectoryExt	2205	RobotOM.IRobotLinearReleaseData.UY	701
RobotOM.IRobotKernelPreferences.SetLanguage	2206	RobotOM.IRobotLinearReleaseData.UZ	701
RobotOM.IRobotLabel	18	RobotOM.IRobotLinearReleaseDef	707
RobotOM.IRobotLabel.Data	19	RobotOM.IRobotLinearReleaseDef.EdgeIdx	708
RobotOM.IRobotLabel.Name	19	RobotOM.IRobotLinearReleaseDef.EdgeObject	708
RobotOM.IRobotLabel.Type	19	RobotOM.IRobotLinearReleaseDef.LabelName	709
RobotOM.IRobotLabel.UniqueId	20	RobotOM.IRobotLinearReleaseDef.Object	709
RobotOM.IRobotLabelServer	20	RobotOM.IRobotLinearReleaseDef.PartIdx	709
RobotOM.IRobotLabelServer.Create	23	RobotOM.IRobotLinearReleaseDefinitionType	701
RobotOM.IRobotLabelServer.CreateLike	23	RobotOM.IRobotLinearReleaseDefList	710
RobotOM.IRobotLabelServer.Delete	23	RobotOM.IRobotLinearReleaseDefList.Add	711
RobotOM.IRobotLabelServer.Exist	24	RobotOM.IRobotLinearReleaseDefList.Clear	711
RobotOM.IRobotLabelServer.FindWithId	24	RobotOM.IRobotLinearReleaseDefList.Count	710
RobotOM.IRobotLabelServer.Get	24	RobotOM.IRobotLinearReleaseDefList.Get	711
RobotOM.IRobotLabelServer.GetAll	24	RobotOM.IRobotLinearReleaseDefList.Remove	712
RobotOM.IRobotLabelServer.GetAvailableNames	25	RobotOM.IRobotLinearReleaseServer	702
RobotOM.IRobotLabelServer.GetDefault	25	RobotOM.IRobotLinearReleaseServer.Count	703
RobotOM.IRobotLabelServer.GetMany	25	RobotOM.IRobotLinearReleaseServer.Find	704
RobotOM.IRobotLabelServer.GetPredefinedName	26	RobotOM.IRobotLinearReleaseServer.FindEdge	704
RobotOM.IRobotLabelServer.GetUniqueId	26	RobotOM.IRobotLinearReleaseServer.FindLabel	705
RobotOM.IRobotLabelServer.IsAvailable	26	RobotOM.IRobotLinearReleaseServerFindObject	705

RobotOM.IRobotLinearReleaseServer.Get	705	RobotOM.IRobotLoadRecordLinear.SetPoint	73
RobotOM.IRobotLinearReleaseServer.GetLabel	706	RobotOM.IRobotLoadRecordLinear.UniqueId	73
RobotOM.IRobotLinearReleaseServer.Remove	706	RobotOM.IRobotLoadRecordLinear3D	61
RobotOM.IRobotLinearReleaseServer.Set	706	RobotOM.IRobotLoadRecordLinear3D.GetPoint	62
RobotOM.IRobotLinearReleaseServer.SetMany	707	RobotOM.IRobotLoadRecordLinear3D.SetPoint	62
RobotOM.IRobotLoadRecord	55	RobotOM.IRobotLoadRecordLinear3D.UniqueId	62
RobotOM.IRobotLoadRecord.Description	56	RobotOM.IRobotLoadRecordMngr	58
RobotOM.IRobotLoadRecord.GetValue	58	RobotOM.IRobotLoadRecordMngr.Count	59
RobotOM.IRobotLoadRecord.Objects	57	RobotOM.IRobotLoadRecordMngr.Create	60
RobotOM.IRobotLoadRecord.ObjectType	57	RobotOM.IRobotLoadRecordMngr.Delete	60
RobotOM.IRobotLoadRecord.SetValue	58	RobotOM.IRobotLoadRecordMngr.Get	60
RobotOM.IRobotLoadRecord.Type	57	RobotOM.IRobotLoadRecordMngr.New	60
RobotOM.IRobotLoadRecord2	74	RobotOM.IRobotLoadRecordThermalIn3Points	65
RobotOM.IRobotLoadRecord2.GetGeoLimits	75	RobotOM.IRobotLoadRecordThermalIn3Points.GetGeoLimits	67
RobotOM.IRobotLoadRecord2.SetGeoLimits	76	RobotOM.IRobotLoadRecordThermalIn3Points.GetPoint	67
RobotOM.IRobotLoadRecord2.UniqueId	75	RobotOM.IRobotLoadRecordThermalIn3Points.SetGeoLimits	67
RobotOM.IRobotLoadRecordBarTrapezoidal	76	RobotOM.IRobotLoadRecordThermalIn3Points.SetPoint	68
RobotOM.IRobotLoadRecordBarTrapezoidal.GetPoint	78	RobotOM.IRobotLoadRecordThermalIn3Points.UniqueId	66
RobotOM.IRobotLoadRecordBarTrapezoidal.PointCount	77	RobotOM.IRobotLoadRecordType	54
RobotOM.IRobotLoadRecordBarTrapezoidal.SetPoint	78	RobotOM.IRobotMassActivationRecordValues	46
RobotOM.IRobotLoadRecordCommon	80	RobotOM.IRobotMassEccentricities	204
RobotOM.IRobotLoadRecordCommon.IsAutoGenerated	80	RobotOM.IRobotMassEccentricities.IsDirX	205
RobotOM.IRobotLoadRecordDead	78	RobotOM.IRobotMassEccentricities.IsDirY	205
RobotOM.IRobotLoadRecordDead.FiniteElems	79	RobotOM.IRobotMassEccentricities.RelativeValues	206
RobotOM.IRobotLoadRecordIn3Points	63	RobotOM.IRobotMassEccentricities.ValueDirX	206
RobotOM.IRobotLoadRecordIn3Points.GetGeoLimits	64	RobotOM.IRobotMassEccentricities.ValueDirY	206
RobotOM.IRobotLoadRecordIn3Points.GetPoint	64	RobotOM.IRobotMassSumServer	923
RobotOM.IRobotLoadRecordIn3Points.SetGeoLimits	65	RobotOM.IRobotMassSumServer.Current	924
RobotOM.IRobotLoadRecordIn3Points.SetPoint	65	RobotOM.IRobotMassSumServer.PartCoeff	924
RobotOM.IRobotLoadRecordIn3Points.UniqueId	64	RobotOM.IRobotMassSumServer.Relative	925
RobotOM.IRobotLoadRecordInContour	68	RobotOM.IRobotMassSumServer.Total	925
RobotOM.IRobotLoadRecordInContour.GetContourPoint	70	RobotOM.IRobotMaterialData	654
RobotOM.IRobotLoadRecordInContour.GetGeoLimits	70	RobotOM.IRobotMaterialData.CB71_Category	657
RobotOM.IRobotLoadRecordInContour.GetPoint	70	RobotOM.IRobotMaterialData.CB71_Humidity	657
RobotOM.IRobotLoadRecordInContour.GetVector	71	RobotOM.IRobotMaterialData.CB71_Nature	658
RobotOM.IRobotLoadRecordInContour.SetContourPoint	71	RobotOM.IRobotMaterialData.CB71_Retreat	658
RobotOM.IRobotLoadRecordInContour.SetGeoLimits	71	RobotOM.IRobotMaterialData.CS	658
RobotOM.IRobotLoadRecordInContour.SetPoint	71	RobotOM.IRobotMaterialData.Default	658
RobotOM.IRobotLoadRecordInContour.SetVector	72	RobotOM.IRobotMaterialData.DumpCoef	659
RobotOM.IRobotLoadRecordInContour.UniqueId	69	RobotOM.IRobotMaterialData.E	659
RobotOM.IRobotLoadRecordLinear	72	RobotOM.IRobotMaterialData.E_5	659
RobotOM.IRobotLoadRecordLinear.GetPoint	73		

RobotOM.IRobotMaterialData.E_Trans 659	RobotOM.IRobotMaterialQuantitySurvey 1074
RobotOM.IRobotMaterialData.EC_Deformation 660	RobotOM.IRobotMaterialQuantitySurvey.Count 1075
RobotOM.IRobotMaterialData.GMean 660	RobotOM.IRobotMaterialQuantitySurvey.GetName 1075
RobotOM.IRobotMaterialData.Kirchoff 660	RobotOM.IRobotMaterialQuantitySurvey.GetType 1076
RobotOM.IRobotMaterialData.LoadFromDBase 666	RobotOM.IRobotMaterialQuantitySurvey.GetVolume 1076
RobotOM.IRobotMaterialData.LX 660	RobotOM.IRobotMaterialQuantitySurvey.GetWeight 1076
RobotOM.IRobotMaterialData.Name 661	RobotOM.IRobotMaterialTimberType 667
RobotOM.IRobotMaterialData.NU 661	RobotOM.IRobotMaterialType 667
RobotOM.IRobotMaterialData.Nuance 661	RobotOM.IRobotMeshAccessType 827
RobotOM.IRobotMaterialData.PN_Deformation 661	RobotOM.IRobotMeshCoonsParams 813
RobotOM.IRobotMaterialData.PN_E_Additional 662	RobotOM.IRobotMeshCoonsParams.ForcingRatio 814
RobotOM.IRobotMaterialData.PN_E_Trans 662	RobotOM.IRobotMeshCoonsParams.PanelDivisionType 814
RobotOM.IRobotMaterialData.RE 662	RobotOM.IRobotMeshDelaunayParams 816
RobotOM.IRobotMaterialData.RE_AxCompr 662	RobotOM.IRobotMeshDelaunayParams.EmittersDefault 817
RobotOM.IRobotMaterialData.RE_AxTens 663	RobotOM.IRobotMeshDelaunayParams.EmittersSmoothing 818
RobotOM.IRobotMaterialData.RE_Bending 663	RobotOM.IRobotMeshDelaunayParams.EmittersUser 818
RobotOM.IRobotMaterialData.RE_Shear 663	RobotOM.IRobotMeshDelaunayParams.H_max 818
RobotOM.IRobotMaterialData.RE_TrCompr 664	RobotOM.IRobotMeshDelaunayParams.H0 819
RobotOM.IRobotMaterialData.RE_TrTens 664	RobotOM.IRobotMeshDelaunayParams.MeshDensity 819
RobotOM.IRobotMaterialData.RO 664	RobotOM.IRobotMeshDelaunayParams.NumberOfLevels 819
RobotOM.IRobotMaterialData.RT 664	RobotOM.IRobotMeshDelaunayParams.Q 820
RobotOM.IRobotMaterialData.SaveToDBase 666	RobotOM.IRobotMeshDelaunayParams.RegularMesh 820
RobotOM.IRobotMaterialData.SecondName 665	RobotOM.IRobotMeshDelaunayParams.Type 820
RobotOM.IRobotMaterialData.Steel_Thermal 665	RobotOM.IRobotMeshDelaunayType 808
RobotOM.IRobotMaterialData.Timber_Type 665	RobotOM.IRobotMeshForcingRatio 805
RobotOM.IRobotMaterialData.Type 665	RobotOM.IRobotMeshGeneration 811
RobotOM.IRobotMaterialDatabase 1300	RobotOM.IRobotMeshGeneration.Division1 812
RobotOM.IRobotMaterialDatabase.Get 1302	RobotOM.IRobotMeshGeneration.Division2 812
RobotOM.IRobotMaterialDatabase.GetAll 1302	RobotOM.IRobotMeshGeneration.ElementSize 813
RobotOM.IRobotMaterialDatabase.GetDefault 1302	RobotOM.IRobotMeshGeneration.Type 813
RobotOM.IRobotMaterialDatabase.Load 1303	RobotOM.IRobotMeshGenerationType 809
RobotOM.IRobotMaterialDatabase.LoadFromFile 1303	RobotOM.IRobotMeshImplementDegree 806
RobotOM.IRobotMaterialDatabase.Name 1301	RobotOM.IRobotMeshMethod 809
RobotOM.IRobotMaterialDatabase.SetDefault 1303	RobotOM.IRobotMeshMethod.ForcingRatio 810
RobotOM.IRobotMaterialElasticModel 668	RobotOM.IRobotMeshMethod.ImplementDegree 810
RobotOM.IRobotMaterialElasticModel.Coeff 669	RobotOM.IRobotMeshMethod.Method 811
RobotOM.IRobotMaterialElasticModel.Model 669	RobotOM.IRobotMeshMethodType 808
RobotOM.IRobotMaterialElasticModel.UnloadingCoeff 670	RobotOM.IRobotMeshPanelDivType 806
RobotOM.IRobotMaterialElasticModel.UnloadingMethod 670	RobotOM.IRobotMeshParams 825
RobotOM.IRobotMaterialElasticType 668	RobotOM.IRobotMeshParams.Flag 825
RobotOM.IRobotMaterialElasticUnloadingMethod 670	RobotOM.IRobotMeshParams.MeshType 826
RobotOM.IRobotMaterialModel 667	RobotOM.IRobotMeshParams.SurfaceParams 826

RobotOM.IRobotMeshParams.VolumeParams 826	RobotOM.IRobotMobileCaseFlag 332
RobotOM.IRobotMeshRefinementType 827	RobotOM.IRobotMobileCaseRoute 333
RobotOM.IRobotMeshSurfaceFEType 807	RobotOM.IRobotMobileCaseRoute.BeginingRouteLimit 334
RobotOM.IRobotMeshSurfaceFiniteElems 815	RobotOM.IRobotMobileCaseRoute.EndRouteLimit 334
RobotOM.IRobotMeshSurfaceFiniteElems.ConversionCoeff 815	RobotOM.IRobotMobileCaseRoute.Geometry 334
RobotOM.IRobotMeshSurfaceFiniteElems.ForceRatio 816	RobotOM.IRobotMobileCaseRoute.GetFactors 336
RobotOM.IRobotMeshSurfaceFiniteElems.Type 816	RobotOM.IRobotMobileCaseRoute.LoadDirection 335
RobotOM.IRobotMeshSurfaceParams 820	RobotOM.IRobotMobileCaseRoute.SetFactors 336
RobotOM.IRobotMeshSurfaceParams.Coons 821	RobotOM.IRobotMobileCaseRoute.Step 335
RobotOM.IRobotMeshSurfaceParams.Delaunay 822	RobotOM.IRobotMobileCaseRoute.Tolerance 335
RobotOM.IRobotMeshSurfaceParams.FiniteElems 822	RobotOM.IRobotMobileCaseSegmentFactors 336
RobotOM.IRobotMeshSurfaceParams.Generation 822	RobotOM.IRobotMobileCaseSegmentFactors.Gamma 337
RobotOM.IRobotMeshSurfaceParams.Method 822	RobotOM.IRobotMobileCaseSegmentFactors.HL 338
RobotOM.IRobotMeshType 805	RobotOM.IRobotMobileCaseSegmentFactors.HR 338
RobotOM.IRobotMeshVolumeParams 823	RobotOM.IRobotMobileCaseSegmentFactors.LL 338
RobotOM.IRobotMeshVolumeParams.AdditionalSurfaceMeshing 824	RobotOM.IRobotMobileCaseSegmentFactors.LR 339
RobotOM.IRobotMeshVolumeParams.FiniteElemsType 824	RobotOM.IRobotMobileCaseSegmentFactors.VL 339
RobotOM.IRobotMeshVolumeParams.MeshDensity 824	RobotOM.IRobotMobileCaseSegmentFactors.VR 339
RobotOM.IRobotMeshVolumetricFEType 807	RobotOM.IRobotMobileDistributedRecordValues 52
RobotOM.IRobotMobileCase 324	RobotOM.IRobotMobilePointForceRecordValues 52
RobotOM.IRobotMobileCase.ApplicationPlaneBars 325	RobotOM.IRobotModalAnalysisAlgorithm 191
RobotOM.IRobotMobileCase.ApplicationPlaneType 325	RobotOM.IRobotModalAnalysisBase 199
RobotOM.IRobotMobileCase.Components 326	RobotOM.IRobotModalAnalysisBase.Add 199
RobotOM.IRobotMobileCase.FindByFlag 328	RobotOM.IRobotModalAnalysisBase.Count 200
RobotOM.IRobotMobileCase.Flag 326	RobotOM.IRobotModalAnalysisBase.Delete 200
RobotOM.IRobotMobileCase.GetRoute 328	RobotOM.IRobotModalAnalysisBase.Get 200
RobotOM.IRobotMobileCase.IsAuxiliary 326	RobotOM.IRobotModalAnalysisLimits 201
RobotOM.IRobotMobileCase.Label 327	RobotOM.IRobotModalAnalysisLimits.DefineLimits 203
RobotOM.IRobotMobileCase.NatureName 327	RobotOM.IRobotModalAnalysisLimits.FrequencyLimitValue 202
RobotOM.IRobotMobileCase.SetNatureExt 329	RobotOM.IRobotModalAnalysisLimits.PeriodLimitValue 202
RobotOM.IRobotMobileCase.SetRoute 329	RobotOM.IRobotModalAnalysisLimits.PulsationLimitValue 202
RobotOM.IRobotMobileCase.UniqueId 327	RobotOM.IRobotModalAnalysisLimitType 192
RobotOM.IRobotMobileCase.Vehicle 327	RobotOM.IRobotModalAnalysisMassMatrixType 191
RobotOM.IRobotMobileCaseApplicationPlaneType 340	RobotOM.IRobotModalAnalysisMode 190
RobotOM.IRobotMobileCaseComponent 331	RobotOM.IRobotModalAnalysisParams 192
RobotOM.IRobotMobileCaseComponent.Point 331	RobotOM.IRobotModalAnalysisParams.Acceleration 194
RobotOM.IRobotMobileCaseComponent.Records 332	RobotOM.IRobotModalAnalysisParams.Base 194
RobotOM.IRobotMobileCaseComponentMngr 329	RobotOM.IRobotModalAnalysisParams.Damping 194
RobotOM.IRobotMobileCaseComponentMngr.Count 330	RobotOM.IRobotModalAnalysisParams.DisregardDensity 194
RobotOM.IRobotMobileCaseComponentMngr.Get 330	RobotOM.IRobotModalAnalysisParams.IncludeDampingInCalculations 195

RobotOM.IRobotModalAnalysisParams.IterationsCount	195	c	
RobotOM.IRobotModalAnalysisParams.Limits	195	1580	
RobotOM.IRobotModalAnalysisParams.MassEccentricities	196	RobotOM.IRobotModelGenerationParams.ToleranceValue	
		1581	
RobotOM.IRobotModalAnalysisParams.MassMatrix	196	RobotOM.IRobotModeSelection	15
RobotOM.IRobotModalAnalysisParams.MassParticipation	196	RobotOM.IRobotModeSelection.Combination	15
RobotOM.IRobotModalAnalysisParams.Method	197	RobotOM.IRobotModeSelection.Mode	16
RobotOM.IRobotModalAnalysisParams.Mode	197	RobotOM.IRobotModeSelection.Type	16
RobotOM.IRobotModalAnalysisParams.ModesCount	197	RobotOM.IRobotModeSelectionType	1035
RobotOM.IRobotModalAnalysisParams.Shifts	198	RobotOM.IRobotMultiCollection	13
RobotOM.IRobotModalAnalysisParams.SturmVerification	198	RobotOM.IRobotMultiCollection.Exist	14
RobotOM.IRobotModalAnalysisParams.Tolerance	198	RobotOM.IRobotMultiCollection.Get	14
RobotOM.IRobotModalAnalysisShifts	203	RobotOM.IRobotMultiCollection.Set	14
RobotOM.IRobotModalAnalysisShifts.IterationsCount	204	RobotOM.IRobotMultiSelection	11
RobotOM.IRobotModalAnalysisShifts.SetDefault	204	RobotOM.IRobotMultiSelection.CaseCmpnt	11
RobotOM.IRobotModalWithStaticForcesAnalysisParams	187	RobotOM.IRobotMultiSelection.Exist	12
RobotOM.IRobotModalWithStaticForcesAnalysisParams.CreateFromStatic	189	RobotOM.IRobotMultiSelection.Get	13
		RobotOM.IRobotMultiSelection.Modes	12
RobotOM.IRobotModalWithStaticForcesAnalysisParams.GetStaticStateParams	190	RobotOM.IRobotMultiSelection.Set	13
		RobotOM.IRobotNamesArray	1629
RobotOM.IRobotModalWithStaticForcesAnalysisParams.Nonlinearity	189	RobotOM.IRobotNamesArray.Count	1630
		RobotOM.IRobotNamesArray.Find	1631
RobotOM.IRobotModalWithStaticForcesAnalysisParams.SetStaticStateParams	190	RobotOM.IRobotNamesArray.Get	1631
		RobotOM.IRobotNamesArray.Set	1631
RobotOM.IRobotModeCombinationType	1035	RobotOM.IRobotNamesArray.GetSize	1632
RobotOM.IRobotModelGenerationParams	1577	RobotOM.IRobotNode	522
RobotOM.IRobotModelGenerationParams.AssemblingCase	1578	RobotOM.IRobotNode.GetCalcSupport	526
RobotOM.IRobotModelGenerationParams.GenerateNodes_BarsAndFiniteElems	1578	RobotOM.IRobotNode.GetEmitter	526
		RobotOM.IRobotNode.HasCalcSupport	524
RobotOM.IRobotModelGenerationParams.GenerateNodes_DiagonalBars	1579	RobotOM.IRobotNode.HasEmitter	524
		RobotOM.IRobotNode.IsCalc	524
RobotOM.IRobotModelGenerationParams.GenerateNodes_VertHorizBars	1579	RobotOM.IRobotNode.RemoveEmitter	526
		RobotOM.IRobotNode.SetEmitter	527
RobotOM.IRobotModelGenerationParams.MaxElementLength	1579	RobotOM.IRobotNode.UniqueId	524
		RobotOM.IRobotNode.X	525
RobotOM.IRobotModelGenerationParams.NeglectedBars	1580	RobotOM.IRobotNode.Y	525
RobotOM.IRobotModelGenerationParams.NeglectedGeoObjects	1580	RobotOM.IRobotNode.Z	525
		RobotOM.IRobotNodeAccelerationRecordValues	54
RobotOM.IRobotModelGenerationParams.ToleranceAutomatic		RobotOM.IRobotNodeAuxiliaryRecordValues	40
		RobotOM.IRobotNodeBucklingServer	861
		RobotOM.IRobotNodeBucklingServer.EigenVector	862
		RobotOM.IRobotNodeBucklingServer.EigenVectorCmb	862
		RobotOM.IRobotNodeCompatibilityData	485

RobotOM.IRobotNodeCompatibilityData.Alpha	486	RobotOM.IRobotNodeDisplacementServer.Value	861
RobotOM.IRobotNodeCompatibilityData.AX	487	RobotOM.IRobotNodeDisplacementServer.ValueEx	861
RobotOM.IRobotNodeCompatibilityData.AY	487	RobotOM.IRobotNodeForceInPointRecordValues	49
RobotOM.IRobotNodeCompatibilityData.AZ	487	RobotOM.IRobotNodeForceRecordValues	39
RobotOM.IRobotNodeCompatibilityData.Beta	488	RobotOM.IRobotNodeResultServer	848
RobotOM.IRobotNodeCompatibilityData.BX	488	RobotOM.IRobotNodeResultServer.Buckling	848
RobotOM.IRobotNodeCompatibilityData.BY	488	RobotOM.IRobotNodeResultServer.Displacements	849
RobotOM.IRobotNodeCompatibilityData.BZ	489	RobotOM.IRobotNodeResultServer.PseudostaticForces	849
RobotOM.IRobotNodeCompatibilityData.Gamma	489	RobotOM.IRobotNodeResultServer.Reactions	849
RobotOM.IRobotNodeCompatibilityData.HX	489	RobotOM.IRobotNodeRigidLinkData	475
RobotOM.IRobotNodeCompatibilityData.HY	489	RobotOM.IRobotNodeRigidLinkData.RX	476
RobotOM.IRobotNodeCompatibilityData.HZ	490	RobotOM.IRobotNodeRigidLinkData.RY	477
RobotOM.IRobotNodeCompatibilityData.KX	490	RobotOM.IRobotNodeRigidLinkData.RZ	477
RobotOM.IRobotNodeCompatibilityData.KY	490	RobotOM.IRobotNodeRigidLinkData.UX	477
RobotOM.IRobotNodeCompatibilityData.KZ	491	RobotOM.IRobotNodeRigidLinkData.UY	478
RobotOM.IRobotNodeCompatibilityData.NonlinearModel	491	RobotOM.IRobotNodeRigidLinkData.UZ	478
RobotOM.IRobotNodeCompatibilityData.RX	491	RobotOM.IRobotNodeRigidLinkDef	478
RobotOM.IRobotNodeCompatibilityData.RY	491	RobotOM.IRobotNodeRigidLinkDef.LabelName	479
RobotOM.IRobotNodeCompatibilityData.RZ	492	RobotOM.IRobotNodeRigidLinkDef.Master	479
RobotOM.IRobotNodeCompatibilityData.UX	492	RobotOM.IRobotNodeRigidLinkDef.Slaves	480
RobotOM.IRobotNodeCompatibilityData.UY	492	RobotOM.IRobotNodeRigidLinkServer	480
RobotOM.IRobotNodeCompatibilityData.UZ	493	RobotOM.IRobotNodeRigidLinkServer.Count	481
RobotOM.IRobotNodeCompatibilityDef	493	RobotOM.IRobotNodeRigidLinkServer.Find	482
RobotOM.IRobotNodeCompatibilityDef.Bars	494	RobotOM.IRobotNodeRigidLinkServer.FindLabel	482
RobotOM.IRobotNodeCompatibilityDef.Compatible	494	RobotOM.IRobotNodeRigidLinkServer.FindMaster	482
RobotOM.IRobotNodeCompatibilityDef.LabelName	494	RobotOM.IRobotNodeRigidLinkServer.FindSlave	483
RobotOM.IRobotNodeCompatibilityDef.Main	495	RobotOM.IRobotNodeRigidLinkServer.Get	483
RobotOM.IRobotNodeCompatibilityServer	495	RobotOM.IRobotNodeRigidLinkServer.GetLabel	483
RobotOM.IRobotNodeCompatibilityServer.Count	496	RobotOM.IRobotNodeRigidLinkServer.Remove	484
RobotOM.IRobotNodeCompatibilityServer.Find	497	RobotOM.IRobotNodeRigidLinkServer.RemoveSlave	484
RobotOM.IRobotNodeCompatibilityServer.FindCompatible	497	RobotOM.IRobotNodeRigidLinkServer.Set	484
RobotOM.IRobotNodeCompatibilityServer.FindLabel	497	RobotOM.IRobotNodeServer	527
RobotOM.IRobotNodeCompatibilityServer.FindMain	498	RobotOM.IRobotNodeServer.CompatibleNodes	528
RobotOM.IRobotNodeCompatibilityServer.Get	498	RobotOM.IRobotNodeServer.Create	530
RobotOM.IRobotNodeCompatibilityServer.GetLabel	498	RobotOM.IRobotNodeServer.FindWithId	530
RobotOM.IRobotNodeCompatibilityServer.Remove	499	RobotOM.IRobotNodeServer.FreeNumber	529
RobotOM.IRobotNodeCompatibilityServer.Set	499	RobotOM.IRobotNodeServer.GetCalcNodes	531
RobotOM.IRobotNodeDisplacementRecordValues	40	RobotOM.IRobotNodeServer.GetCalcSupport	531
RobotOM.IRobotNodeDisplacementServer	859	RobotOM.IRobotNodeServer.GetConnectedBars	531
RobotOM.IRobotNodeDisplacementServer.DynCombValue	860	RobotOM.IRobotNodeServer.GetUniqued	532
RobotOM.IRobotNodeDisplacementServer.DynValue	861	RobotOM.IRobotNodeServer.GetUserNodes	532
		RobotOM.IRobotNodeServer.HasCalcSupport	532

RobotOM.IRobotNodeServer.IsCalc	533	RobotOM.IRobotNonlinearAnalysisParams	315
RobotOM.IRobotNodeServer.NonlinearLinks	529	RobotOM.IRobotNonlinearAnalysisParams.Algorithm	317
RobotOM.IRobotNodeServer.RemoveEmitter	533	RobotOM.IRobotNonlinearAnalysisParams.DegreeOfFreedom	
RobotOM.IRobotNodeServer.RigidLinks	529	317	
RobotOM.IRobotNodeServer.SetEmitter	533	RobotOM.IRobotNonlinearAnalysisParams.DisplacementsRel	
RobotOM.IRobotNodeSupportData	499	ativeCodeTolerance	
RobotOM.IRobotNodeSupportData.Advanced	502	317	
RobotOM.IRobotNodeSupportData.Alpha	502	RobotOM.IRobotNonlinearAnalysisParams.GetSettingsFromP	
RobotOM.IRobotNodeSupportData.AX	502	references	
RobotOM.IRobotNodeSupportData.AY	502	322	
RobotOM.IRobotNodeSupportData.AZ	503	RobotOM.IRobotNonlinearAnalysisParams.IncrementLengthR	
RobotOM.IRobotNodeSupportData.Beta	503	eductionFactor	
RobotOM.IRobotNodeSupportData.BX	503	318	
RobotOM.IRobotNodeSupportData.BY	503	RobotOM.IRobotNonlinearAnalysisParams.IncrementLengthR	
RobotOM.IRobotNodeSupportData.BZ	504	eductionNumber	
RobotOM.IRobotNodeSupportData.ElasticLinear	504	318	
RobotOM.IRobotNodeSupportData.ElasticSurface	504	RobotOM.IRobotNonlinearAnalysisParams.LineSearchMethod	
RobotOM.IRobotNodeSupportData.Gamma	505	Factor	
RobotOM.IRobotNodeSupportData.GetAdvanced	508	318	
RobotOM.IRobotNodeSupportData.GetOneDir	509	RobotOM.IRobotNonlinearAnalysisParams.LoadIncrementNu	
RobotOM.IRobotNodeSupportData.GlobalCoordSystem	505	mber	
RobotOM.IRobotNodeSupportData.HX	505	318	
RobotOM.IRobotNodeSupportData.HY	505	RobotOM.IRobotNonlinearAnalysisParams.MatrixUpdateAfter	
RobotOM.IRobotNodeSupportData.HZ	506	EachIteration	
RobotOM.IRobotNodeSupportData.IsFixed	509	319	
RobotOM.IRobotNodeSupportData.KX	506	RobotOM.IRobotNonlinearAnalysisParams.MatrixUpdateAfter	
RobotOM.IRobotNodeSupportData.KY	506	EachSubdivision	
RobotOM.IRobotNodeSupportData.KZ	506	319	
RobotOM.IRobotNodeSupportData.NonlinearModel	506	RobotOM.IRobotNonlinearAnalysisParams.MaxDisplacement	
RobotOM.IRobotNodeSupportData.RX	507	319	
RobotOM.IRobotNodeSupportData.RY	507	RobotOM.IRobotNonlinearAnalysisParams.MaximumIteration	
RobotOM.IRobotNodeSupportData.RZ	507	NumberForOneIncrement	
RobotOM.IRobotNodeSupportData.SetAdvanced	509	320	
RobotOM.IRobotNodeSupportData.SetFixed	509	RobotOM.IRobotNonlinearAnalysisParams.MaximumNumber	
RobotOM.IRobotNodeSupportData.SetOneDir	510	OfBFGSCorrections	
RobotOM.IRobotNodeSupportData.UX	507	320	
RobotOM.IRobotNodeSupportData.UY	507	RobotOM.IRobotNonlinearAnalysisParams.MaximumNumber	
RobotOM.IRobotNodeSupportData.UZ	508	OfLineSearches	
RobotOM.IRobotNodeSupportFixingDirection	510	320	
RobotOM.IRobotNodeSupportOneDirectionFixingType	510	RobotOM.IRobotNonlinearAnalysisParams.MaxLoadFactor	
RobotOM.IRobotNodeVelocityRecordValues	53	321	
RobotOM.IRobotNonlinearAnalysisAlgorithmType	314	RobotOM.IRobotNonlinearAnalysisParams.NodeNumber	
		321	
		RobotOM.IRobotNonlinearAnalysisParams.PDelta	
		321	
		RobotOM.IRobotNonlinearAnalysisParams.ResidualForcesRel	
		ativeCodeTolerance	
		321	
		RobotOM.IRobotNonlinearAnalysisParams.ResultListEachIter	
		ation	
		322	
		RobotOM.IRobotNonlinearAnalysisParams.SaveSettingsInPre	
		ferences	
		323	
		RobotOM.IRobotNonlinearAnalysisParams.Stiff	
		322	

RobotOM.IRobotNonlinearHingeComponentType	604	RobotOM.IRobotNonlinearHingeModelServer.Get	590
RobotOM.IRobotNonlinearHingeData	601	RobotOM.IRobotNonlinearHingeModelType	598
RobotOM.IRobotNonlinearHingeData.GetModel	602	RobotOM.IRobotNonlinearHingeModelUnloadingType	600
RobotOM.IRobotNonlinearHingeData.IsActive	603	RobotOM.IRobotNonlinearHingeServer	606
RobotOM.IRobotNonlinearHingeData.NormalStress	602	RobotOM.IRobotNonlinearHingeServer.Count	607
RobotOM.IRobotNonlinearHingeData.Remove	603	RobotOM.IRobotNonlinearHingeServer.Get	608
RobotOM.IRobotNonlinearHingeData.SetModel	603	RobotOM.IRobotNonlinearHingeServer.Remove	608
RobotOM.IRobotNonlinearHingeDef	604	RobotOM.IRobotNonlinearHingeServer.Set	608
RobotOM.IRobotNonlinearHingeDef.Bar	605	RobotOM.IRobotNonlinearLink	674
RobotOM.IRobotNonlinearHingeDef.LabelName	605	RobotOM.IRobotNonlinearLink.GetCurveType	676
RobotOM.IRobotNonlinearHingeDef.Offset	606	RobotOM.IRobotNonlinearLink.GetParams	677
RobotOM.IRobotNonlinearHingeDef.Relative	606	RobotOM.IRobotNonlinearLink.ModelType	675
RobotOM.IRobotNonlinearHingeModel	590	RobotOM.IRobotNonlinearLink.Name	675
RobotOM.IRobotNonlinearHingeModel.GetAxisParams	595	RobotOM.IRobotNonlinearLink.SetCurveType	677
RobotOM.IRobotNonlinearHingeModel.GetPoints	595	RobotOM.IRobotNonlinearLink.SetParams	677
RobotOM.IRobotNonlinearHingeModel.LimitCoordX	592	RobotOM.IRobotNonlinearLink.Symetry	676
RobotOM.IRobotNonlinearHingeModel.LimitCoordXValue	592	RobotOM.IRobotNonlinearLinkCurveType	681
RobotOM.IRobotNonlinearHingeModel.LimitCoordY	592	RobotOM.IRobotNonlinearLinkMngr	694
RobotOM.IRobotNonlinearHingeModel.LimitCoordYValue	593	RobotOM.IRobotNonlinearLinkMngr.Delete	694
RobotOM.IRobotNonlinearHingeModel.MixedUnloadingValue	593	RobotOM.IRobotNonlinearLinkMngr.Get	695
RobotOM.IRobotNonlinearHingeModel.ModelType	593	RobotOM.IRobotNonlinearLinkMngr.isDefined	695
RobotOM.IRobotNonlinearHingeModel.Name	594	RobotOM.IRobotNonlinearLinkMngr.Set	695
RobotOM.IRobotNonlinearHingeModel.SetAxisParams	596	RobotOM.IRobotNonlinearLinkModelType	693
RobotOM.IRobotNonlinearHingeModel.SetPoints	596	RobotOM.IRobotNonlinearLinkParams	678
RobotOM.IRobotNonlinearHingeModel.Symetry	594	RobotOM.IRobotNonlinearLinkParams.CurveType	678
RobotOM.IRobotNonlinearHingeModel.UnloadingMethod	594	RobotOM.IRobotNonlinearLinkParamsBLinear	679
RobotOM.IRobotNonlinearHingeModel.AxisParams	596	RobotOM.IRobotNonlinearLinkParamsBLinear.D1	680
RobotOM.IRobotNonlinearHingeModel.AxisParams.ImmediateOccupancy	597	RobotOM.IRobotNonlinearLinkParamsBLinear.K1	680
RobotOM.IRobotNonlinearHingeModel.AxisParams.LifeSafety	597	RobotOM.IRobotNonlinearLinkParamsBLinear.K2	681
RobotOM.IRobotNonlinearHingeModel.AxisParams.StructuralStability	598	RobotOM.IRobotNonlinearLinkParamsCustom	688
RobotOM.IRobotNonlinearHingeModelPoints	598	RobotOM.IRobotNonlinearLinkParamsCustom.Count	689
RobotOM.IRobotNonlinearHingeModelPoints.Count	599	RobotOM.IRobotNonlinearLinkParamsCustom.Get	690
RobotOM.IRobotNonlinearHingeModelPoints.Get	600	RobotOM.IRobotNonlinearLinkParamsCustom.New	690
RobotOM.IRobotNonlinearHingeModelPoints.Set	600	RobotOM.IRobotNonlinearLinkParamsCustom.Remove	690
RobotOM.IRobotNonlinearHingeModelServer	588	RobotOM.IRobotNonlinearLinkParamsCustom.Set	691
RobotOM.IRobotNonlinearHingeModelServer.Count	589	RobotOM.IRobotNonlinearLinkParamsCustomSegment	691
RobotOM.IRobotNonlinearHingeModelServer.Create	589	RobotOM.IRobotNonlinearLinkParamsCustomSegment.Constant	692
RobotOM.IRobotNonlinearHingeModelServer.Delete	589	RobotOM.IRobotNonlinearLinkParamsCustomSegment.Expression	692
RobotOM.IRobotNonlinearHingeModelServer.Find	590	RobotOM.IRobotNonlinearLinkParamsCustomSegment.OriginPoint	692

RobotOM.IRobotNonlinearLinkParamsGapHook 687	RobotOM.IRobotNumbersDictionary.Find 1655
RobotOM.IRobotNonlinearLinkParamsGapHook.D 688	RobotOM.IRobotNumbersDictionary.FindGet 1655
RobotOM.IRobotNonlinearLinkParamsGapHook.K 688	RobotOM.IRobotNumbersDictionary.Get 1655
RobotOM.IRobotNonlinearLinkParamsLinear 679	RobotOM.IRobotNumbersDictionary.Set 1656
RobotOM.IRobotNonlinearLinkParamsLinear.K 679	RobotOM.IRobotObjAttributes 757
RobotOM.IRobotNonlinearLinkParamsParabolic 681	RobotOM.IRobotObjAttributes.DirZ 759
RobotOM.IRobotNonlinearLinkParamsParabolic.Dlim 682	RobotOM.IRobotObjAttributes.GetDirX 760
RobotOM.IRobotNonlinearLinkParamsParabolic.Dmax 683	RobotOM.IRobotObjAttributes.GetLabelName 760
RobotOM.IRobotNonlinearLinkParamsParabolic.Flim 683	RobotOM.IRobotObjAttributes.GetLabels 760
RobotOM.IRobotNonlinearLinkParamsParabolic.K 683	RobotOM.IRobotObjAttributes.GetLCS 760
RobotOM.IRobotNonlinearLinkParamsPlastic 683	RobotOM.IRobotObjAttributes.GetLCSDisplayPosition 761
RobotOM.IRobotNonlinearLinkParamsPlastic.Dlim 684	RobotOM.IRobotObjAttributes.HasLabel 761
RobotOM.IRobotNonlinearLinkParamsPlastic.Flim 684	RobotOM.IRobotObjAttributes.Mesherd 759
RobotOM.IRobotNonlinearLinkParamsPlastic.K 685	RobotOM.IRobotObjAttributes.RemoveLabel 761
RobotOM.IRobotNonlinearLinkParamsPlastic.W 685	RobotOM.IRobotObjAttributes.SetDirX 761
RobotOM.IRobotNonlinearLinkParamsPlasticHardening 685	RobotOM.IRobotObjAttributes.SetLabel 762
RobotOM.IRobotNonlinearLinkParamsPlasticHardening.Dlim 686	RobotOM.IRobotObjectsArray 1634
RobotOM.IRobotNonlinearLinkParamsPlasticHardening.Flim 686	RobotOM.IRobotObjectsArray.Count 1635
RobotOM.IRobotNonlinearLinkParamsPlasticHardening.K0 686	RobotOM.IRobotObjectsArray.Get 1635
RobotOM.IRobotNonlinearLinkParamsPlasticHardening.K1 687	RobotOM.IRobotObjectsArray.Set 1636
RobotOM.IRobotNonlinearLinkParamsPlasticHardening.W 687	RobotOM.IRobotObjectsArray.GetSize 1636
RobotOM.IRobotNonlinearLinkSemiAxisType 693	RobotOM.IRobotObjectStructuralType 846
RobotOM.IRobotNonlinearLinkServer 672	RobotOM.IRobotObjectType 36
RobotOM.IRobotNonlinearLinkServer.Count 672	RobotOM.IRobotObjEdge 754
RobotOM.IRobotNonlinearLinkServer.Create 673	RobotOM.IRobotObjEdge.Divide 755
RobotOM.IRobotNonlinearLinkServer.Find 673	RobotOM.IRobotObjEdge.DivideByPlane 756
RobotOM.IRobotNonlinearLinkServer.Get 673	RobotOM.IRobotObjEdge.DivideN 756
RobotOM.IRobotNonlinearLinkServer.Remove 674	RobotOM.IRobotObjEdge.GetLabel 756
RobotOM.IRobotNumbersArray 1632	RobotOM.IRobotObjEdge.GetLabelName 756
RobotOM.IRobotNumbersArray.Count 1633	RobotOM.IRobotObjEdge.GetLabels 757
RobotOM.IRobotNumbersArray.Get 1633	RobotOM.IRobotObjEdge.HasLabel 757
RobotOM.IRobotNumbersArray.Set 1634	RobotOM.IRobotObjEdge.Path 754
RobotOM.IRobotNumbersArray.GetSize 1634	RobotOM.IRobotObjEdge.RemoveLabel 757
RobotOM.IRobotNumbersCollection 1636	RobotOM.IRobotObjEdge.SetLabel 757
RobotOM.IRobotNumbersCollection.Count 1637	RobotOM.IRobotObjEdgeCollection 764
RobotOM.IRobotNumbersCollection.Get 1637	RobotOM.IRobotObjEdgeSelection 801
RobotOM.IRobotNumbersDictionary 1653	RobotOM.IRobotObjEdgeSelection.Count 802
RobotOM.IRobotNumbersDictionary.Count 1654	RobotOM.IRobotObjEdgeSelection.FromText 802
RobotOM.IRobotNumbersDictionary.Delete 1654	RobotOM.IRobotObjEdgeSelection.Get 803
	RobotOM.IRobotObjEdgeSelection.ToText 803
	RobotOM.IRobotObjLocalXDirDefinitionType 788
	RobotOM.IRobotObjMesh 797

RobotOM.IRobotObjMesh.Freeze 798	RobotOM.IRobotObjObject.Name 768
RobotOM.IRobotObjMesh.Generate 799	RobotOM.IRobotObjObject.NameTemplate 768
RobotOM.IRobotObjMesh.GetBasePoints 800	RobotOM.IRobotObjObject.Nodes 768
RobotOM.IRobotObjMesh.GetQuality 800	RobotOM.IRobotObjObject.PartsCount 769
RobotOM.IRobotObjMesh.IsGenerated 798	RobotOM.IRobotObjObject.Reference 769
RobotOM.IRobotObjMesh.Params 799	RobotOM.IRobotObjObject.SetHostedObjects 772
RobotOM.IRobotObjMesh.Remove 800	RobotOM.IRobotObjObject.StructuralType 769
RobotOM.IRobotObjMesh.SetBasePoints 801	RobotOM.IRobotObjObject.UniquelD 769
RobotOM.IRobotObjModifExtrusion 743	RobotOM.IRobotObjObject.Update 772
RobotOM.IRobotObjModifExtrusion.Vector 744	RobotOM.IRobotObjObjectServer 773
RobotOM.IRobotObjModification 739	RobotOM.IRobotObjObjectServer.AutoRecalcHoles 775
RobotOM.IRobotObjModification.Add 742	RobotOM.IRobotObjObjectServer.BeginMultiOperation 778
RobotOM.IRobotObjModification.Clear 742	RobotOM.IRobotObjObjectServer.CalcArea 778
RobotOM.IRobotObjModification.Filled 740	RobotOM.IRobotObjObjectServer.CalcVol 779
RobotOM.IRobotObjModification.NDiv 741	RobotOM.IRobotObjObjectServer.Create 779
RobotOM.IRobotObjModification.Operations 741	RobotOM.IRobotObjObjectServer.CreateArc 779
RobotOM.IRobotObjModification.Type 741	RobotOM.IRobotObjObjectServer.CreateCircle 780
RobotOM.IRobotObjModificationCollection 763	RobotOM.IRobotObjObjectServer.CreateCone 780
RobotOM.IRobotObjModificationType 739	RobotOM.IRobotObjObjectServer.CreateContour 780
RobotOM.IRobotObjModifLathe 744	RobotOM.IRobotObjObjectServer.CreateCube 781
RobotOM.IRobotObjModifLathe.Angle 745	RobotOM.IRobotObjObjectServer.CreateCylinder 781
RobotOM.IRobotObjModifLathe.AxsP1 745	RobotOM.IRobotObjObjectServer.CreateOnFiniteElems 781
RobotOM.IRobotObjModifLathe.AxsP2 746	RobotOM.IRobotObjObjectServer.CreatePolyline 782
RobotOM.IRobotObjModifPyramid 746	RobotOM.IRobotObjObjectServer.CreateSolid 782
RobotOM.IRobotObjModifPyramid.AddPoint 748	RobotOM.IRobotObjObjectServer.EndMultiOperation 782
RobotOM.IRobotObjModifPyramid.ClearPoints 748	RobotOM.IRobotObjObjectServer.FindWithId 783
RobotOM.IRobotObjModifPyramid.Factors 747	RobotOM.IRobotObjObjectServer.FreeNumber 776
RobotOM.IRobotObjModifPyramid.Points 747	RobotOM.IRobotObjObjectServer.GetAnalyzeTTMethodEnabled 783
RobotOM.IRobotObjObject 764	RobotOM.IRobotObjObjectServer.GetFiniteElemsData 783
RobotOM.IRobotObjObject.AnalyzeTTMethod 766	RobotOM.IRobotObjObjectServer.GetHost 784
RobotOM.IRobotObjObject.CalcArea 770	RobotOM.IRobotObjObjectServer.GetHostedObjects 784
RobotOM.IRobotObjObject.CalcVol 770	RobotOM.IRobotObjObjectServer.GetName 784
RobotOM.IRobotObjObject.FiniteElems 766	RobotOM.IRobotObjObjectServer.GetNameTemplate 785
RobotOM.IRobotObjObject.GetFiniteElemsData 771	RobotOM.IRobotObjObjectServer.GetStructuralType 785
RobotOM.IRobotObjObject.GetHostedObjects 771	RobotOM.IRobotObjObjectServer.GetUniquelD 785
RobotOM.IRobotObjObject.GetPart 771	RobotOM.IRobotObjObjectServer.IsVolume 786
RobotOM.IRobotObjObject.GetPartType 772	RobotOM.IRobotObjObjectServer.LinearReleases 776
RobotOM.IRobotObjObject.Host 767	RobotOM.IRobotObjObjectServer.Mesh 776
RobotOM.IRobotObjObject.Initialize 772	RobotOM.IRobotObjObjectServer.SetAnalyzeTTMethod 786
RobotOM.IRobotObjObject.IsVolume 767	RobotOM.IRobotObjObjectServer.SetHost 786
RobotOM.IRobotObjObject.Main 767	RobotOM.IRobotObjObjectServer.SetHostedObjects 787
RobotOM.IRobotObjObject.Mesh 767	

RobotOM.IRobotObjObjectServer.SetNameTemplate 787	RobotOM.IRobotObjPartReference 793
RobotOM.IRobotObjObjectServer.SetStructuralType 787	RobotOM.IRobotObjPartReference.CalcArea 794
RobotOM.IRobotObjOperation 742	RobotOM.IRobotObjPartReference.CalcVol 795
RobotOM.IRobotObjOperation.Type 743	RobotOM.IRobotObjPartReference.FiniteElems 794
RobotOM.IRobotObjOperationCollection 743	RobotOM.IRobotObjPartReference.Nodes 794
RobotOM.IRobotObjOperationType 739	RobotOM.IRobotObjPartType 773
RobotOM.IRobotObjOperMeshing 752	RobotOM.IRobotOutputFormat 1495
RobotOM.IRobotObjOperMeshing.Add 753	RobotOM.IRobotPageSetup 1484
RobotOM.IRobotObjOperMeshing.Clear 753	RobotOM.IRobotPageSetup.Footer 1486
RobotOM.IRobotObjOperMeshing.Points 752	RobotOM.IRobotPageSetup.FromEdgeFooter 1487
RobotOM.IRobotObjOperMeshing.Vectors 753	RobotOM.IRobotPageSetup.FromEdgeHeader 1487
RobotOM.IRobotObjOperRotation 750	RobotOM.IRobotPageSetup.GetTemplateName 1493
RobotOM.IRobotObjOperRotation.Angle 751	RobotOM.IRobotPageSetup.Gutter 1487
RobotOM.IRobotObjOperRotation.AxsP1 751	RobotOM.IRobotPageSetup.Header 1488
RobotOM.IRobotObjOperRotation.AxsP2 751	RobotOM.IRobotPageSetup.IsCurrent 1488
RobotOM.IRobotObjOperScaling 749	RobotOM.IRobotPageSetup.Load 1493
RobotOM.IRobotObjOperScaling.Center 750	RobotOM.IRobotPageSetup.LoadCurrent 1494
RobotOM.IRobotObjOperScaling.Factor 750	RobotOM.IRobotPageSetup.MarginBottom 1488
RobotOM.IRobotObjOperTranslation 748	RobotOM.IRobotPageSetup.MarginLeft 1488
RobotOM.IRobotObjOperTranslation.Vector 749	RobotOM.IRobotPageSetup.MarginRight 1489
RobotOM.IRobotObjPart 762	RobotOM.IRobotPageSetup.MarginTop 1489
RobotOM.IRobotObjPart.Attribs 763	RobotOM.IRobotPageSetup.PageOrientation 1489
RobotOM.IRobotObjPart.GetGeometry 763	RobotOM.IRobotPageSetup.PaperHeight 1490
RobotOM.IRobotObjPart.Type 763	RobotOM.IRobotPageSetup.PaperSize 1490
RobotOM.IRobotObjPart2 795	RobotOM.IRobotPageSetup.PaperWidth 1490
RobotOM.IRobotObjPart2.CalcArea 797	RobotOM.IRobotPageSetup.Save 1494
RobotOM.IRobotObjPart2.CalcVol 797	RobotOM.IRobotPageSetup.SaveAs 1494
RobotOM.IRobotObjPart2.FiniteElems 796	RobotOM.IRobotPageSetup.StartPageNumber 1491
RobotOM.IRobotObjPart2.Nodes 796	RobotOM.IRobotPageSetup.TemplateCount 1491
RobotOM.IRobotObjPartMain 788	RobotOM.IRobotPageSetup.TemplateName 1491
RobotOM.IRobotObjPartMain.AddModification 792	RobotOM.IRobotPageSetup.TextFrame 1492
RobotOM.IRobotObjPartMain.CalcArea 792	RobotOM.IRobotPageSetup.TitlePage 1492
RobotOM.IRobotObjPartMain.CalcVol 792	RobotOM.IRobotPageSetup.Toc 1492
RobotOM.IRobotObjPartMain.ClearModifications 793	RobotOM.IRobotPageSetup.Variables 1493
RobotOM.IRobotObjPartMain.CurveDiv 789	RobotOM.IRobotPageSetupFrameType 1507
RobotOM.IRobotObjPartMain.DefPoints 790	RobotOM.IRobotPageSetupOrientation 1507
RobotOM.IRobotObjPartMain.Edges 790	RobotOM.IRobotPageSetupTableOfContents 1508
RobotOM.IRobotObjPartMain.FiniteElems 790	RobotOM.IRobotPageSetupTableOfContents.Active 1508
RobotOM.IRobotObjPartMain.Geometry 790	RobotOM.IRobotPageSetupTableOfContents.IncludeTitle 1509
RobotOM.IRobotObjPartMain.ModelPoints 791	RobotOM.IRobotPageSetupTableOfContents.Location 1509
RobotOM.IRobotObjPartMain.Modifications 791	RobotOM.IRobotPageSetupTocLocation 1509
RobotOM.IRobotObjPartMain.Nodes 791	RobotOM.IRobotPanelCalcModelData 714

RobotOM.IRobotPanelCalcModelData.Diaphragm	714	RobotOM.IRobotParamSchema.GetObjectsWithParam	2192
RobotOM.IRobotPanelCalcModelData.FinElemType	715	RobotOM.IRobotParamSchema.GetObjectsWithParamVal	2193
RobotOM.IRobotPanelCalcModelData.LoadTransfer	715	RobotOM.IRobotParamSchema.GetParam	2193
RobotOM.IRobotPanelCalcModelDiaphragm	713	RobotOM.IRobotParamSchema.RemoveAllParams	2193
RobotOM.IRobotPanelCalcModelFinElemType	712	RobotOM.IRobotParamSchema.RemoveParam	2194
RobotOM.IRobotPanelCalcModelLoadTransfer	713	RobotOM.IRobotParamSchema.ResetParam	2194
RobotOM.IRobotPanelCut	976	RobotOM.IRobotParamSchema.SetParam	2194
RobotOM.IRobotPanelCut.Active	976	RobotOM.IRobotParamSchemaDef	2177
RobotOM.IRobotPanelCut.Color	977	RobotOM.IRobotParamSchemaDef.AddParam	2179
RobotOM.IRobotPanelCut.DefType	977	RobotOM.IRobotParamSchemaDef.AddSimpleParam	2180
RobotOM.IRobotPanelCut.Point1	977	RobotOM.IRobotParamSchemaDef.GetParam	2180
RobotOM.IRobotPanelCut.Point2	978	RobotOM.IRobotParamSchemaDef.GetParamName	2180
RobotOM.IRobotPanelCut.Point3	978	RobotOM.IRobotParamSchemaDef.Name	2178
RobotOM.IRobotPanelCutDefinitionType	978	RobotOM.IRobotParamSchemaDef.ParamCount	2179
RobotOM.IRobotPanelCutMngr	972	RobotOM.IRobotParamSchemaDef.ParamNameToUniqueId	2181
RobotOM.IRobotPanelCutMngr.Count	973	RobotOM.IRobotParamSchemaDef.RemoveParam	2181
RobotOM.IRobotPanelCutMngr.Create	974	RobotOM.IRobotParamSchemaMngr	2181
RobotOM.IRobotPanelCutMngr.Find	974	RobotOM.IRobotParamSchemaMngr.Clear	2183
RobotOM.IRobotPanelCutMngr.Get	974	RobotOM.IRobotParamSchemaMngr.Create	2183
RobotOM.IRobotPanelCutMngr.GetName	975	RobotOM.IRobotParamSchemaMngr.Exist	2184
RobotOM.IRobotPanelCutMngr.Remove	975	RobotOM.IRobotParamSchemaMngr.Get	2184
RobotOM.IRobotPanelCutMngr.Store	975	RobotOM.IRobotParamSchemaMngr.GetByName	2184
RobotOM.IRobotParamCollection	2174	RobotOM.IRobotParamSchemaMngr.Remove	2185
RobotOM.IRobotParamCollection.Count	2175	RobotOM.IRobotParamSchemaMngr.RemoveByName	2185
RobotOM.IRobotParamCollection.Find	2176	RobotOM.IRobotParamSchemaMngr.SchemaCount	2182
RobotOM.IRobotParamCollection.GetFullName	2176	RobotOM.IRobotParamServer	2169
RobotOM.IRobotParamCollection.GetId	2176	RobotOM.IRobotParamServer.GetAllParams	2171
RobotOM.IRobotParamCollection.GetName	2177	RobotOM.IRobotParamServer.GetAllParamsForSchema	2171
RobotOM.IRobotParamCollection.GetValue	2177	RobotOM.IRobotParamServer.GetObjectsWithParam	2171
RobotOM.IRobotParamDef	2185	RobotOM.IRobotParamServer.GetObjectsWithParamVal	2172
RobotOM.IRobotParamDef.DefaultValue	2187	RobotOM.IRobotParamServer.GetParam	2172
RobotOM.IRobotParamDef.GuiGetName	2189	RobotOM.IRobotParamServer.ParamUniqueIdToName	2173
RobotOM.IRobotParamDef.GuiReadOnly	2187	RobotOM.IRobotParamServer.RemoveParam	2173
RobotOM.IRobotParamDef.GuiSetName	2189	RobotOM.IRobotParamServer.RemoveSchemaParams	2173
RobotOM.IRobotParamDef.GuiVisible	2187	RobotOM.IRobotParamServer.ResetParam	2174
RobotOM.IRobotParamDef.Name	2188	RobotOM.IRobotParamServer.Schemas	2170
RobotOM.IRobotParamDef.UniqueId	2188	RobotOM.IRobotParamServer_SetParam	2174
RobotOM.IRobotParamDef.ValueList	2188	RobotOM.IRobotParamValue_Type	2190
RobotOM.IRobotParamDef.ValueType	2188	RobotOM.IRobotPointAuxiliaryRecordValues	44
RobotOM.IRobotParamSchema	2190	RobotOM.IRobotPointsArray	1649
RobotOM.IRobotParamSchema.Def	2191	RobotOM.IRobotPointsArray.Count	1650
RobotOM.IRobotParamSchema.GetAllParams	2192		

RobotOM.IRobotPointsArray.Get 1650	RobotOM.IRobotProject.ActiveModel 1336
RobotOM.IRobotPointsArray.Set 1651	RobotOM.IRobotProject.AxisMngr 1337
RobotOM.IRobotPointsArray.GetSize 1651	RobotOM.IRobotProject.Backgrounds 1337
RobotOM.IRobotPredefinedLabel 28	RobotOM.IRobotProject.CalcEngine 1337
RobotOM.IRobotPredefinedSelection 1035	RobotOM.IRobotProject.Close 1342
RobotOM.IRobotPreferences 1533	RobotOM.IRobotProject.ComponentMngr 1337
RobotOM.IRobotPreferences.CalculationsType 1534	RobotOM.IRobotProject.ConcrReinfEngine 1338
RobotOM.IRobotPreferences.CloudCalculationsEnabled 1534	RobotOM.IRobotProject.Connections 1338
RobotOM.IRobotPreferences.GetDirectory 1536	RobotOM.IRobotProject.DimServer 1338
RobotOM.IRobotPreferences.GetLanguage 1536	RobotOM.IRobotProject.ExtFileName 1338
RobotOM.IRobotPreferences.Multiprocessing 1535	RobotOM.IRobotProject.ExtFileParams 1339
RobotOM.IRobotPreferences.OpenGL 1535	RobotOM.IRobotProject.FileInsertParams 1339
RobotOM.IRobotPreferencesEvents 1538	RobotOM.IRobotProject.FileName 1339
RobotOM.IRobotPreferencesEvents.OnDialogOK 1539	RobotOM.IRobotProject.InsertExtFile 1343
RobotOM.IRobotPressureRecordValues 45	RobotOM.IRobotProject.IsActive 1339
RobotOM.IRobotPrintable 1470	RobotOM.IRobotProject.Name 1340
RobotOM.IRobotPrintable.Comment 1471	RobotOM.IRobotProject.New 1343
RobotOM.IRobotPrintable.IncludeDateTime 1471	RobotOM.IRobotProject.NewFromTemplate 1343
RobotOM.IRobotPrintable.SaveToFile 1472	RobotOM.IRobotProject.Open 1344
RobotOM.IRobotPrintable.StartFromNewPage 1472	RobotOM.IRobotProject.OpenExtFile 1344
RobotOM.IRobotPrintable.Title 1472	RobotOM.IRobotProject.Preferences 1340
RobotOM.IRobotPrintEngine 1473	RobotOM.IRobotProject.PrintEngine 1340
RobotOM.IRobotPrintEngine.AddScToReport 1476	RobotOM.IRobotProject.ReadExtFileParams 1344
RobotOM.IRobotPrintEngine.AddTemplateToReport 1476	RobotOM.IRobotProject.Save 1345
RobotOM.IRobotPrintEngine.AddToReport 1477	RobotOM.IRobotProject.SaveAs 1345
RobotOM.IRobotPrintEngine.ClosePreview 1477	RobotOM.IRobotProject.SaveAsExtFile 1345
RobotOM.IRobotPrintEngine.CreateReportFromOrganizer 1477	RobotOM.IRobotProject.SaveToFormat 1345
RobotOM.IRobotPrintEngine.ExternalPreviewReport 1478	RobotOM.IRobotProject.Structure 1341
RobotOM.IRobotPrintEngine.IsWhilePreview 1474	RobotOM.IRobotProject.Type 1341
RobotOM.IRobotPrintEngine.OrganizerItems 1474	RobotOM.IRobotProject.UniqueId 1341
RobotOM.IRobotPrintEngine.PageSetup 1475	RobotOM.IRobotProject.ViewMngr 1341
RobotOM.IRobotPrintEngine.PreviewReport 1478	RobotOM.IRobotProjectComponent 1262
RobotOM.IRobotPrintEngine.PrintReport 1478	RobotOM.IRobotProjectComponent.Data 1263
RobotOM.IRobotPrintEngine.RemoveFromReport 1478	RobotOM.IRobotProjectComponent.Name 1263
RobotOM.IRobotPrintEngine.RemoveScFromReport 1479	RobotOM.IRobotProjectComponent.Type 1263
RobotOM.IRobotPrintEngine.ReportTemplates 1475	RobotOM.IRobotProjectComponentMngr 1263
RobotOM.IRobotPrintEngine.ResetReport 1479	RobotOM.IRobotProjectComponentMngr.Create 1265
RobotOM.IRobotPrintEngine.SaveReportToFile 1479	RobotOM.IRobotProjectComponentMngr.Get 1265
RobotOM.IRobotPrintEngine.SaveReportToOrganizer 1479	RobotOM.IRobotProjectComponentMngr.GetLevelName 1266
RobotOM.IRobotPrintEngine.SaveReportToTemplate 1480	RobotOM.IRobotProjectComponentMngr.LevelCount 1264
RobotOM.IRobotPrintEngine.ScreenCaptures 1475	RobotOM.IRobotProjectComponentMngr.StdLevelName 1265
RobotOM.IRobotProject 1334	RobotOM.IRobotProjectComponentType 1261

RobotOM.IRobotProjectEvents 1350	RobotOM.IRobotPushOverAnalysisParams.MaxDisplacement 390
RobotOM.IRobotProjectEvents.OnClose 1350	RobotOM.IRobotPushOverAnalysisParams.Node 390
RobotOM.IRobotProjectEvents.OnSave 1351	RobotOM.IRobotPushOverAnalysisParams.Nonlinearm 391
RobotOM.IRobotProjectPreferences 1288	RobotOM.IRobotPushOverAnalysisParams.NonlinearmParams 391
RobotOM.IRobotProjectPreferences.CalcModelCoherence 1290	RobotOM.IRobotPushOverDirection 392
RobotOM.IRobotProjectPreferences.EurocodeFactors 1290	RobotOM.IRobotPushOverLoadDefinitionMethod 391
RobotOM.IRobotProjectPreferences.GetActiveCode 1294	RobotOM.IRobotQuitOption 1537
RobotOM.IRobotProjectPreferences.GetActiveCodeNumber 1295	RobotOM.IRobotReactionData 880
RobotOM.IRobotProjectPreferences.GetCurrentDatabase 1295	RobotOM.IRobotReactionData.FX 881
RobotOM.IRobotProjectPreferences.KinematicConstraints 1291	RobotOM.IRobotReactionData.FY 881
RobotOM.IRobotProjectPreferences.Materials 1291	RobotOM.IRobotReactionData.FZ 881
RobotOM.IRobotProjectPreferences.MeshAutoAdjust 1291	RobotOM.IRobotReactionData.MX 881
RobotOM.IRobotProjectPreferences.MeshAutoAdjustIterationCount 1291	RobotOM.IRobotReactionData.MY 881
RobotOM.IRobotProjectPreferences.MeshParams 1292	RobotOM.IRobotReactionData.MZ 881
RobotOM.IRobotProjectPreferences.MeshParamsFloors 1292	RobotOM.IRobotReactionServer 849
RobotOM.IRobotProjectPreferences.MeshParamsWalls 1292	RobotOM.IRobotReactionServer.DDC 852
RobotOM.IRobotProjectPreferences.Save 1295	RobotOM.IRobotReactionServer.DDCEx 852
RobotOM.IRobotProjectPreferences.SectionsActive 1293	RobotOM.IRobotReactionServer.DDCLocal 852
RobotOM.IRobotProjectPreferences.SectionsFound 1293	RobotOM.IRobotReactionServer.DDCLocalEx 852
RobotOM.IRobotProjectPreferences.SetActiveCode 1296	RobotOM.IRobotReactionServer.DDCSum 853
RobotOM.IRobotProjectPreferences.SetActiveCodeNumber 1296	RobotOM.IRobotReactionServer.DDCSumEx 853
RobotOM.IRobotProjectPreferences.SetCurrentDatabase 1296	RobotOM.IRobotReactionServer.DynCombDDC 853
RobotOM.IRobotProjectPreferences.Units 1293	RobotOM.IRobotReactionServer.DynCombDDCLocal 854
RobotOM.IRobotProjectPreferences.VehiclesActive 1293	RobotOM.IRobotReactionServer.DynCombDDCSum 854
RobotOM.IRobotProjectPreferences.VehiclesFound 1294	RobotOM.IRobotReactionServer.DynCombLocal 854
RobotOM.IRobotProjectPreferencesEvents 1307	RobotOM.IRobotReactionServer.DynCombSumForce 855
RobotOM.IRobotProjectPreferencesEvents.OnDialogOK 1307	RobotOM.IRobotReactionServer.DynCombValue 855
RobotOM.IRobotProjectSaveFormat 1349	RobotOM.IRobotReactionServer.DynDDC 855
RobotOM.IRobotProjectType 1333	RobotOM.IRobotReactionServer.DynDDCLocal 856
RobotOM.IRobotProtectionInfo 1644	RobotOM.IRobotReactionServer.DynDDCSum 856
RobotOM.IRobotProtectionInfo.IsEnabled 1645	RobotOM.IRobotReactionServer.DynLocal 856
RobotOM.IRobotPseudostaticForceServer 863	RobotOM.IRobotReactionServer.DynSum 856
RobotOM.IRobotPseudostaticForceServer.CombValue 863	RobotOM.IRobotReactionServer.DynSumForce 857
RobotOM.IRobotPseudostaticForceServer.Value 864	RobotOM.IRobotReactionServer.DynValue 857
RobotOM.IRobotPushOverAnalysisParams 388	RobotOM.IRobotReactionServer.Local 857
RobotOM.IRobotPushOverAnalysisParams.Direction 389	RobotOM.IRobotReactionServer.LocalEx 857
RobotOM.IRobotPushOverAnalysisParams.LoadDefinition 390	RobotOM.IRobotReactionServer.Sum 858
	RobotOM.IRobotReactionServer.SumEx 858
	RobotOM.IRobotReactionServer.SumForce 858
	RobotOM.IRobotReactionServer.SumForceEx 859

RobotOM.IRobotReactionServer.Value 859	RobotOM.IRobotResultRow.GetValue 1018
RobotOM.IRobotReactionServer.ValueEx 859	RobotOM.IRobotResultRow.GetValueType 1019
RobotOM.IRobotReinforceCalcMethods 951	RobotOM.IRobotResultRow.IsAvailable 1019
RobotOM.IRobotReportItem 1501	RobotOM.IRobotResultRowSet 1021
RobotOM.IRobotReportItem.CreateView 1506	RobotOM.IRobotResultRowSet.Clear 1023
RobotOM.IRobotReportItem.ExcludeFromReport 1503	RobotOM.IRobotResultRowSet.CurrentRow 1022
RobotOM.IRobotReportItem.GetPageTemplate 1506	RobotOM.IRobotResultRowSet.MoveFirst 1023
RobotOM.IRobotReportItem.HasNoteAfter 1503	RobotOM.IRobotResultRowSet.MoveNext 1023
RobotOM.IRobotReportItem.HasNoteBefore 1503	RobotOM.IRobotResultRowSet.ResultIds 1022
RobotOM.IRobotReportItem.IncludeDateTime 1504	RobotOM.IRobotResultServer 997
RobotOM.IRobotReportItem.NoteAfter 1504	RobotOM.IRobotResultServer.Advanced 998
RobotOM.IRobotReportItem.NoteBefore 1504	RobotOM.IRobotResultServer.Any 998
RobotOM.IRobotReportItem.StartFromNewPage 1505	RobotOM.IRobotResultServer.Bars 999
RobotOM.IRobotReportItem.Title 1505	RobotOM.IRobotResultServer.CalculationResume 999
RobotOM.IRobotReportItem.TitleText 1505	RobotOM.IRobotResultServer.Extremes 999
RobotOM.IRobotReportItem.Type 1505	RobotOM.IRobotResultServer.FiniteElems 1000
RobotOM.IRobotReportItemList 1500	RobotOM.IRobotResultServer.Nodes 1000
RobotOM.IRobotReportItemList.Count 1501	RobotOM.IRobotResultServer.Query 1001
RobotOM.IRobotReportItemList.Get 1501	RobotOM.IRobotResultServer.Status 1000
RobotOM.IRobotReportItemType 1510	RobotOM.IRobotResultServer.Storeys 1000
RobotOM.IRobotReportStdElementRtf 1497	RobotOM.IRobotResultServer.Total 1001
RobotOM.IRobotReportStdElementRtf.Active 1498	RobotOM.IRobotResultStatusType 1027
RobotOM.IRobotReportStdElementRtf.Frame 1498	RobotOM.IRobotRtfView 1455
RobotOM.IRobotReportStdElementRtf.LoadFromFile 1499	RobotOM.IRobotRtfView.AppendFromFile 1457
RobotOM.IRobotReportStdElementRtf.RestoreDefaults 1499	RobotOM.IRobotRtfView.Evaluate 1457
RobotOM.IRobotReportStdElementRtf.SaveToFile 1500	RobotOM.IRobotRtfView.LoadFromFile 1457
RobotOM.IRobotReportTemplateMngr 1495	RobotOM.IRobotRtfView.Printable 1456
RobotOM.IRobotReportTemplateMngr.Count 1496	RobotOM.IRobotRtfView.SaveToFile 1458
RobotOM.IRobotReportTemplateMngr.Find 1496	RobotOM.IRobotScreenCaptureMngr 1458
RobotOM.IRobotReportTemplateMngr.Get 1497	RobotOM.IRobotScreenCaptureMngr.Count 1459
RobotOM.IRobotReportTemplateMngr.Remove 1497	RobotOM.IRobotScreenCaptureMngr.Find 1459
RobotOM.IRobotResultParamType 1019	RobotOM.IRobotScreenCaptureMngr.Get 1460
RobotOM.IRobotResultQueryParams 1024	RobotOM.IRobotScreenCaptureMngr.Remove 1460
RobotOM.IRobotResultQueryParams.GetParam 1026	RobotOM.IRobotSectionDatabase 1304
RobotOM.IRobotResultQueryParams.IsParamSet 1026	RobotOM.IRobotSectionDatabase.Description 1305
RobotOM.IRobotResultQueryParams.Reset 1026	RobotOM.IRobotSectionDatabase.FullName 1305
RobotOM.IRobotResultQueryParams.ResultIds 1025	RobotOM.IRobotSectionDatabase.GetAll 1306
RobotOM.IRobotResultQueryParams.Selection 1025	RobotOM.IRobotSectionDatabase.Load 1306
RobotOM.IRobotResultQueryParams.SetParam 1027	RobotOM.IRobotSectionDatabase.LoadFromFile 1307
RobotOM.IRobotResultQueryReturnType 1027	RobotOM.IRobotSectionDatabase.Name 1305
RobotOM.IRobotResultRow 1017	RobotOM.IRobotSectionDatabaseList 1297
RobotOM.IRobotResultRow.GetParam 1018	RobotOM.IRobotSectionDatabaseList.Add 1298

RobotOM.IRobotSectionDatabaseList.AddFromFile 1299	215
RobotOM.IRobotSectionDatabaseList.ChangeIndex 1299	
RobotOM.IRobotSectionDatabaseList.Count 1298	
RobotOM.IRobotSectionDatabaseList.Find 1299	
RobotOM.IRobotSectionDatabaseList.Get 1299	
RobotOM.IRobotSectionDatabaseList.GetDatabase 1300	
RobotOM.IRobotSectionDatabaseList.Remove 1300	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params 210	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.BehaviorFactor 211	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.Direction 211	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.DirectionType 211	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.Filter 212	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.ResidualMode 212	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.Site 212	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.SpectrumType 212	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.StructureType 213	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.Topography 213	
RobotOM.IRobotSeismicAnalysis_AFPS_90_Params.ZoneType 213	
RobotOM.IRobotSeismicAnalysis_AFPS_90_SiteType 254	
RobotOM.IRobotSeismicAnalysis_AFPS_90_StructureType 254	
RobotOM.IRobotSeismicAnalysis_AFPS_90_ZoneType 250	
RobotOM.IRobotSeismicAnalysis_CHINESE_DesignType 256	
RobotOM.IRobotSeismicAnalysis_CHINESE_EarthquakeType 257	
RobotOM.IRobotSeismicAnalysis_CHINESE_IntensityType 256	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params 214	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.DesignStandard 215	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.Direction 215	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.EarthquakeType 215	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.Factor 216	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.Filter 216	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.Intensity 216	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.Site 217	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.SiteTg 217	
RobotOM.IRobotSeismicAnalysis_CHINESE_Params.StructureType 217	
RobotOM.IRobotSeismicAnalysis_CHINESE_SiteType 256	
RobotOM.IRobotSeismicAnalysis_CHINESE_StructureType 255	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params 217	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.Direction 218	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.DirectionType 219	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.Filter 219	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.PlasticityCoeff 219	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.Soil 220	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.StructureType 220	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_Params.ZoneType 220	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_SoilType 257	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_StructureType 258	
RobotOM.IRobotSeismicAnalysis_CIRSOC_103_ZoneType 251	
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_Params 221	
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_Params.Direction 221	
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_Params.Filter 221	

r 222	aviorFactor 295
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_Params.Seis micCoeff 222	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Dire ction 295
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_Params.Seis micProtectionCoeff 222	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Dire ctionType 296
RobotOM.IRobotSeismicAnalysis_DM_16_1_96_ProtectionCo effType 259	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Exci tationDir 296
RobotOM.IRobotSeismicAnalysis_EAK_2000_GroundCategor yType 265	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Filte r 296
RobotOM.IRobotSeismicAnalysis_EAK_2000_ImportanceFact orType 264	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Resi dualMode 296
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params 246	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.S 297
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Behavi orFactor 247	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Spe ctrum 297
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Directi on 247	RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Tb 297
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Directi onType 248	RobotOM.IRobotSeismicAnalysis(EC8_General_Params.Tc 297
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Filter 248	RobotOM.IRobotSeismicAnalysis(EC8_General_Params.Td 298
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Found ationFactor 248	RobotOM.IRobotSeismicAnalysis(EC8_GroundType 298
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Ground Category 249	RobotOM.IRobotSeismicAnalysis(EC8_Params 299
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Import anceFactor 249	RobotOM.IRobotSeismicAnalysis(EC8_Params.Ag 300
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Vertical BehaviorFactor 249	RobotOM.IRobotSeismicAnalysis(EC8_Params.BehaviorFact or 300
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.Vertical FoundationFactor 249	RobotOM.IRobotSeismicAnalysis(EC8_Params.Direction 300
RobotOM.IRobotSeismicAnalysis_EAK_2000_Params.ZoneT ype 250	RobotOM.IRobotSeismicAnalysis(EC8_Params.DirectionType 301
RobotOM.IRobotSeismicAnalysis_EAK_2000_ZoneType 253	RobotOM.IRobotSeismicAnalysis(EC8_Params.ExcitationDir 301
RobotOM.IRobotSeismicAnalysis_EC8_General_Params 293	RobotOM.IRobotSeismicAnalysis(EC8_Params.Filter 301
RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Ag 294	RobotOM.IRobotSeismicAnalysis(EC8_Params.GroundType 301
RobotOM.IRobotSeismicAnalysis_EC8_General_Params.B 295	RobotOM.IRobotSeismicAnalysis(EC8_Params.ResidualMod e 302
RobotOM.IRobotSeismicAnalysis_EC8_General_Params.Beh	RobotOM.IRobotSeismicAnalysis(EC8_Params.Spectrum 302
	RobotOM.IRobotSeismicAnalysis(EC8_Params.SpectrumTyp e 302
	RobotOM.IRobotSeismicAnalysis(EC8_SpectrumType 298

RobotOM.IRobotSeismicAnalysis_IBC_2000_Params	242	274
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.BehaviorFactor	243	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.FactorQ
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.Direction	244	274
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.ExcitationDir	244	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.Filter
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.Filter	244	274
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.le	245	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.Soil
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.S1	245	274
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.SiteClass	245	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.Spectrum
RobotOM.IRobotSeismicAnalysis_IBC_2000_Params.Ss	246	275
RobotOM.IRobotSeismicAnalysis_IBC_2000_SiteClassType	264	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.Zone
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params	279	275
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.BehaviorFactor	280	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_SoilType
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.Direction	281	275
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.ExcitationDir	281	RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_ZoneType
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.Filter	281	275
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.I	281	RobotOM.IRobotSeismicAnalysis_P_100_2006_Params
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.S1	282	276
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.SiteClass	282	RobotOM.IRobotSeismicAnalysis_P_100_2006_Params.Agg
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.Ss	282	277
RobotOM.IRobotSeismicAnalysis_IBC_2006_Params.TL	283	RobotOM.IRobotSeismicAnalysis_P_100_2006_Params.B0
RobotOM.IRobotSeismicAnalysis_IBC_2006_SiteClassType	283	277
RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Direction	276	RobotOM.IRobotSeismicAnalysis_P_100_2006_Params.BehaviorFactor
RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params	273	277
RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.Direction	274	RobotOM.IRobotSeismicAnalysis_P_100_2006_Params.ExcitationDir
RobotOM.IRobotSeismicAnalysis_ITALY_ORDINANZA_Params.ExcitationDir		278

224	287
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.Excitati onDir 224	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Spec trumType 287
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.Filter 224	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Struc tureType 288
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.Importa nceClass 224	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Topo graphy 288
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.Psi 225	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Zone Type 288
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.Tc 225	RobotOM.IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope 290
RobotOM.IRobotSeismicAnalysis_P_100_92_Params.ZoneTy pe 225	RobotOM.IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope .IsActive 291
RobotOM.IRobotSeismicAnalysis_P_100_92_ZoneType 251	RobotOM.IRobotSeismicAnalysis_PS_92_2008_SiteEnvelope .SetActive 291
RobotOM.IRobotSeismicAnalysis_PS_69_DampingType 259	RobotOM.IRobotSeismicAnalysis_PS_92_2008_SiteType 289
RobotOM.IRobotSeismicAnalysis_PS_69_Params 225	RobotOM.IRobotSeismicAnalysis_PS_92_2008_StructureTyp e 289
RobotOM.IRobotSeismicAnalysis_PS_69_Params.Alpha 226	RobotOM.IRobotSeismicAnalysis_PS_92_2008_ZoneType 289
RobotOM.IRobotSeismicAnalysis_PS_69_Params.Damping 227	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params 228
RobotOM.IRobotSeismicAnalysis_PS_69_Params.Delta 227	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Behav iorFactor 229
RobotOM.IRobotSeismicAnalysis_PS_69_Params.Filter 228	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Direction 230
RobotOM.IRobotSeismicAnalysis_PS_69_Params.Soil 228	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.DirectionTy pe 230
RobotOM.IRobotSeismicAnalysis_PS_69_SoilType 260	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Excitatio nDir 230
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params 284	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Filter 231
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Ag 285	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.ResidualM ode 231
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Beha viorFactor 285	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Site 231
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Dire ction 285	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.SiteEnv elope 231
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Direc tionType 286	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.SpectrumT ype 232
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Excit ationDir 286	RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.StructureT ype 232
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Filter 286	
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Resi dualMode 287	
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Site 287	
RobotOM.IRobotSeismicAnalysis_PS_92_2008_Params.Site Envelope	

RobotOM.IRobotSeismicAnalysis_PS_92_Params.Topography 232	RobotOM.IRobotSeismicAnalysis_RPA_88_Params.Soil 235
RobotOM.IRobotSeismicAnalysis_PS_92_Params.ZoneType 233	RobotOM.IRobotSeismicAnalysis_RPA_88_Params.ZoneType 235
RobotOM.IRobotSeismicAnalysis_PS_92_SiteEnvelope 265	RobotOM.IRobotSeismicAnalysis_RPA_88_Params.ZoneType 236
RobotOM.IRobotSeismicAnalysis_PS_92_SiteEnvelope.IsActive 266	RobotOM.IRobotSeismicAnalysis_RPA_88_SoilType 262
RobotOM.IRobotSeismicAnalysis_PS_92_SiteEnvelope.SetActive 266	RobotOM.IRobotSeismicAnalysis_RPA_88_UsageType 261
RobotOM.IRobotSeismicAnalysis_PS_92_SiteType 261	RobotOM.IRobotSeismicAnalysis_RPA_88_ZoneType 252
RobotOM.IRobotSeismicAnalysis_PS_92_StructureType 260	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params 267
RobotOM.IRobotSeismicAnalysis_PS_92_ZoneType 252	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.BehaviorFactor 268
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params 270	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.DirectionType 268
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.BehaviorFactor 271	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.ExcitationDir 268
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.ExcitationDir 271	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.Filter 269
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.Filter 272	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.ResidualMode 269
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.QualityCoef 272	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.Site 269
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.ResidualMode 272	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.StructureClass 269
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.Site 272	RobotOM.IRobotSeismicAnalysis_RPS_2000_Params.Zone 270
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.Usage 272	RobotOM.IRobotSeismicAnalysis_RPS_2000_SiteType 267
RobotOM.IRobotSeismicAnalysis_RPA_2003_Params.Zone 273	RobotOM.IRobotSeismicAnalysis_RPS_2000_StructureClass 267
RobotOM.IRobotSeismicAnalysis_RPA_2003_SiteType 270	RobotOM.IRobotSeismicAnalysis_RPS_2000_ZoneType 267
RobotOM.IRobotSeismicAnalysis_RPA_2003_UsageType 270	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params 236
RobotOM.IRobotSeismicAnalysis_RPA_2003_ZoneType 270	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.BehaviorFactor 237
RobotOM.IRobotSeismicAnalysis_RPA_88_CategoryType 262	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.Direction 237
RobotOM.IRobotSeismicAnalysis_RPA_88_Params 233	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.Filter 238
RobotOM.IRobotSeismicAnalysis_RPA_88_Params.Category 234	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.SoilType 238
RobotOM.IRobotSeismicAnalysis_RPA_88_Params.Direction 234	RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.StructureImportance 238
RobotOM.IRobotSeismicAnalysis_RPA_88_Params.Filter 234	
RobotOM.IRobotSeismicAnalysis_RPA_88_Params.QualityFactor 235	

RobotOM.IRobotSeismicAnalysis_TURKISH_23098_Params.ZoneType 239	RobotOM.IRobotSeismicResultsSaveParams.NMQ 1574
RobotOM.IRobotSeismicAnalysis_TURKISH_23098_SoilType 263	RobotOM.IRobotSeismicResultsSaveParams.OnlyQuadraticCombs 1574
RobotOM.IRobotSeismicAnalysis_TURKISH_23098_ZoneType 253	RobotOM.IRobotSeismicResultsSaveParams.PanelsDir 1574
RobotOM.IRobotSeismicAnalysis_UBC_97_Params 239	RobotOM.IRobotSeismicResultsSaveParams.PointNumber 1575
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.BehaviorFactor 240	RobotOM.IRobotSeismicResultsSaveParams.Reactions 1575
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.ClosestDistance 240	RobotOM.IRobotSeismicResultsSaveParams.Reduced 1575
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.Direction 240	RobotOM.IRobotSeismicResultsSaveParams.Save 1576
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.ExcitationDir 241	RobotOM.IRobotSeismicResultsSaveParams.Stresses 1576
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.Filter 241	RobotOM.IRobotSelection 3
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.I 241	RobotOM.IRobotSelection.Add 5
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.Soil 242	RobotOM.IRobotSelection.AddOne 5
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.Source 242	RobotOM.IRobotSelection.AddText 6
RobotOM.IRobotSeismicAnalysis_UBC_97_Params.ZoneType 242	RobotOM.IRobotSelection.And 6
RobotOM.IRobotSeismicAnalysis_UBC_97_SoilType 263	RobotOM.IRobotSelection.AndText 6
RobotOM.IRobotSeismicAnalysis_UBC_97_SourceType 264	RobotOM.IRobotSelection.Clear 7
RobotOM.IRobotSeismicAnalysis_UBC_97_ZoneType 252	RobotOM.IRobotSelection.Contains 7
RobotOM.IRobotSeismicAnalysisDirectionType 258	RobotOM.IRobotSelection.Count 4
RobotOM.IRobotSeismicAnalysisSpectrumType 255	RobotOM.IRobotSelection.Exclude 7
RobotOM.IRobotSeismicResidualMode 292	RobotOM.IRobotSelection.ExcludeOne 7
RobotOM.IRobotSeismicResidualMode.AugmentationFactor 292	RobotOM.IRobotSelection.ExcludeText 8
RobotOM.IRobotSeismicResidualMode.DefinitionMethod 293	RobotOM.IRobotSelection.FromText 8
RobotOM.IRobotSeismicResidualMode.IsActive 293	RobotOM.IRobotSelection.Get 8
RobotOM.IRobotSeismicResidualMode.LimitFrequency 293	RobotOM.IRobotSelection.ToText 9
RobotOM.IRobotSeismicResidualModeDefinitionType 291	RobotOM.IRobotSelection.Type 4
RobotOM.IRobotSeismicResultsPanelDirection 1576	RobotOM.IRobotSelectionFactory 1031
RobotOM.IRobotSeismicResultsSaveParams 1571	RobotOM.IRobotSelectionFactory.Create 1032
RobotOM.IRobotSeismicResultsSaveParams.Displacements 1573	RobotOM.IRobotSelectionFactory.CreateByLabel 1033
RobotOM.IRobotSeismicResultsSaveParams.Fluxes 1573	RobotOM.IRobotSelectionFactory.CreateByStorey 1033
RobotOM.IRobotSeismicResultsSaveParams.LocalDisplacements 1573	RobotOM.IRobotSelectionFactory.CreateEdgeSelection 1033
	RobotOM.IRobotSelectionFactory.CreateFull 1034
	RobotOM.IRobotSelectionFactory.CreateMulti 1034
	RobotOM.IRobotSelectionFactory.CreatePredefined 1034
	RobotOM.IRobotSelectionFactory.Get 1034
	RobotOM.IRobotSELFSeismic_AS_1170_4_Params 463
	RobotOM.IRobotSELFSeismic_AS_1170_4_Params.Kp 464
	RobotOM.IRobotSELFSeismic_AS_1170_4_Params.Mi 464
	RobotOM.IRobotSELFSeismic_AS_1170_4_Params.Probability 465
	RobotOM.IRobotSELFSeismic_AS_1170_4_Params.SoilCategory 465

RobotOM.IRobotSELFSeismic_AS_1170_4_Params.Sp 465	RobotOM.IRobotSELFSeismic_EC_8_Params.SpectrumType 460
RobotOM.IRobotSELFSeismic_AS_1170_4_Params.StructureTypeX 465	RobotOM.IRobotSELFSeismic_EC_8_Params.StructureTypeX 461
RobotOM.IRobotSELFSeismic_AS_1170_4_Params.StructureTypeY 466	RobotOM.IRobotSELFSeismic_EC_8_Params.StructureTypeY 461
RobotOM.IRobotSELFSeismic_AS_1170_4_Params.Z 466	RobotOM.IRobotSELFSeismic_EC_8_SiteClass 462
RobotOM.IRobotSELFSeismic_AS_1170_4_ProbabilityType 467	RobotOM.IRobotSELFSeismic_EC_8_SpectrumType 463
RobotOM.IRobotSELFSeismic_AS_1170_4_SoilCategoryType 466	RobotOM.IRobotSELFSeismic_EC_8_StructureType 462
RobotOM.IRobotSELFSeismic_AS_1170_4_StructureType 467	RobotOM.IRobotSELFSeismicAnalysisParams 451
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params 454	RobotOM.IRobotSELFSeismicAnalysisParams.CodeName 452
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.I 455	RobotOM.IRobotSELFSeismicAnalysisParams.CodeNumber 452
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.R 456	RobotOM.IRobotSELFSeismicAnalysisParams.CodeParams 452
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.S1 456	RobotOM.IRobotSELFSeismicAnalysisParams.Eccentricities 453
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.SiteClass 456	RobotOM.IRobotSELFSeismicAnalysisParams.GetExcitationDir 454
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.Ss 456	RobotOM.IRobotSELFSeismicAnalysisParams.StructureParams 453
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.StructureTypeX 457	RobotOM.IRobotSELFSeismicAnalysisParams.TBaseMethod 453
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.StructureTypeY 457	RobotOM.IRobotSELFSeismicEngine 444
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.TBaseValueX 457	RobotOM.IRobotSELFSeismicEngine.GenerateLoadCases 445
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.TBaseValueY 457	RobotOM.IRobotSELFSeismicEngine.GenerationParams 444
RobotOM.IRobotSELFSeismic_ASCE_7_10_Params.TL 458	RobotOM.IRobotSELFSeismicGenerationParams 445
RobotOM.IRobotSELFSeismic_ASCE_7_10_SiteClassType 454	RobotOM.IRobotSELFSeismicGenerationParams.CodeName 446
RobotOM.IRobotSELFSeismic_ASCE_7_10_StructureType 458	RobotOM.IRobotSELFSeismicGenerationParams.CodeNumber 446
RobotOM.IRobotSELFSeismic_EC_8_Params 458	RobotOM.IRobotSELFSeismicGenerationParams.ExcitationDir 447
RobotOM.IRobotSELFSeismic_EC_8_Params.Ag 459	RobotOM.IRobotSELFSeismicGenerationParams.ModalCaseParams 447
RobotOM.IRobotSELFSeismic_EC_8_Params.Beta 460	RobotOM.IRobotSELFSeismicGenerationParams.SeismicParams 447
RobotOM.IRobotSELFSeismic_EC_8_Params.Q 460	RobotOM.IRobotSELFSeismicGenerationParams.TBaseMethod 447
RobotOM.IRobotSELFSeismic_EC_8_Params.SiteClass 460	RobotOM.IRobotSELFSeismicLevelDefinitionMethod 450
RobotOM.IRobotSELFSeismic_EC_8_Params.SiteClassEnvelopeCheck 461	RobotOM.IRobotSELFSeismicStructureParams 448
RobotOM.IRobotSELFSeismic_EC_8_Params.SiteClassEnvelopeIsChecked 462	

RobotOM.IRobotSELFSeismicStructureParams.BaseLevelCoordZ 448	RobotOM.IRobotSnowWindParams.IsSnow 94 RobotOM.IRobotSnowWindParams.IsWind 94 RobotOM.IRobotSnowWindParams.TotalDepth 94 RobotOM.IRobotSnowWindParams.WithCavities 95 RobotOM.IRobotSnowWindParams.WithParapets 95 RobotOM.IRobotSolidPropertiesData 735 RobotOM.IRobotSolidPropertiesData.DampCoef 736 RobotOM.IRobotSolidPropertiesData.E 737 RobotOM.IRobotSolidPropertiesData.LoadFromDBase 738 RobotOM.IRobotSolidPropertiesData.LX 737 RobotOM.IRobotSolidPropertiesData.MaterialModel 737 RobotOM.IRobotSolidPropertiesData.NU 738 RobotOM.IRobotSolidPropertiesData.RO 738 RobotOM.IRobotSparseMSolverMethod 1565 RobotOM.IRobotSparseMSolverParams 1565 RobotOM.IRobotSparseMSolverParams.Method 1565 RobotOM.IRobotSpectralAnalysisAbscissaXAxisType 313 RobotOM.IRobotSpectralAnalysisOrdinateYAxisType 314 RobotOM.IRobotSpectralAnalysisParams 303 RobotOM.IRobotSpectralAnalysisParams.Direction 304 RobotOM.IRobotSpectralAnalysisParams.ExcitationDir 304 RobotOM.IRobotSpectralAnalysisParams.Filter 304 RobotOM.IRobotSpectralAnalysisParams.Spectrum 304 RobotOM.IRobotSpectralAnalysisPointsCollection 310 RobotOM.IRobotSpectralAnalysisPointsCollection.Add 311 RobotOM.IRobotSpectralAnalysisPointsCollection.Clear 311 RobotOM.IRobotSpectralAnalysisPointsCollection.Count 310 RobotOM.IRobotSpectralAnalysisPointsCollection.Get 312 RobotOM.IRobotSpectralAnalysisPointsCollection.LoadFromFile 312 RobotOM.IRobotSpectralAnalysisPointsCollection.Remove 312 RobotOM.IRobotSpectralAnalysisPointsCollection.SaveToFile 313 RobotOM.IRobotSpectralAnalysisPointsCollection.Set 313 RobotOM.IRobotSpectralAnalysisSpectrum 305 RobotOM.IRobotSpectralAnalysisSpectrum.AbscissaXAxis 306 RobotOM.IRobotSpectralAnalysisSpectrum.AbscissaXAxisLog arithmicScale 306 RobotOM.IRobotSpectralAnalysisSpectrum.AddFromTimeHist ory 308 RobotOM.IRobotSpectralAnalysisSpectrum.Average 309
RobotOM.IRobotSELFSeismicStructureParams.BaseLevelDef Method 449	
RobotOM.IRobotSELFSeismicStructureParams.BaseLevelSto rey 449	
RobotOM.IRobotSELFSeismicStructureParams.TopLevelCoo dZ 449	
RobotOM.IRobotSELFSeismicStructureParams.TopLevelDef Method 450	
RobotOM.IRobotSELFSeismicStructureParams.TopLevelSto rey 450	
RobotOM.IRobotSELFSeismicTBaseMethod 451	
RobotOM.IRobotSerializable 1651	
RobotOM.IRobotSerializable.Read 1652	
RobotOM.IRobotSerializable.Write 1653	
RobotOM.IRobotSimpleCase 417	
RobotOM.IRobotSimpleCase.GetAnalysisParams 421	
RobotOM.IRobotSimpleCase.GetSeismicCode 421	
RobotOM.IRobotSimpleCase.IsAuxiliary 418	
RobotOM.IRobotSimpleCase.Label 418	
RobotOM.IRobotSimpleCase.MainMode 419	
RobotOM.IRobotSimpleCase.ModesCount 419	
RobotOM.IRobotSimpleCase.NatureName 419	
RobotOM.IRobotSimpleCase.Records 419	
RobotOM.IRobotSimpleCase.SetAnalysisParams 421	
RobotOM.IRobotSimpleCase.SetNatureExt 422	
RobotOM.IRobotSimpleCase.TimeStepCount 420	
RobotOM.IRobotSimpleCase.UniqueId 420	
RobotOM.IRobotSnowWindEngine 95	
RobotOM.IRobotSnowWindEngine.CodeParams 96	
RobotOM.IRobotSnowWindEngine.Generate 97	
RobotOM.IRobotSnowWindEngine.Generate3D 97	
RobotOM.IRobotSnowWindEngine.GenerateStruct3D 97	
RobotOM.IRobotSnowWindEngine.Params 96	
RobotOM.IRobotSnowWindEngine.ShowParamsDlg 98	
RobotOM.IRobotSnowWindParams 92	
RobotOM.IRobotSnowWindParams.BaseOnGround 93	
RobotOM.IRobotSnowWindParams.BaySpacing 93	
RobotOM.IRobotSnowWindParams.Envelope 93	
RobotOM.IRobotSnowWindParams.IsolatedRoofs 93	

RobotOM.IRobotSpectralAnalysisSpectrum.Damping 306	RobotOM.IRobotStoreyMngr.Delete 1055
RobotOM.IRobotSpectralAnalysisSpectrum.LoadFromFile 309	RobotOM.IRobotStoreyMngr.DeleteAll 1056
RobotOM.IRobotSpectralAnalysisSpectrum.Name 307	RobotOM.IRobotStoreyMngr.DisregardedObjects 1054
RobotOM.IRobotSpectralAnalysisSpectrum.OrdinateYAxis 307	RobotOM.IRobotStoreyMngr.FilterStorey 1054
RobotOM.IRobotSpectralAnalysisSpectrum.OrdinateYAxisLog arithmicScale 307	RobotOM.IRobotStoreyMngr.Find 1056
RobotOM.IRobotSpectralAnalysisSpectrum.Points 308	RobotOM.IRobotStoreyMngr.Get 1056
RobotOM.IRobotSpectralAnalysisSpectrum.SaveToFile 309	RobotOM.IRobotStoreyReducedForces 1064
RobotOM.IRobotSpectralCoefficients 925	RobotOM.IRobotStoreyReducedForces.FX 1066
RobotOM.IRobotSpectralCoefficients.ModeCoef 926	RobotOM.IRobotStoreyReducedForces.FX_ToColumns 1066
RobotOM.IRobotSpectralCoefficients.PartCoef 926	RobotOM.IRobotStoreyReducedForces.FX_ToWalls 1066
RobotOM.IRobotSpectralCoefficients.SpectrCoef 927	RobotOM.IRobotStoreyReducedForces.FY 1066
RobotOM.IRobotStorey 1047	RobotOM.IRobotStoreyReducedForces.FY_ToColumns 1067
RobotOM.IRobotStorey.AutomaticSelection 1048	RobotOM.IRobotStoreyReducedForces.FY_ToWalls 1067
RobotOM.IRobotStorey.Color 1049	RobotOM.IRobotStoreyReducedForces.FZ 1067
RobotOM.IRobotStorey.Ex1 1049	RobotOM.IRobotStoreyReducedForces.FZ_ToColumns 1067
RobotOM.IRobotStorey.Ey1 1049	RobotOM.IRobotStoreyReducedForces.FZ_ToWalls 1068
RobotOM.IRobotStorey.Height 1049	RobotOM.IRobotStoreyReducedForces.MX 1068
RobotOM.IRobotStorey.Lx 1050	RobotOM.IRobotStoreyReducedForces.MY 1068
RobotOM.IRobotStorey.Ly 1050	RobotOM.IRobotStoreyReducedForces.MZ 1068
RobotOM.IRobotStorey.Name 1050	RobotOM.IRobotStoreyResultServer 1072
RobotOM.IRobotStorey.Objects 1051	RobotOM.IRobotStoreyResultServer.Displacements 1073
RobotOM.IRobotStorey.SetHeight 1051	RobotOM.IRobotStoreyResultServer.ReducedForces 1073
RobotOM.IRobotStorey.SetTopLevel 1052	RobotOM.IRobotStoreyResultServer.Values 1074
RobotOM.IRobotStorey.TopLevel 1051	RobotOM.IRobotStoreySelection 1056
RobotOM.IRobotStoreyDisplacements 1069	RobotOM.IRobotStoreySelection.Add 1057
RobotOM.IRobotStoreyDisplacements.DrUX 1070	RobotOM.IRobotStoreySelection.AddAll 1058
RobotOM.IRobotStoreyDisplacements.DrUY 1070	RobotOM.IRobotStoreySelection.Clear 1058
RobotOM.IRobotStoreyDisplacements.MaxUX 1070	RobotOM.IRobotStoreySelection.GetNames 1058
RobotOM.IRobotStoreyDisplacements.MaxUY 1070	RobotOM.IRobotStoreySelection.GetNumbers 1059
RobotOM.IRobotStoreyDisplacements.MinUX 1071	RobotOM.IRobotStoreySelection.Remove 1059
RobotOM.IRobotStoreyDisplacements.MinUY 1071	RobotOM.IRobotStoreyValues 1059
RobotOM.IRobotStoreyDisplacements.NodeMaxUX 1071	RobotOM.IRobotStoreyValues.Ex0 1060
RobotOM.IRobotStoreyDisplacements.NodeMaxUY 1071	RobotOM.IRobotStoreyValues.Ex1 1061
RobotOM.IRobotStoreyDisplacements.NodeMinUX 1072	RobotOM.IRobotStoreyValues.Ex2 1061
RobotOM.IRobotStoreyDisplacements.NodeMinUY 1072	RobotOM.IRobotStoreyValues.Ey0 1061
RobotOM.IRobotStoreyMngr 1052	RobotOM.IRobotStoreyValues.Ey1 1062
RobotOM.IRobotStoreyMngr.BaseLevel 1053	RobotOM.IRobotStoreyValues.Ey2 1062
RobotOM.IRobotStoreyMngr.Count 1053	RobotOM.IRobotStoreyValues.F 1062
RobotOM.IRobotStoreyMngr.Create2 1055	RobotOM.IRobotStoreyValues.G 1063
RobotOM.IRobotStoreyMngr.Create2Ex 1055	RobotOM.IRobotStoreyValues.Ix 1063
	RobotOM.IRobotStoreyValues.Iy 1063
	RobotOM.IRobotStoreyValues.Iz 1063

RobotOM.IRobotStoreyValues.Mass 1064	RobotOM.IRobotStructuralAxisGridCartesian.Y 1522
RobotOM.IRobotStoreyValues.R 1064	RobotOM.IRobotStructuralAxisGridCartesian.Z 1523
RobotOM.IRobotSTRFileAnalyser 2206	RobotOM.IRobotStructuralAxisGridMngr 1524
RobotOM.IRobotSTRFileAnalyser.InsertParams 2207	RobotOM.IRobotStructuralAxisGridMngr.Activate 1525
RobotOM.IRobotSTRFileAnalyser.InsertToProject 2208	RobotOM.IRobotStructuralAxisGridMngr.Clear 1525
RobotOM.IRobotSTRFileAnalyser.Params 2207	RobotOM.IRobotStructuralAxisGridMngr.Count 1525
RobotOM.IRobotSTRFileAnalyser.ReadParams 2208	RobotOM.IRobotStructuralAxisGridMngr.Create 1526
RobotOM.IRobotSTRParameter 1268	RobotOM.IRobotStructuralAxisGridMngr.Delete 1526
RobotOM.IRobotSTRParameter.Description 1270	RobotOM.IRobotStructuralAxisGridMngr.FindByName 1526
RobotOM.IRobotSTRParameter.DoubleVal_1 1270	RobotOM.IRobotStructuralAxisGridMngr.Get 1527
RobotOM.IRobotSTRParameter.DoubleVal_2 1270	RobotOM.IRobotStructuralAxisGridMngr.GetByName 1527
RobotOM.IRobotSTRParameter.DoubleVal_3 1271	RobotOM.IRobotStructuralAxisGridMngr.IsActive 1527
RobotOM.IRobotSTRParameter.FilePathVal 1271	RobotOM.IRobotStructuralAxisGridType 1518
RobotOM.IRobotSTRParameter.IntegerVal 1271	RobotOM.IRobotStructuralAxisLabelType 1513
RobotOM.IRobotSTRParameter.IsActive 1271	RobotOM.IRobotStructuralAxisSequenceList 1513
RobotOM.IRobotSTRParameter.Name 1272	RobotOM.IRobotStructuralAxisSequenceList.AddSequence 1515
RobotOM.IRobotSTRParameter.SelectionVal 1272	RobotOM.IRobotStructuralAxisSequenceList.AxisCount 1514
RobotOM.IRobotSTRParameter.TextList 1272	RobotOM.IRobotStructuralAxisSequenceList.Clear 1516
RobotOM.IRobotSTRParameter.TextVal 1272	RobotOM.IRobotStructuralAxisSequenceList.DeleteSequence 1516
RobotOM.IRobotSTRParameter.Type 1273	RobotOM.IRobotStructuralAxisSequenceList.FindAxisByPos 1516
RobotOM.IRobotSTRParams 1267	RobotOM.IRobotStructuralAxisSequenceList.GetAxis 1516
RobotOM.IRobotSTRParams.Count 1267	RobotOM.IRobotStructuralAxisSequenceList.GetSequence 1517
RobotOM.IRobotSTRParams.FindParameter 1268	RobotOM.IRobotStructuralAxisSequenceList.SequenceCount 1514
RobotOM.IRobotSTRParams.GetParameter 1268	RobotOM.IRobotStructuralAxisSequenceList.SetAxisLabel 1517
RobotOM.IRobotSTRParamType 1273	RobotOM.IRobotStructuralAxisSequenceList.SetLabelFormat 1517
RobotOM.IRobotStructuralAxisGrid 1518	RobotOM.IRobotStructuralAxisSequenceList.SingleOutAxis 1518
RobotOM.IRobotStructuralAxisGrid.Name 1519	RobotOM.IRobotStructuralAxisSequenceListStartPosition 1515
RobotOM.IRobotStructuralAxisGrid.Save 1520	RobotOM.IRobotStructure 1085
RobotOM.IRobotStructuralAxisGrid.Type 1519	RobotOM.IRobotStructure.ApplyCache 1091
RobotOM.IRobotStructuralAxisGridCartesian 1520	RobotOM.IRobotStructure.Bars 1086
RobotOM.IRobotStructuralAxisGridCartesian.GetRelativeToPoint 1523	RobotOM.IRobotStructure.Cases 1087
RobotOM.IRobotStructuralAxisGridCartesian.IncludeStoreysInZ 1521	RobotOM.IRobotStructure.Clear 1092
RobotOM.IRobotStructuralAxisGridCartesian.RotationAngle 1521	RobotOM.IRobotStructure.CreateCache 1092
RobotOM.IRobotStructuralAxisGridCartesian.RotationAxis 1522	RobotOM.IRobotStructure.Edit 1087
RobotOM.IRobotStructuralAxisGridCartesian.SetRelativeToPoint 1523	RobotOM.IRobotStructure.ExportXml 1092
RobotOM.IRobotStructuralAxisGridCartesian.StoreysInZ 1522	RobotOM.IRobotStructure.ExtParams 1087
RobotOM.IRobotStructuralAxisGridCartesian.X 1522	RobotOM.IRobotStructure.FiniteElems 1088

RobotOM.IRobotStructure.GroupObjects 1088	RobotOM.IRobotStructureCache.SetBarLabel 1095
RobotOM.IRobotStructure.Groups 1088	RobotOM.IRobotStructureCache.SetBarName 1095
RobotOM.IRobotStructure.IsCalcModelGenerated 1088	RobotOM.IRobotStructureEditTools 1043
RobotOM.IRobotStructure.Labels 1089	RobotOM.IRobotStructureEditTools.DivideBar 1044
RobotOM.IRobotStructure.Merge 1093	RobotOM.IRobotStructureEditTools.SelMirror 1045
RobotOM.IRobotStructure.Nodes 1089	RobotOM.IRobotStructureEditTools.SelRotate 1045
RobotOM.IRobotStructure.Objects 1089	RobotOM.IRobotStructureEditTools.SelScale 1045
RobotOM.IRobotStructure.QuantitySurvey 1090	RobotOM.IRobotStructureEditTools.SelTranslate 1046
RobotOM.IRobotStructure.Results 1090	RobotOM.IRobotStructureEditTools.TranslateBar 1046
RobotOM.IRobotStructure.ResultsFreeze 1090	RobotOM.IRobotStructureEditTools.TranslateNode 1046
RobotOM.IRobotStructure.Selections 1090	RobotOM.IRobotStructureEvents 1099
RobotOM.IRobotStructure.Storeys 1091	RobotOM.IRobotStructureEvents.ResultStatusChanged 1100
RobotOM.IRobotStructure.Type 1091	RobotOM.IRobotStructureGeoAnalyser 2210
RobotOM.IRobotStructureAnalysisModalParticipationCoeff 1571	RobotOM.IRobotStructureGeoAnalyser.CanEliminateIsolatedNodes 2211
RobotOM.IRobotStructureAnalysisParams 1560	RobotOM.IRobotStructureGeoAnalyser.CanExtendBars 2211
RobotOM.IRobotStructureAnalysisParams.AutoBarMerging 1561	RobotOM.IRobotStructureGeoAnalyser.CanUseRigidLinks 2212
RobotOM.IRobotStructureAnalysisParams.AutoVerification 1561	RobotOM.IRobotStructureGeoAnalyser.Correct 2213
RobotOM.IRobotStructureAnalysisParams.DSCAlgorithm 1562	RobotOM.IRobotStructureGeoAnalyser.DefineIntersection 2213
RobotOM.IRobotStructureAnalysisParams.EquationSolvingMethod 1562	RobotOM.IRobotStructureGeoAnalyser.Precision 2212
RobotOM.IRobotStructureAnalysisParams.FictitiousRigidityCoeff 1562	RobotOM.IRobotStructureGeoAnalyser.SetEdgeSize 2213
RobotOM.IRobotStructureAnalysisParams.IgnoreWarnings 1563	RobotOM.IRobotStructureGeoAnalyser.SetNodeSize 2214
RobotOM.IRobotStructureAnalysisParams.IterativeParams 1563	RobotOM.IRobotStructureGeoAnalyser.StartCollectInfo 2214
RobotOM.IRobotStructureAnalysisParams.ModalParticipationCoeff 1563	RobotOM.IRobotStructureGeoAnalyser.StopCollectInfo 2214
RobotOM.IRobotStructureAnalysisParams.RLINKElements 1564	RobotOM.IRobotStructureMergeData 1097
RobotOM.IRobotStructureAnalysisParams.SparseMParams 1564	RobotOM.IRobotStructureMergeData.CreateStructure 1098
RobotOM.IRobotStructureApplyInfo 1096	RobotOM.IRobotStructureMergeData.LoadStructure 1099
RobotOM.IRobotStructureApplyInfo.Bars 1096	RobotOM.IRobotStructureMergeData.Structure 1098
RobotOM.IRobotStructureApplyInfo.Nodes 1097	RobotOM.IRobotStructureQuantitySurvey 1083
RobotOM.IRobotStructureAutoVerificationType 1564	RobotOM.IRobotStructureQuantitySurvey.BarSections 1084
RobotOM.IRobotStructureCache 1093	RobotOM.IRobotStructureQuantitySurvey.Materials 1084
RobotOM.IRobotStructureCache.AddBar 1094	RobotOM.IRobotStructureQuantitySurvey.PanelThickness 1084
RobotOM.IRobotStructureCache.AddNode 1094	RobotOM.IRobotStructureValues 1013
RobotOM.IRobotStructureCache.EnsureNodeExist 1095	RobotOM.IRobotStructureValues.GetEx2 1015
	RobotOM.IRobotStructureValues.GetEy2 1015
	RobotOM.IRobotStructureValues.GetG 1015
	RobotOM.IRobotStructureValues.GetIx 1016
	RobotOM.IRobotStructureValues.GetLy 1016
	RobotOM.IRobotStructureValues.GetLz 1016
	RobotOM.IRobotStructureValues.GetMass 1017

RobotOM.IRobotStructureValues.GetT	1017	136
RobotOM.IRobotSupportColumnFixingType	522	RobotOM.IRobotSWCodeECPParams.PermDoorRightSide 136
RobotOM.IRobotSupportEquivalentColumnElasticity	520	RobotOM.IRobotSWCodeECPParams.PermDoorRightSidePresent 136
RobotOM.IRobotSupportEquivalentColumnElasticity.Fixing1	521	RobotOM.IRobotSWCodeECPParams.PermFront 136
RobotOM.IRobotSupportEquivalentColumnElasticity.Fixing2	521	RobotOM.IRobotSWCodeECPParams.PermLeftSide 137
RobotOM.IRobotSupportEquivalentColumnElasticity.L1	521	RobotOM.IRobotSWCodeECPParams.PermRear 137
RobotOM.IRobotSupportEquivalentColumnElasticity.L2	521	RobotOM.IRobotSWCodeECPParams.PermRightSide 137
RobotOM.IRobotSupportEquivalentColumnElasticity.ThroughTwoStories	522	RobotOM.IRobotSWCodeECPParams.ReferenceLevel 137
RobotOM.IRobotSupportEquivalentElasticity	516	RobotOM.IRobotSWCodeECPParams.SnowBarCoeffGet 145
RobotOM.IRobotSupportEquivalentElasticity.E	516	RobotOM.IRobotSWCodeECPParams.SnowBarCoeffSet 145
RobotOM.IRobotSupportEquivalentElasticity.PoissonRatio	517	RobotOM.IRobotSWCodeECPParams.SnowGutterBars 138
RobotOM.IRobotSupportEquivalentElasticity.SetMaterial	517	RobotOM.IRobotSWCodeECPParams.SnowPressureExtreme 138
RobotOM.IRobotSupportEquivalentElasticity.Type	517	RobotOM.IRobotSWCodeECPParams.SnowPressureNormal 138
RobotOM.IRobotSupportEquivalentElasticityType	518	RobotOM.IRobotSWCodeECPParams.SnowRedistribution 138
RobotOM.IRobotSupportEquivalentWallElasticity	518	RobotOM.IRobotSWCodeECPParams.StructureAge 139
RobotOM.IRobotSupportEquivalentWallElasticity.L1	519	RobotOM.IRobotSWCodeECPParams.StructureAgeCode 139
RobotOM.IRobotSupportEquivalentWallElasticity.L2	519	RobotOM.IRobotSWCodeECPParams.StructureHeight 139
RobotOM.IRobotSupportEquivalentWallElasticity.ThroughTwoStories	519	RobotOM.IRobotSWCodeECPParams.StructureP 139
RobotOM.IRobotSurfaceOnObjectRecordValues	51	RobotOM.IRobotSWCodeECPParams.WindAutoCd 140
RobotOM.IRobotSWCodeECCdType	147	RobotOM.IRobotSWCodeECPParams.WindBarCoeffGet 145
RobotOM.IRobotSWCodeECGroundType	146	RobotOM.IRobotSWCodeECPParams.WindBarCoeffSet 145
RobotOM.IRobotSWCodeECPParams	130	RobotOM.IRobotSWCodeECPParams.WindCALT 140
RobotOM.IRobotSWCodeECPParams.Altitude	133	RobotOM.IRobotSWCodeECPParams.WindCd 140
RobotOM.IRobotSWCodeECPParams.GlobalCDIR	133	RobotOM.IRobotSWCodeECPParams.WindCdType 141
RobotOM.IRobotSWCodeECPParams.GroundType	133	RobotOM.IRobotSWCodeECPParams.WindCt 141
RobotOM.IRobotSWCodeECPParams.LeftWind2NordAngle	133	RobotOM.IRobotSWCodeECPParams.WindCtAuto 141
RobotOM.IRobotSWCodeECPParams.NodalLoadsForAllBars	134	RobotOM.IRobotSWCodeECPParams.WindCTEM 142
RobotOM.IRobotSWCodeECPParams.NodalLoadsForBarsList	134	RobotOM.IRobotSWCodeECPParams.WindE 142
RobotOM.IRobotSWCodeECPParams.PermDoorFront	134	RobotOM.IRobotSWCodeECPParams.WindKT 142
RobotOM.IRobotSWCodeECPParams.PermDoorFrontPresent	134	RobotOM.IRobotSWCodeECPParams.WindPressureAutomatic 142
RobotOM.IRobotSWCodeECPParams.PermDoorLeftSide	135	RobotOM.IRobotSWCodeECPParams.WindQref 143
RobotOM.IRobotSWCodeECPParams.PermDoorLeftSidePresent	135	RobotOM.IRobotSWCodeECPParams.WindQref0 143
RobotOM.IRobotSWCodeECPParams.PermDoorRightSide	135	RobotOM.IRobotSWCodeECPParams.WindQref0p 143
RobotOM.IRobotSWCodeECPParams.PermDoorRear	135	RobotOM.IRobotSWCodeECPParams.WindSiteType 143
RobotOM.IRobotSWCodeECPParams.PermDoorRearPresent		RobotOM.IRobotSWCodeECPParams.WindVref0 144
		RobotOM.IRobotSWCodeECPParams.WindZ0 144
		RobotOM.IRobotSWCodeECPParams.WindZMin 144
		RobotOM.IRobotSWCodeECSiteType 146

RobotOM.IRobotSWCodeFRParams 109	RobotOM.IRobotSWCodeFRParams.SnowPressureNormalAutomatic 120
RobotOM.IRobotSWCodeFRParams.Altitude 113	RobotOM.IRobotSWCodeFRParams.SnowRedistribution 120
RobotOM.IRobotSWCodeFRParams.IsolatedRoofGetLocation 126	RobotOM.IRobotSWCodeFRParams.SnowRegion 120
RobotOM.IRobotSWCodeFRParams.IsolatedRoofs 113	RobotOM.IRobotSWCodeFRParams.SnowType 120
RobotOM.IRobotSWCodeFRParams.IsolatedRoofSetLocation 127	RobotOM.IRobotSWCodeFRParams.SnowWaterOutflow 121
RobotOM.IRobotSWCodeFRParams.NodalLoadsForAllBars 113	RobotOM.IRobotSWCodeFRParams.StructureHeight 121
RobotOM.IRobotSWCodeFRParams.NodalLoadsForBarsList 114	RobotOM.IRobotSWCodeFRParams.SurfaceLower 121
RobotOM.IRobotSWCodeFRParams.OpenStructure 114	RobotOM.IRobotSWCodeFRParams.SurfaceUpper 121
RobotOM.IRobotSWCodeFRParams.PermDoorFront 114	RobotOM.IRobotSWCodeFRParams.WindBarCoeffGet 128
RobotOM.IRobotSWCodeFRParams.PermDoorFrontPresent 114	RobotOM.IRobotSWCodeFRParams.WindBarCoeffSet 128
RobotOM.IRobotSWCodeFRParams.PermDoorLeftSide 115	RobotOM.IRobotSWCodeFRParams.WindCoastalArea 122
RobotOM.IRobotSWCodeFRParams.PermDoorLeftSidePresent 115	RobotOM.IRobotSWCodeFRParams.WindDeltaCoeff 122
RobotOM.IRobotSWCodeFRParams.PermDoorRear 115	RobotOM.IRobotSWCodeFRParams.WindDeltaCoeffAutomatic 122
RobotOM.IRobotSWCodeFRParams.PermDoorRearPresent 115	RobotOM.IRobotSWCodeFRParams.WindDynamicAction 122
RobotOM.IRobotSWCodeFRParams.PermDoorRightSide 116	RobotOM.IRobotSWCodeFRParams.WindDynamicActionCoefficient 123
RobotOM.IRobotSWCodeFRParams.PermDoorRightSidePresent 116	RobotOM.IRobotSWCodeFRParams.WindDynamicActionCoefficientAutomatic 123
RobotOM.IRobotSWCodeFRParams.PermFront 116	RobotOM.IRobotSWCodeFRParams.WindDynamicActionPeriod 123
RobotOM.IRobotSWCodeFRParams.PermLeftSide 116	RobotOM.IRobotSWCodeFRParams.WindDynamicActionSteeleStructure 124
RobotOM.IRobotSWCodeFRParams.PermRear 117	RobotOM.IRobotSWCodeFRParams.WindFacadeOffset 124
RobotOM.IRobotSWCodeFRParams.PermRightSide 117	RobotOM.IRobotSWCodeFRParams.WindMultipleRoof 124
RobotOM.IRobotSWCodeFRParams.ReferenceLevel 117	RobotOM.IRobotSWCodeFRParams.WindPressure 124
RobotOM.IRobotSWCodeFRParams.RiseOfRoof 117	RobotOM.IRobotSWCodeFRParams.WindPressureAutomatic 125
RobotOM.IRobotSWCodeFRParams.RiseOfRoofAutomatic 118	RobotOM.IRobotSWCodeFRParams.WindPressureCeCiMinimum 125
RobotOM.IRobotSWCodeFRParams.SnowBarCoeffGet 127	RobotOM.IRobotSWCodeFRParams.WindPressureVariable 125
RobotOM.IRobotSWCodeFRParams.SnowBarCoeffSet 127	RobotOM.IRobotSWCodeFRParams.WindRegion 125
RobotOM.IRobotSWCodeFRParams.SnowGutterBars 118	RobotOM.IRobotSWCodeFRParams.WindSite 126
RobotOM.IRobotSWCodeFRParams.SnowIsWaterOutflow 118	RobotOM.IRobotSWCodeFRParams.WindType 126
RobotOM.IRobotSWCodeFRParams.SnowObstacles 119	RobotOM.IRobotSWCodeFRSnowType 129
RobotOM.IRobotSWCodeFRParams.SnowPressureExtreme 119	RobotOM.IRobotSWCodeFRSurfaceType 129
RobotOM.IRobotSWCodeFRParams.SnowPressureExtremeAutomatic 119	RobotOM.IRobotSWCodeFRWindSite 128
RobotOM.IRobotSWCodeFRParams.SnowPressureNormal 119	RobotOM.IRobotSWCodeFRWindType 128
	RobotOM.IRobotSWCodePLParams 98

RobotOM.IRobotSWCodePLParams.Altitude 100	RobotOM.IRobotSWStruct3D.FrameElemCount 83
RobotOM.IRobotSWCodePLParams.IsolatedRoofGetLocation 106	RobotOM.IRobotSWStruct3D.GetFrame 83
RobotOM.IRobotSWCodePLParams.IsolatedRoofs 100	RobotOM.IRobotSWStruct3D.SetFrame 83
RobotOM.IRobotSWCodePLParams.IsolatedRoofSetLocation 107	RobotOM.IRobotSWStruct3DElement 84
RobotOM.IRobotSWCodePLParams.NodalLoadsForAllBars 101	RobotOM.IRobotSWStruct3DElement.Bar 84
RobotOM.IRobotSWCodePLParams.NodalLoadsForBarsList 101	RobotOM.IRobotSWStruct3DElement.IsFacadeLoaded 85
RobotOM.IRobotSWCodePLParams.PermFront 101	RobotOM.IRobotSWStruct3DElement.IsFacadeOnly 85
RobotOM.IRobotSWCodePLParams.PermLeftSide 101	RobotOM.IRobotSWStruct3DElement.Purlins 85
RobotOM.IRobotSWCodePLParams.PermRear 102	RobotOM.IRobotSWStruct3DFrame 85
RobotOM.IRobotSWCodePLParams.PermRightSide 102	RobotOM.IRobotSWStruct3DFrame.ElemCount 86
RobotOM.IRobotSWCodePLParams.ReferenceLevel 102	RobotOM.IRobotSWStruct3DFrame.GetElem 87
RobotOM.IRobotSWCodePLParams.SnowBarCoeffGet 107	RobotOM.IRobotSWStruct3DFrame.SetElem 87
RobotOM.IRobotSWCodePLParams.SnowBarCoeffSet 107	RobotOM.IRobotSWStruct3DGenParams 87
RobotOM.IRobotSWCodePLParams.SnowPressure 102	RobotOM.IRobotSWStruct3DGenParams.Bars 88
RobotOM.IRobotSWCodePLParams.SnowPressureAutomatic 103	RobotOM.IRobotSWStruct3DGenParams.FacadeLoadedBars 88
RobotOM.IRobotSWCodePLParams.SnowRedistribution 103	RobotOM.IRobotSWStruct3DGenParams.FacadeOnlyBars 89
RobotOM.IRobotSWCodePLParams.SnowZone 103	RobotOM.IRobotSWStruct3DGenParams.FrameCount 89
RobotOM.IRobotSWCodePLParams.StructureHeight 103	RobotOM.IRobotSWStruct3DGenParams.GetPurlins 90
RobotOM.IRobotSWCodePLParams.WindBarCoeffGet 107	RobotOM.IRobotSWStruct3DGenParams.Offsets 89
RobotOM.IRobotSWCodePLParams.WindBarCoeffSet 108	RobotOM.IRobotSWStruct3DGenParams.SetPurlins 90
RobotOM.IRobotSWCodePLParams.WindDynamicAction 104	RobotOM.IRobotSWStruct3DGenParams.Spacings 89
RobotOM.IRobotSWCodePLParams.WindDynamicDecrement 104	RobotOM.IRobotSWStruct3DPurlinGenParams 90
RobotOM.IRobotSWCodePLParams.WindDynamicPeriod 104	RobotOM.IRobotSWStruct3DPurlinGenParams.Locations 91
RobotOM.IRobotSWCodePLParams.WindMultipleRoofs 104	RobotOM.IRobotSWStruct3DPurlinGenParams.RelativeLocati ons 91
RobotOM.IRobotSWCodePLParams.WindPressure 105	RobotOM.IRobotSWStruct3DPurlinGenParams.SectionName 92
RobotOM.IRobotSWCodePLParams.WindPressureAutomatic 105	RobotOM.IRobotTable 1357
RobotOM.IRobotSWCodePLParams.WindPressureDistribOnH eight 105	RobotOM.IRobotTable.AddColumn 1362
RobotOM.IRobotSWCodePLParams.WindSite 106	RobotOM.IRobotTable.ColCount 1359
RobotOM.IRobotSWCodePLParams.WindZone 106	RobotOM.IRobotTable.Configuration 1359
RobotOM.IRobotSWCodePLSnowZone 109	RobotOM.IRobotTable.GetColWidth 1362
RobotOM.IRobotSWCodePLWindPressDistribType 129	RobotOM.IRobotTable.GetDataType 1362
RobotOM.IRobotSWCodePLWindSite 108	RobotOM.IRobotTable.GetRowHeight 1363
RobotOM.IRobotSWCodePLWindZone 108	RobotOM.IRobotTable.MakeScreenCapture 1363
RobotOM.IRobotSWStruct3D 82	RobotOM.IRobotTable.Printable 1360
RobotOM.IRobotSWStruct3D.FrameCount 82	RobotOM.IRobotTable.RowCount 1360
	RobotOM.IRobotTable.Select 1363
	RobotOM.IRobotTable.SelectAllFromStore 1364
	RobotOM.IRobotTable.SelectFromStore 1364
	RobotOM.IRobotTable.SelectMode 1364
	RobotOM.IRobotTable.SelectModeFromStore 1365

RobotOM.IRobotTable.SetColWidth 1365	RobotOM.IRobotThicknessHomoData.GetP1 721
RobotOM.IRobotTable.SetRowHeight 1365	RobotOM.IRobotThicknessHomoData.GetP2 721
RobotOM.IRobotTable.StoreModeSelection 1365	RobotOM.IRobotThicknessHomoData.GetP3 721
RobotOM.IRobotTable.StoreSelection 1366	RobotOM.IRobotThicknessHomoData.GetReduction 722
RobotOM.IRobotTable.Title 1360	RobotOM.IRobotThicknessHomoData.SetP1 722
RobotOM.IRobotTable.UserControl 1360	RobotOM.IRobotThicknessHomoData.SetP2 722
RobotOM.IRobotTable.Visible 1361	RobotOM.IRobotThicknessHomoData.SetP3 722
RobotOM.IRobotTable.Window 1361	RobotOM.IRobotThicknessHomoData.SetReduction 723
RobotOM.IRobotTableConfig 1356	RobotOM.IRobotThicknessHomoData.Thick1 719
RobotOM.IRobotTableConfig.SetFlag 1357	RobotOM.IRobotThicknessHomoData.Thick2 719
RobotOM.IRobotTableConfig.SetValue 1357	RobotOM.IRobotThicknessHomoData.Thick3 720
RobotOM.IRobotTableConfigFlag 1355	RobotOM.IRobotThicknessHomoData.ThickConst 720
RobotOM.IRobotTableConfigValue 1356	RobotOM.IRobotThicknessHomoData.Type 720
RobotOM.IRobotTableDataType 1354	RobotOM.IRobotThicknessHomoType 723
RobotOM.IRobotTableFrame 1366	RobotOM.IRobotThicknessMatrix 733
RobotOM.IRobotTableFrame.Count 1367	RobotOM.IRobotThicknessMatrix.GetValue 734
RobotOM.IRobotTableFrame.Current 1367	RobotOM.IRobotThicknessMatrix.SetValue 734
RobotOM.IRobotTableFrame.Get 1368	RobotOM.IRobotThicknessMatrixValue 735
RobotOM.IRobotTableFrame.GetName 1368	RobotOM.IRobotThicknessOrthoData 724
RobotOM.IRobotTableFrameSetName 1369	RobotOM.IRobotThicknessOrthoData.A 726
RobotOM.IRobotTableFrame.Window 1368	RobotOM.IRobotThicknessOrthoData.A1 726
RobotOM.IRobotTableScreenCaptureParams 1369	RobotOM.IRobotThicknessOrthoData.A2 727
RobotOM.IRobotTableScreenCaptureParams.AddInfoTab 1370	RobotOM.IRobotThicknessOrthoData.B 727
RobotOM.IRobotTableScreenCaptureParams.AdjustColumnWidth 1370	RobotOM.IRobotThicknessOrthoData.B1 727
RobotOM.IRobotTableScreenCaptureParams.Comment 1371	RobotOM.IRobotThicknessOrthoData.DirType 727
RobotOM.IRobotTableScreenCaptureParams.IncludeDateAndTime 1371	RobotOM.IRobotThicknessOrthoData.DisregardBendStiffDirY 728
RobotOM.IRobotTableScreenCaptureParams.Name 1371	RobotOM.IRobotThicknessOrthoData.ES 728
RobotOM.IRobotTableScreenCaptureParams.UpdateType 1371	RobotOM.IRobotThicknessOrthoData.GetVector 733
RobotOM.IRobotTableScreenCaptureUpdateType 1372	RobotOM.IRobotThicknessOrthoData.H 728
RobotOM.IRobotTableType 1352	RobotOM.IRobotThicknessOrthoData.H0 729
RobotOM.IRobotThermalIn3PointsRecordValues 47	RobotOM.IRobotThicknessOrthoData.H1 729
RobotOM.IRobotThermalRecordValues 49	RobotOM.IRobotThicknessOrthoData.H2 729
RobotOM.IRobotThicknessData 715	RobotOM.IRobotThicknessOrthoData.HA 729
RobotOM.IRobotThicknessData.Data 716	RobotOM.IRobotThicknessOrthoData.HB 730
RobotOM.IRobotThicknessData.ElasticFoundation 716	RobotOM.IRobotThicknessOrthoData.HC 730
RobotOM.IRobotThicknessData.MaterialName 717	RobotOM.IRobotThicknessOrthoData.Matrix 730
RobotOM.IRobotThicknessData.ThicknessType 717	RobotOM.IRobotThicknessOrthoData.N1 731
RobotOM.IRobotThicknessData.Uplift 717	RobotOM.IRobotThicknessOrthoData.N2 731
RobotOM.IRobotThicknessHomoData 718	RobotOM.IRobotThicknessOrthoData.SetVector 733
	RobotOM.IRobotThicknessOrthoData.T 731
	RobotOM.IRobotThicknessOrthoData.Thick1 731
	RobotOM.IRobotThicknessOrthoData.Thick2 732

RobotOM.IRobotThicknessOrthoData.Thick3 732	RobotOM.IRobotTimeHistoryHHTParams.CoeffAlpha 384
RobotOM.IRobotThicknessOrthoData.Type 732	RobotOM.IRobotTimeHistoryHHTParams.Nonlinearity 384
RobotOM.IRobotThicknessOrthoData.VS 732	RobotOM.IRobotTimeHistoryHHTParams.NonlinearParams 384
RobotOM.IRobotThicknessOrthoDirType 724	RobotOM.IRobotTimeHistoryModalDecompositionParams 368
RobotOM.IRobotThicknessOrthoType 723	RobotOM.IRobotTimeHistoryModalDecompositionParams.Count 369
RobotOM.IRobotThicknessQuantitySurvey 1080	RobotOM.IRobotTimeHistoryModalDecompositionParams.Delete 369
RobotOM.IRobotThicknessQuantitySurvey.Count 1081	RobotOM.IRobotTimeHistoryModalDecompositionParams.DeleteMode 370
RobotOM.IRobotThicknessQuantitySurvey.GetArea 1081	RobotOM.IRobotTimeHistoryModalDecompositionParams.Get 370
RobotOM.IRobotThicknessQuantitySurvey.GetName 1082	RobotOM.IRobotTimeHistoryModalDecompositionParams.GetDamping 370
RobotOM.IRobotThicknessQuantitySurvey.GetUnitWeight 1082	RobotOM.IRobotTimeHistoryModalDecompositionParams.IsDefined 371
RobotOM.IRobotThicknessQuantitySurvey.GetVolume 1082	RobotOM.IRobotTimeHistoryModalDecompositionParams.SetDamping 371
RobotOM.IRobotThicknessQuantitySurvey.GetWeight 1083	RobotOM.IRobotTimeHistoryNewmarkAccelParams 385
RobotOM.IRobotThicknessType 717	RobotOM.IRobotTimeHistoryNewmarkAccelParams.Alpha 385
RobotOM.IRobotThicknessUpliftType 718	RobotOM.IRobotTimeHistoryNewmarkAccelParams.Beta 386
RobotOM.IRobotTimeHistoryAnalysisMethod 366	RobotOM.IRobotTimeHistoryNewmarkAccelParams.Nonlinearity 386
RobotOM.IRobotTimeHistoryAnalysisParams 361	RobotOM.IRobotTimeHistoryNewmarkAccelParams.NonlinearParams 386
RobotOM.IRobotTimeHistoryAnalysisParams.Count 362	RobotOM.IRobotTimeHistoryNewmarkParams 366
RobotOM.IRobotTimeHistoryAnalysisParams.Delete 365	RobotOM.IRobotTimeHistoryNewmarkParams.Alpha 367
RobotOM.IRobotTimeHistoryAnalysisParams.Division 362	RobotOM.IRobotTimeHistoryNewmarkParams.Beta 367
RobotOM.IRobotTimeHistoryAnalysisParams.End 363	RobotOM.IRobotTimeHistoryNewmarkParams.MassMatrixType 368
RobotOM.IRobotTimeHistoryAnalysisParams.Find 365	RobotOM.IRobotTimeHistoryNonlinearParams 380
RobotOM.IRobotTimeHistoryAnalysisParams.Get 365	RobotOM.IRobotTimeHistoryNonlinearParams.MatrixUpdateAfterEachIteration 381
RobotOM.IRobotTimeHistoryAnalysisParams.InitialCase 363	RobotOM.IRobotTimeHistoryNonlinearParams.MatrixUpdateAfterEachSubdivision 381
RobotOM.IRobotTimeHistoryAnalysisParams.Method 363	RobotOM.IRobotTimeHistoryNonlinearParams.MaximumIterationsForOneIncrement 381
RobotOM.IRobotTimeHistoryAnalysisParams.MethodParams 364	
RobotOM.IRobotTimeHistoryAnalysisParams.Set 366	
RobotOM.IRobotTimeHistoryAnalysisParams.TimeStep 364	
RobotOM.IRobotTimeHistoryFunctionList 375	
RobotOM.IRobotTimeHistoryFunctionList.AddFromFile 377	
RobotOM.IRobotTimeHistoryFunctionList.Count 376	
RobotOM.IRobotTimeHistoryFunctionList.Create 377	
RobotOM.IRobotTimeHistoryFunctionList.CreateSum 378	
RobotOM.IRobotTimeHistoryFunctionList.Delete 378	
RobotOM.IRobotTimeHistoryFunctionList.Find 378	
RobotOM.IRobotTimeHistoryFunctionList.Get 379	
RobotOM.IRobotTimeHistoryFunctionList.GetName 379	
RobotOM.IRobotTimeHistoryFunctionList.SaveToFile 379	
RobotOM.IRobotTimeHistoryFunctionList.Store 380	
RobotOM.IRobotTimeHistoryHHTParams 382	
RobotOM.IRobotTimeHistoryHHTParams.Alpha 383	
RobotOM.IRobotTimeHistoryHHTParams.Beta 383	

RobotOM.IRobotTimeHistoryNonlinearParams.PDelta	382	RobotOM.IRobotUnitEditionServer	1283
RobotOM.IRobotTimeHistoryNonlinearParams.ResidualForce sRelativeCodeTolerance	382	RobotOM.IRobotUnitEditionServer.Count	1284
RobotOM.IRobotTimeHistoryPointsCollection	371	RobotOM.IRobotUnitEditionServer.Delete	1284
RobotOM.IRobotTimeHistoryPointsCollection.Add	373	RobotOM.IRobotUnitEditionServer.Find	1284
RobotOM.IRobotTimeHistoryPointsCollection.Clear	373	RobotOM.IRobotUnitEditionServer.Get	1285
RobotOM.IRobotTimeHistoryPointsCollection.Count	372	RobotOM.IRobotUnitEditionServer.New	1285
RobotOM.IRobotTimeHistoryPointsCollection.Delete	374	RobotOM.IRobotUnitEditionServer.Set	1285
RobotOM.IRobotTimeHistoryPointsCollection.Find	374	RobotOM.IRobotUnitEditionType	1277
RobotOM.IRobotTimeHistoryPointsCollection.Get	374	RobotOM.IRobotUnitMngr	1278
RobotOM.IRobotTimeHistoryPointsCollection.LoadFromFile	375	RobotOM.IRobotUnitMngr.Count	1280
RobotOM.IRobotTimeHistoryPointsCollection.SaveToFile	375	RobotOM.IRobotUnitMngr.Get	1280
RobotOM.IRobotTimeHistoryResults	900	RobotOM.IRobotUnitMngr.GetCoeff	1280
RobotOM.IRobotTimeHistoryResults.ARX	901	RobotOM.IRobotUnitMngr.GetCoeff2	1281
RobotOM.IRobotTimeHistoryResults.ARY	901	RobotOM.IRobotUnitMngr.GetName	1281
RobotOM.IRobotTimeHistoryResults.ARZ	901	RobotOM.IRobotUnitMngr.Refresh	1281
RobotOM.IRobotTimeHistoryResults.AX	902	RobotOM.IRobotUnitMngr.Set	1282
RobotOM.IRobotTimeHistoryResults.AY	902	RobotOM.IRobotUnitMngr.UnitEdition	1279
RobotOM.IRobotTimeHistoryResults.AZ	902	RobotOM.IRobotUnitMngr.UseMetricAsDefault	1279
RobotOM.IRobotTimeHistoryResults.Time	903	RobotOM.IRobotUnitEvents	1287
RobotOM.IRobotTimeHistoryResults.VRX	903	RobotOM.IRobotUnitEvents.UnitsChanged	1288
RobotOM.IRobotTimeHistoryResults.VRY	903	RobotOM.IRobotUnitType	1275
RobotOM.IRobotTimeHistoryResults.VRZ	903	RobotOM.IRobotUniversalResultAccess	1002
RobotOM.IRobotTimeHistoryResults.VX	904	RobotOM.IRobotUniversalResultAccess.Available	1004
RobotOM.IRobotTimeHistoryResults.VY	904	RobotOM.IRobotUniversalResultAccess.Bar	1004
RobotOM.IRobotTimeHistoryResults.VZ	904	RobotOM.IRobotUniversalResultAccess.CalcPoint	1004
RobotOM.IRobotTimeHistoryResultServer	928	RobotOM.IRobotUniversalResultAccess.DivCount	1005
RobotOM.IRobotTimeHistoryResultServer.Value	929	RobotOM.IRobotUniversalResultAccess.DivPoint	1005
RobotOM.IRobotTranslateOptions	1043	RobotOM.IRobotUniversalResultAccess.Element	1005
RobotOM.IRobotUniformRecordValues	48	RobotOM.IRobotUniversalResultAccess.GetDirX	1010
RobotOM.IRobotUnitComplexData	1282	RobotOM.IRobotUniversalResultAccess.Layer	1005
RobotOM.IRobotUnitComplexData.Name2	1283	RobotOM.IRobotUniversalResultAccess.LayerArbitraryValue	1006
RobotOM.IRobotUnitData	1275	RobotOM.IRobotUniversalResultAccess.LinearSupports	1006
RobotOM.IRobotUnitData.E	1276	RobotOM.IRobotUniversalResultAccess.LoadCase	1006
RobotOM.IRobotUnitData.Name	1276	RobotOM.IRobotUniversalResultAccess.LoadCaseCmpnt	1007
RobotOM.IRobotUnitData.Precision	1277	RobotOM.IRobotUniversalResultAccess.Mode	1007
RobotOM.IRobotUnitData.Type	1277	RobotOM.IRobotUniversalResultAccess.ModeCmb	1007
RobotOM.IRobotUnitEditionData	1286	RobotOM.IRobotUniversalResultAccess.Node	1007
RobotOM.IRobotUnitEditionData.Coefficient	1286	RobotOM.IRobotUniversalResultAccess.Panel	1008
RobotOM.IRobotUnitEditionData.Type	1287	RobotOM.IRobotUniversalResultAccess.ReducedCutPos	1008
RobotOM.IRobotUnitEditionData.Unit	1287	RobotOM.IRobotUniversalResultAccess.ReinforceCalcMethod	

1008	RobotOM.IRobotVehicleDatabase.LongName 354
RobotOM.IRobotUniversalResultAccess.RelativePoint 1009	RobotOM.IRobotVehicleDatabase.Name 354
RobotOM.IRobotUniversalResultAccess.Reset 1011	RobotOM.IRobotVehicleDatabaseList 349
RobotOM.IRobotUniversalResultAccess.ResultId 1009	RobotOM.IRobotVehicleDatabaseList.Add 351
RobotOM.IRobotUniversalResultAccess.ResultType 1009	RobotOM.IRobotVehicleDatabaseList.AddFromFile 351
RobotOM.IRobotUniversalResultAccess.ResultValue 1009	RobotOM.IRobotVehicleDatabaseList.Count 350
RobotOM.IRobotUniversalResultAccess.ResultValue3D 1011	RobotOM.IRobotVehicleDatabaseList.Create 351
RobotOM.IRobotUniversalResultAccess.SetDirX 1011	RobotOM.IRobotVehicleDatabaseList.Default 350
RobotOM.IRobotUniversalResultAccess.Storey 1010	RobotOM.IRobotVehicleDatabaseList.Find 352
RobotOM.IRobotUniversalResultType 1013	RobotOM.IRobotVehicleDatabaseList.Get 352
RobotOM.IRobotUpliftDirection 540	RobotOM.IRobotVehicleDatabaseList.GetDatabase 352
RobotOM.IRobotUpliftSense 541	RobotOM.IRobotVehicleDatabaseList.Remove 353
RobotOM.IRobotValuesArray 1627	RobotOM.IRobotVehicleLoad 345
RobotOM.IRobotValuesArray.Count 1628	RobotOM.IRobotVehicleLoad.DX 346
RobotOM.IRobotValuesArray.Get 1629	RobotOM.IRobotVehicleLoad.DY 347
RobotOM.IRobotValuesArray.Set 1629	RobotOM.IRobotVehicleLoad.F 347
RobotOM.IRobotValuesArray.GetSize 1629	RobotOM.IRobotVehicleLoad.S 347
RobotOM.IRobotVariableMngr 1480	RobotOM.IRobotVehicleLoad.Type 348
RobotOM.IRobotVariableMngr.AddExtension 1481	RobotOM.IRobotVehicleLoad.X 348
RobotOM.IRobotVariableMngr.Delete 1481	RobotOM.IRobotVehicleLoadMngr 343
RobotOM.IRobotVariableMngr.Exist 1482	RobotOM.IRobotVehicleLoadMngr.Count 344
RobotOM.IRobotVariableMngr.GetPredefinedValue 1482	RobotOM.IRobotVehicleLoadMngr.Delete 344
RobotOM.IRobotVariableMngr.GetValue 1482	RobotOM.IRobotVehicleLoadMngr.Get 344
RobotOM.IRobotVariableMngr.RemoveExtension 1483	RobotOM.IRobotVehicleLoadMngr.New 345
RobotOM.IRobotVariableMngr.SetPredefinedValue 1483	RobotOM.IRobotVehicleLoadType 348
RobotOM.IRobotVariableMngr.SetValue 1483	RobotOM.IRobotView 1389
RobotOM.IRobotVariableMngrExtension 1511	RobotOM.IRobotView.CopyToClipboard 1394
RobotOM.IRobotVariableMngrExtension.GetIndexedValue 1512	RobotOM.IRobotView.DiagramMagnification 1391
RobotOM.IRobotVariableMngrExtension.GetValue 1512	RobotOM.IRobotView.GetRotationPoint 1394
RobotOM.IRobotVariablePredefinedId 1483	RobotOM.IRobotView.GetScale 1394
RobotOM.IRobotVehicleData 340	RobotOM.IRobotView.GetSize 1395
RobotOM.IRobotVehicleData.b 341	RobotOM.IRobotView.GetWorkPoint 1395
RobotOM.IRobotVehicleData.d1 341	RobotOM.IRobotView.GetZoom 1395
RobotOM.IRobotVehicleData.d2 341	RobotOM.IRobotView.IsLocal 1396
RobotOM.IRobotVehicleData.LoadFromDBase 342	RobotOM.IRobotView.ParamsBarMap 1391
RobotOM.IRobotVehicleData.Loads 342	RobotOM.IRobotView.ParamsDiagram 1391
RobotOM.IRobotVehicleData.StoreToDBase 343	RobotOM.IRobotView.ParamsDisplay 1391
RobotOM.IRobotVehicleDatabase 353	RobotOM.IRobotView.ParamsFeMap 1392
RobotOM.IRobotVehicleDatabase.Description 354	RobotOM.IRobotView.Printable 1392
RobotOM.IRobotVehicleDatabase.GetAll 355	RobotOM.IRobotView.Projection 1392
RobotOM.IRobotVehicleDatabase.Load 355	RobotOM.IRobotView.Redraw 1396
RobotOM.IRobotVehicleDatabase.LoadFromFile 356	RobotOM.IRobotView.Rotate 1396

RobotOM.IRobotView.Selection 1392	RobotOM.IRobotViewDetailedAnalysisParams.Set 1429
RobotOM.IRobotView.SetGlobal 1396	RobotOM.IRobotViewDetailedAnalysisParams.SetColor 1430
RobotOM.IRobotView.SetLocal 1397	RobotOM.IRobotViewDetailedAnalysisResultType 1426
RobotOM.IRobotView.SetRotationPoint 1397	RobotOM.IRobotViewDetailedAnalysisTableTab 1409
RobotOM.IRobotView.setScale 1397	RobotOM.IRobotViewDiagramDescriptionType 1421
RobotOM.IRobotView.GetSize 1397	RobotOM.IRobotViewDiagramFillingType 1421
RobotOM.IRobotView.SetWorkPoint 1398	RobotOM.IRobotViewDiagramParams 1415
RobotOM.IRobotView.SetZoom 1398	RobotOM.IRobotViewDiagramParams.Descriptions 1416
RobotOM.IRobotView.Title 1393	RobotOM.IRobotViewDiagramParams.Filling 1416
RobotOM.IRobotView.Type 1393	RobotOM.IRobotViewDiagramParams.GetColor 1418
RobotOM.IRobotView.Visible 1393	RobotOM.IRobotViewDiagramParams.GetScale 1418
RobotOM.IRobotView2 1438	RobotOM.IRobotViewDiagramParams.On 1418
RobotOM.IRobotView2.ParamsPanelCut 1440	RobotOM.IRobotViewDiagramParams.PositiveNegative 1417
RobotOM.IRobotView2.UserControl 1440	RobotOM.IRobotViewDiagramParams.ReactionsInLocalSystem 1417
RobotOM.IRobotView2.Window 1440	RobotOM.IRobotViewDiagramParams.Set 1419
RobotOM.IRobotView3 1453	RobotOM.IRobotViewDiagramParamsSetColor 1419
RobotOM.IRobotView3.MakeScreenCapture 1454	RobotOM.IRobotViewDiagramParams.SetScale 1419
RobotOM.IRobotViewBarMapParams 1424	RobotOM.IRobotViewDiagramParams.Values 1417
RobotOM.IRobotViewBarMapParams.CurrentResult 1425	RobotOM.IRobotViewDiagramPositionType 1441
RobotOM.IRobotViewBarMapParams.Descriptions 1425	RobotOM.IRobotViewDiagramResultType 1420
RobotOM.IRobotViewBarMapParams.MapThicknessCoeff 1425	RobotOM.IRobotViewDiagrams 1422
RobotOM.IRobotViewBarMapParams.StructureDeformation 1425	RobotOM.IRobotViewDiagramSignDifferType 1422
RobotOM.IRobotViewBarMapResultType 1423	RobotOM.IRobotViewDiagramValueType 1454
RobotOM.IRobotViewBarMaps 1422	RobotOM.IRobotViewDisplayAttributes 1411
RobotOM.IRobotViewDetailedAnalysis 1406	RobotOM.IRobotViewDisplayParams 1409
RobotOM.IRobotViewDetailedAnalysis.CurrentTableTab 1407	RobotOM.IRobotViewDisplayParams.HiddenLines 1410
RobotOM.IRobotViewDetailedAnalysis.MakeScreenCapture 1408	RobotOM.IRobotViewDisplayParams.On 1410
RobotOM.IRobotViewDetailedAnalysis.ParamsDetailed 1407	RobotOM.IRobotViewDisplayParams.Set 1411
RobotOM.IRobotViewDetailedAnalysis.UserControl 1408	RobotOM.IRobotViewDisplayParams.SymbolSize 1410
RobotOM.IRobotViewDetailedAnalysis.Window 1408	RobotOM.IRobotViewFeMapCrossPresentationType 1437
RobotOM.IRobotViewDetailedAnalysisParams 1427	RobotOM.IRobotViewFeMapLayerType 1438
RobotOM.IRobotViewDetailedAnalysisParams.Descriptions 1428	RobotOM.IRobotViewFeMapLocalSystemType 1437
RobotOM.IRobotViewDetailedAnalysisParams.Filling 1428	RobotOM.IRobotViewFeMapParams 1431
RobotOM.IRobotViewDetailedAnalysisParams.GetColor 1429	RobotOM.IRobotViewFeMapParams.CrossPresentation 1433
RobotOM.IRobotViewDetailedAnalysisParams.On 1429	RobotOM.IRobotViewFeMapParams.CurrentResult 1433
RobotOM.IRobotViewDetailedAnalysisParams.PositiveNegative 1428	RobotOM.IRobotViewFeMapParams.DeformationActive 1433
RobotOM.IRobotViewDetailedAnalysisParams.ReinforceShowTheoreticAndRealVals 1428	RobotOM.IRobotViewFeMapParams.DeformationConstScale 1434
	RobotOM.IRobotViewFeMapParams.Direction 1434
	RobotOM.IRobotViewFeMapParams.DirectionData 1434
	RobotOM.IRobotViewFeMapParams.DirectionNode 1434

RobotOM.IRobotViewFeMapParams.Isolines 1435	RobotOM.IRobotViewInfluenceLinesParams.Layer 1383
RobotOM.IRobotViewFeMapParams.Layer 1435	RobotOM.IRobotViewInfluenceLinesParams.LayerArbitraryVal 1384
RobotOM.IRobotViewFeMapParams.LayerArbitraryVal 1435	RobotOM.IRobotViewInfluenceLinesParams.Position 1384
RobotOM.IRobotViewFeMapParams.MapScale 1436	RobotOM.IRobotViewInfluenceLinesParams.RangeFrom 1384
RobotOM.IRobotViewFeMapParams.SetDirectionData 1436	RobotOM.IRobotViewInfluenceLinesParams.RangeTo 1385
RobotOM.IRobotViewFeMapParams.Smoothing 1436	RobotOM.IRobotViewInfluenceLinesParams.Set 1385
RobotOM.IRobotViewFeMapParams.WithDescription 1436	RobotOM.IRobotViewInfluenceLinesResultType 1386
RobotOM.IRobotViewFeMapResultType 1430	RobotOM.IRobotViewMngr 1462
RobotOM.IRobotViewFeMaps 1423	RobotOM.IRobotViewMngr.CreateFromSc 1466
RobotOM.IRobotViewFeMapSmoothingType 1438	RobotOM.IRobotViewMngr.CreateRtfView 1466
RobotOM.IRobotViewGlobalAnalysis 1447	RobotOM.IRobotViewMngr.CreateTable 1466
RobotOM.IRobotViewGlobalAnalysis.MakeScreenCapture 1449	RobotOM.IRobotViewMngr.CreateView 1466
RobotOM.IRobotViewGlobalAnalysis.ParamsGlobalAnalysis 1448	RobotOM.IRobotViewMngr.CurrentLayout 1464
RobotOM.IRobotViewGlobalAnalysisParams 1442	RobotOM.IRobotViewMngr.GetTable 1467
RobotOM.IRobotViewGlobalAnalysisParams.IsOn 1443	RobotOM.IRobotViewMngr.GetView 1467
RobotOM.IRobotViewGlobalAnalysisParams.Results 1443	RobotOM.IRobotViewMngr.NewViewAsTab 1464
RobotOM.IRobotViewGlobalAnalysisParams.Set 1444	RobotOM.IRobotViewMngr.RecycleViews 1464
RobotOM.IRobotViewGlobalAnalysisParamsType 1446	RobotOM.IRobotViewMngr.Refresh 1467
RobotOM.IRobotViewGlobalAnalysisResultsParams 1444	RobotOM.IRobotViewMngr.ShowDialog 1468
RobotOM.IRobotViewGlobalAnalysisResultsParams.NPointsValue 1445	RobotOM.IRobotViewMngr.TableCount 1464
RobotOM.IRobotViewGlobalAnalysisResultsParams.RelativeValue 1445	RobotOM.IRobotViewMngr.ViewCount 1465
RobotOM.IRobotViewGlobalAnalysisResultsParams.Type 1445	RobotOM.IRobotViewPanelCutParams 1374
RobotOM.IRobotViewGlobalAnalysisResultsType 1446	RobotOM.IRobotViewPanelCutParams.CurrentReinforcementResult 1375
RobotOM.IRobotViewHiddenLinesDisplayType 1415	RobotOM.IRobotViewPanelCutParams.CurrentResult 1376
RobotOM.IRobotViewInfluenceLines 1379	RobotOM.IRobotViewPanelCutParams.Descriptions 1376
RobotOM.IRobotViewInfluenceLines.MakeScreenCapture 1381	RobotOM.IRobotViewPanelCutParams.Filling 1376
RobotOM.IRobotViewInfluenceLines.ParamsInfluenceLines 1380	RobotOM.IRobotViewPanelCutParams.IntegralValue 1377
RobotOM.IRobotViewInfluenceLinesLayerType 1388	RobotOM.IRobotViewPanelCutParams.Layer 1377
RobotOM.IRobotViewInfluenceLinesLocalSystemType 1387	RobotOM.IRobotViewPanelCutParams.LayerArbitraryValue 1377
RobotOM.IRobotViewInfluenceLinesParams 1381	RobotOM.IRobotViewPanelCutParams.Position 1378
RobotOM.IRobotViewInfluenceLinesParams.Direction 1382	RobotOM.IRobotViewPanelCutParams.Smoothing 1378
RobotOM.IRobotViewInfluenceLinesParams.DirectionData 1382	RobotOM.IRobotViewPanelCuts 1374
RobotOM.IRobotViewInfluenceLinesParams.DirectionNode 1383	RobotOM.IRobotViewProjection 1398
RobotOM.IRobotViewInfluenceLinesParams.Element 1383	RobotOM.IRobotViewReinforcementResultType 1441
RobotOM.IRobotViewInfluenceLinesParams.IsOn 1385	RobotOM.IRobotViewScaleType 1405
	RobotOM.IRobotViewScreenCaptureParams 1449
	RobotOM.IRobotViewScreenCaptureParams.Comment 1450
	RobotOM.IRobotViewScreenCaptureParams.IncludeDateAndTime 1450

RobotOM.IRobotViewScreenCaptureParams.Name 1451	RobotOM.IRobotWindow 1641
RobotOM.IRobotViewScreenCaptureParams.Orientation 1451	RobotOM.IRobotWindow.Activate 1643
RobotOM.IRobotViewScreenCaptureParams.Resolution 1451	RobotOM.IRobotWindow.Caption 1642
RobotOM.IRobotViewScreenCaptureParams.ScaleAutomatic 1451	RobotOM.IRobotWindow.Handle 1642
RobotOM.IRobotViewScreenCaptureParams.ScaleValue 1452	RobotOM.IRobotWindow.IsActive 1643
RobotOM.IRobotViewScreenCaptureParams.UpdateType 1452	RobotOM.IRobotWindow.SendMessage 1644
RobotOM.IRobotViewScreenCaptureResolution 1455	RobotOM.IRobotWindowState 1643
RobotOM.IRobotViewScreenCaptureUpdateType 1452	RobotOM.IRobotWindowState 1644
RobotOM.IRobotViewType 1388	<b>S</b>
RobotOM.IRobotViewVisibilityStatusType 1399	Screen captures 1458
RobotOM.IRobotViewVisibilityStatusValue 1400	Sections 541
RobotOM.IRobotWindLoadsSimulationEngine 468	Seismic analysis parameters 207
RobotOM.IRobotWindLoadsSimulationEngine.Generate 469	Selections 1031
RobotOM.IRobotWindLoadsSimulationEngine.Params 468	Snow/wind loads 81
RobotOM.IRobotWindLoadsSimulationParams 469	Spectral analysis parameters 302
RobotOM.IRobotWindLoadsSimulationParams.DeviationPercent 470	Spread footing 1101
RobotOM.IRobotWindLoadsSimulationParams.DirectionXNEEnabled 471	Steel and timber design 1798
RobotOM.IRobotWindLoadsSimulationParams.DirectionXNYEnabled 471	Stories 1047
RobotOM.IRobotWindLoadsSimulationParams.DirectionXNYPEnabled 471	STR files analyzer 2206
RobotOM.IRobotWindLoadsSimulationParams.DirectionXPENabled 471	Structural axes 1512
RobotOM.IRobotWindLoadsSimulationParams.DirectionXPYNEnabled 472	Structure components 36
RobotOM.IRobotWindLoadsSimulationParams.DirectionXPYPEnabled 472	Structure correction 2209
RobotOM.IRobotWindLoadsSimulationParams.DirectionYNEEnabled 472	Supplementing RC modules with new codes 1791
RobotOM.IRobotWindLoadsSimulationParams.DirectionYPEEnabled 472	Supplementing the required reinforcement module with new codes 1656
RobotOM.IRobotWindLoadsSimulationParams.Elements 473	Support of external file formats 1266
RobotOM.IRobotWindLoadsSimulationParams.OpeningsClosed 473	<b>T</b>
RobotOM.IRobotWindLoadsSimulationParams.TerrainLevel 473	Tables 1352
RobotOM.IRobotWindLoadsSimulationParams.Velocity 474	Time history analysis parameters 360
	Tube connection 2031
	<b>U</b>
	Uniform access to structure components 29
	Units and formats 1274
	<b>V</b>
	Values describing load records 38
	Views and layouts 1352
	Views for HTML format files 1460
	Views for RTF format files 1455

W

Wind loads simulation 468