Autodesk Robot Structural Analysis 2012

Robot Steel API

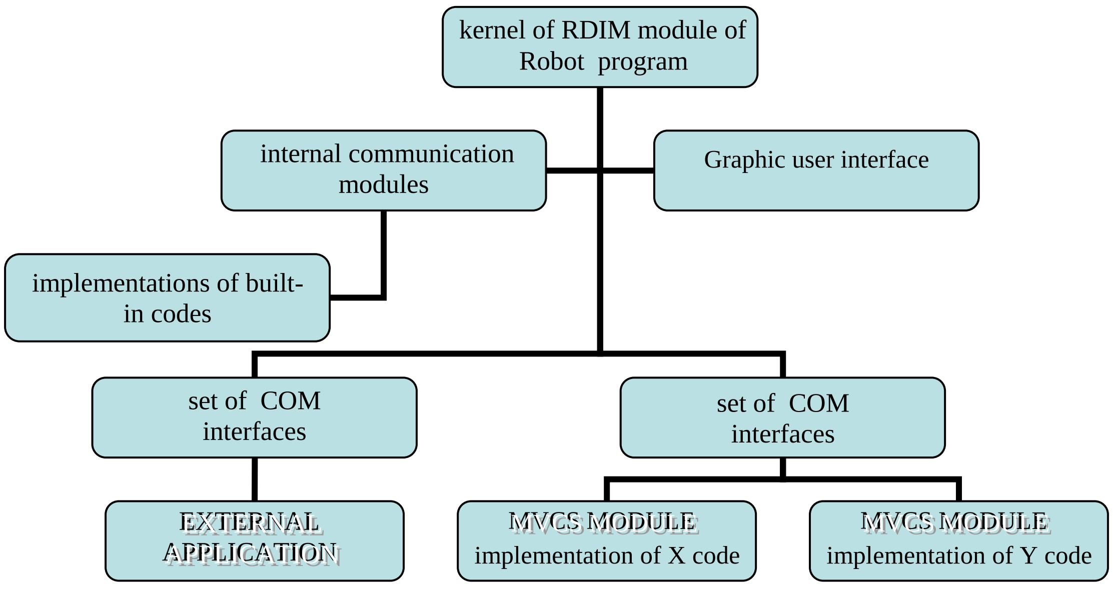
Design of Steel and Timber Elements

Description of COM interfaces and rules of operation and implementation of the code module

Contents

1. Introduction

The Robot program includes a complex COM interface structure which enables the cooperation with external applications. The steel/timber member design module, being a part of the Robot program, is both a client and a server for external COM modules implementing a code-dependent calculation algorithm, dialog boxes for edition of calculation parameters and for presentation of calculation results. In order to improve legibility, in the further part of the document the COM module, which implements the calculation code service for a single member, will be called MVCS (Member Verification Code Service).
The internal module of Robot analogous to MVCS will be called RDIM module.
External applications, using an appropriate set of interfaces, may take adavantage of practically all calculation capabilities of RDIM module and may be extended to include new code services based on the implementation of MVCS modules which apply – for cooperation with RDIM module – a set of interfaces designed specially for this purpose.
The situation described above is illustrated by the following diagram:


kernel of RDIM module of Robot program
internal communication modules
Graphic user interface
implementations of built-in codes
set of COM interfaces
set of COM interfaces
EXTERNAL APPLICATION
MVCS MODULE implementation of X code
MVCS MODULE implementation of Y code

The best method of implementing a code service is to use a framework of such a module written in C++ and delivered together with the Robot program on the installation CD. Through applying source files and a project for Microsoft Visual C++ provided there, a fully functional model of MVCS module can be generated. Once registered, it is a functionally fully operational code service. Modification of a source code and adding a new one enables achieving implementation of a required code fairly quickly.

2. Calculation Rules

The process of structure design comprises several stages. First, structure geometry and loads applied to it are defined. Next, internal forces and displacements are calculated. Afterwards, check of code conditions takes place, structure elements are designed and if need be, modified. A final stage involves verification of all structure elements and if all of them satisfy code conditions, then the design process is completed.

2.1 Assumptions

For the needs of RDIM module, the following is assumed:

- **bar** is an object representing a single structure element (column, spandrel beam, purlin, bracing, etc.). It is given a number, name and label assigned to it, which identifies a set of code parameters,
- **superbar** may comprise a single bar element or may be a series of successive structure elements forming a column, spandrel beam, etc. It is an object of RDIM module and in this module it can be generated and deleted. It is given a number, name and label assigned to it, which identifies a set of code parameters,
- **group** is an object of RDIM module representing a list of bars. It is a set of structure bars to which the user wants to assign the same section. Groups are defined in order to limit the variety of sections in a designed structure. A group may be created and deleted. It is given a number, name and material assigned to it. It also has a section list associated with it. These sections make up a list with a certain order, which means that two identical groups with the same section set are not the same object to the design process.
- **calculation element** is a part of a bar generated in the process of preparing a structure for calculations and unlike a structure element such as bar, is not directly visible to a user. A calculation element is usually a bar and is ascribed all its attributes. A bar is divided into calculation elements smaller than the bar itself in situations when for example a given bar is adjoined by other bar, which thus divides it – at the point where they meet – into smaller calculation elements. Such smaller elements inherit all the properties of the bar within the area in which they coincide with it.
- **each structure bar or superbar** has a label assigned to it which identifies code parameters.
- **calculation points** are points on a bar for which calculations are performed. They may be determined in two ways:
    - by specifying a number of points over the bar length (points are uniformly distributed over the bar length)
    - by providing coordinates of characteristic points

For each calculation point – to the module implementing a regulation of code calculations a few sets of data are provided which allow performing appropriate calculations. These are:

- set of code parameters for a bar,

- set of data describing section parameters, both at a calculation point and averaged values for the entire superbar,
- set of internal forces at a calculation point,
- material parameters ,
- general parameters of calculations performed

2.2 Minimum Calculation Requirements

To ensure correct work of RDIM module, it is required that MVCS code service implement six COM interfaces. These include:

- **IRDimClient** – interface which enables RDIM module to become a client of MVCS server, it provides access to its code service,
- **IRDimCodeService –** implementation of a code service,
- **IRDimMembDefData –** implementation of code-defined properties assigned to a given label,
- **IRDimMembDef –** inherits from IRDimMembDefData and additionally describes certain characteristics of implementation of a given code.
- **IRDimMembCalc –** implementation of code calculations
- **RDimMembRes –** interface which enables RDIM module to present results of code calculations of a bar.

2.3 Designations and Definitions

RDIM module carries out complex calculations taking advantage of modules implementing code services, being either built-in or external modules, accessible via COM.
There are four types of sequential calculations available:

- **bar verification** is a process of bar check based on the code specification. The check runs bar by bar. For each successive load case and each possible component each calculation element of a bar is checked. A calculation element is checked at each point determined in the calculation configuration. The process of bar check based on the code specification is aimed at finding such an intermediate point on a bar, such a load case and such a bar element, for which parameters of code criteria are the worst.
- **group verification** is a process of verification of all bars from the group with material properties assumed for a group considered in calculations. In this case calculation results are the verification results for the bar in a group with the greatest ratio.
- **group design** is a process of section selection performed in such a way so that for each defined section family a section is chosen which fulfills the code requirements ensuring the ratio as close as possible to that defined in the calculation configuration. The selection consists in the next section being taken from the section list of a given family if the section considered at the given stage of calculations fails to satisfy the

code requirements. Group design is a process of group verification modified in the manner described above, during which successive sections which do not meet the code requirements are rejected. Consecutive sections are being rejected until the first one that satisfies code conditions is found. The modified process of group verification just described is carried out for each section family belonging to a set of group's sections.

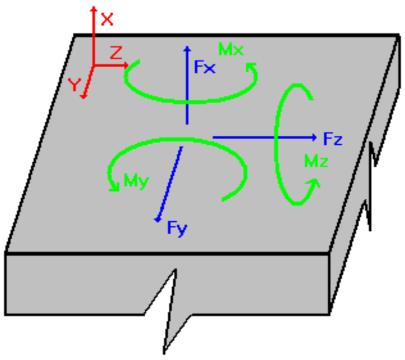
- **group optimization** is the group design, however, in the calculation process additional parameters are taken into account. They are grouped into options and may be considered collectively. Each of these options represent a certain section parameter taken into consideration in calculations:
  - *weight* – if this option is switched on, then the section weight is taken into account and thus the lightest section in a given group is being searched among the sections that fulfill the code criteria
  - *maximum section height* - if this option is switched on, then the maximum section height whose value is determined by the user, is taken into account
  - *minimum section height* - if this option is switched on, then the minimum section height whose value is determined by the user, is taken into account
  - *minimum flange thickness* - if this option is switched on, then the minimum thickness of section flange whose value is determined by the user, is taken into account.
  - *minimum web thickness* - if this option is switched on, then the minimum thickness of section web whose value is determined by the user, is taken into account

  An additional option are *calculations for a full section set*. If it is activated, then during calculations whole section families – from the first section to the last one - are checked. The check normally ends when the section which satisfies code conditions in all the calculation stages is found. It is significant, particularly when not all the sections in the database are arranged in an ascending order, i.e. when the next section is not always "larger" than the preceding one.

At each calculation point MVCS module has access to data that enables performing appropriate calculations. This is made possible by the following interfaces:

- **IRDimMembDef** – bar parameters,
- **IRDimProfDef** – section parameters,
- **IRDimEffDef –** internal forces.

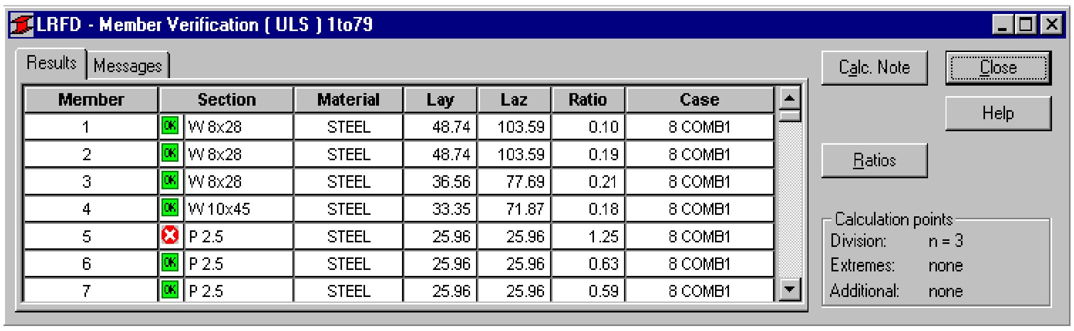
Interpretation of the basic set of internal forces is illustrated in the drawing beside.



- **IRDimMatDef** – material parameters.
- **IRDimCalcState** - general parameters of the calculations performed. The following information is included here: calculation type, calculation point or e.g. efficiency ratio.

2.4 Presentation and Interpretation of Results in RDIM Module

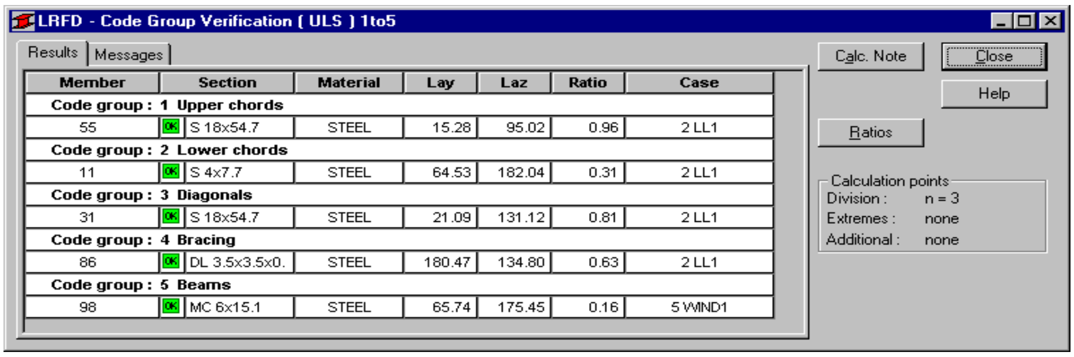
In RDIM module calculation results are presented concurrently with performing the calculations. Short information about every verified member or designed group is instantly displayed in the sheet intended specially for this purpose and provided in the Short Results dialog box. During verification of members each line represents a verified member. In verification of groups, for each group a similar line is displayed, representing a member with the greatest ratio in a group. During group design presentation of short results takes a different form. In this case, for each group maximally three lines per each section family in a group are displayed. Each of maximally three lines presents results for the member with the greatest ratio in the group calculated with parameters of an appropriate section assumed. These are the parameters of the section preceding the designing section, parameters of the designing section and those of the section that is the next after the designing section. If none of the sections in a family meets the code requirements, then only one line is displayed presenting calculation results for the last section in a family.
Below is shown the dialog box presenting example calculations of member verification according to LRFD code.



Member		Section	Material	Lay	Laz	Ratio	Case
1	OK	W 8x28	STEEL	48.74	103.59	0.10	8 COMB1
2	OK	W 8x28	STEEL	48.74	103.59	0.19	8 COMB1
3	OK	W 8x28	STEEL	36.56	77.69	0.21	8 COMB1
4	OK	W 10x45	STEEL	33.35	71.87	0.18	8 COMB1
5	✗	P 2.5	STEEL	25.96	25.96	1.25	8 COMB1
6	OK	P 2.5	STEEL	25.96	25.96	0.63	8 COMB1
7	OK	P 2.5	STEEL	25.96	25.96	0.59	8 COMB1

Calculation points
Division: n = 3
Extremes: none
Additional: none

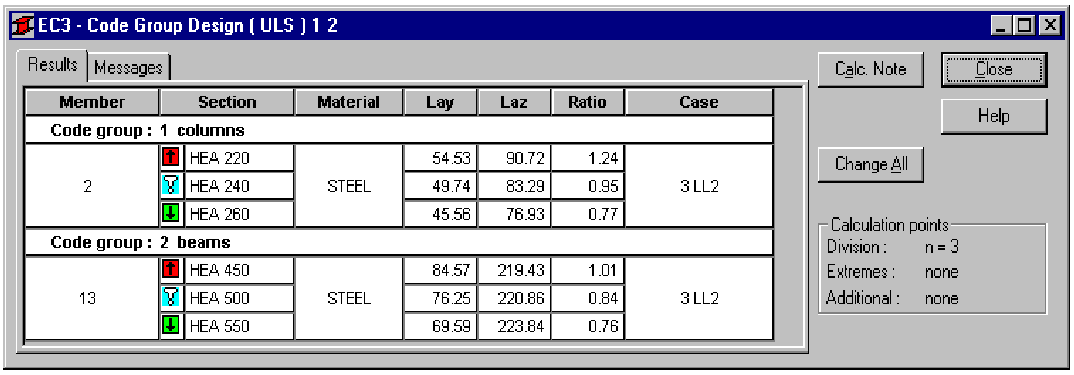
For calculations of group verification arrangement of lines in the dialog box is different.



Member		Section	Material	Lay	Laz	Ratio	Case
Code group : 1 Upper chords							
55	OK	S 18x54.7	STEEL	15.28	95.02	0.96	2 LL1
Code group : 2 Lower chords							
11	OK	S 4x7.7	STEEL	64.53	182.04	0.31	2 LL1
Code group : 3 Diagonals							
31	OK	S 18x54.7	STEEL	21.09	131.12	0.81	2 LL1
Code group : 4 Bracing							
86	OK	DL 3.5x3.5x0.	STEEL	180.47	134.80	0.63	2 LL1
Code group : 5 Beams							
98	OK	MC 6x15.1	STEEL	65.74	175.45	0.16	5 WIND1

Calculation points
Division : n = 3
Extremes : none
Additional : none

For calculations of group design with optimization according to EC3, the dialog box looks as shown below:


EC3 - Code Group Design (ULS) 1 2

Results | Messages

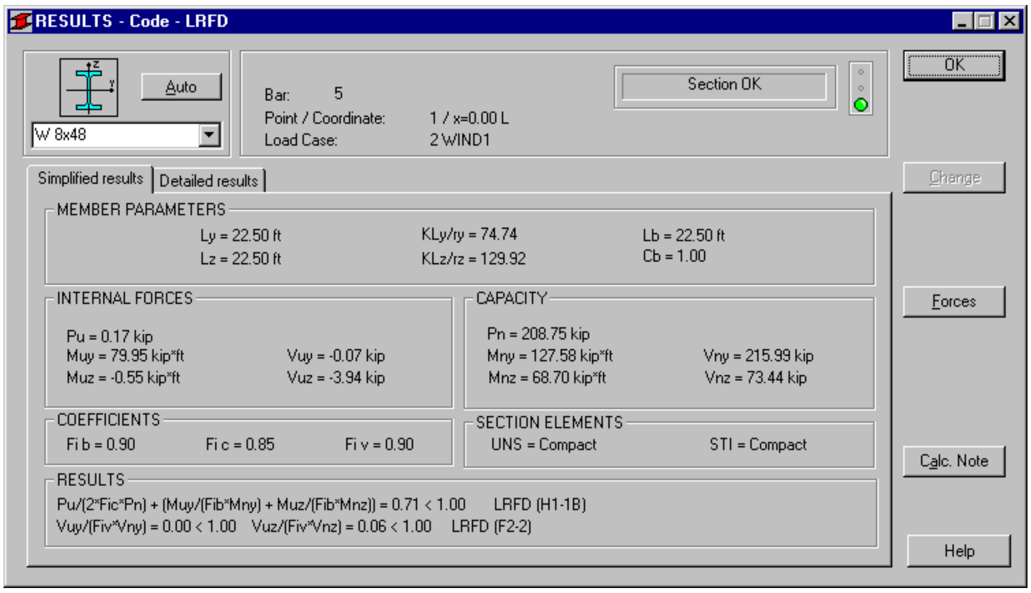
Member		Section	Material	Lay	Laz	Ratio	Case
Code group : 1 columns							
2	🟥	HEA 220	STEEL	54.53	90.72	1.24	3 LL2
	🟦	HEA 240		49.74	83.29	0.95	
	🟩	HEA 260		45.56	76.93	0.77	
Code group : 2 beams							
13	🟥	HEA 450	STEEL	84.57	219.43	1.01	3 LL2
	🟦	HEA 500		76.25	220.86	0.84	
	🟩	HEA 550		69.59	223.84	0.76	

Calculation points
Division : n = 3
Extremes : none
Additional : none

🟩 symbol indicates a section that fulfills the code requirements, whereas ✖ symbol denotes a section which do not satisfy them.
If a member is unstable, then it is designated with ❗ symbol, while an optimal section in a group is given 🟦 symbol.
Information about a calculation result necessary to display the relevant symbols is returned directly by the CalculMember method of the IRDimMembCalc interface. The ‘Ratio’ parameter is displayed using the GetRatio method also of this interface.
Clicking on any line representing calculation results opens a dialog box aimed at thorough presentation of all the parameters and results calculated by the code. This dialog box consists of a part which shows characteristic information about the calculations performed and maximally three tabs presenting in detail results obtained in the process of member calculations. The first tab ‘Simplified Results’ constitutes a set of the most important parameters needed by an engineer to become quickly familiar with the member situation in view of the code. Results presented here concern calculations of Ultimate Limit States. RDIM module prepares this tab using the IRDimMembRes interface implemented in MVCS module. The CreateResWnd method of this interface activated with the I_DMRWT_ULTIMATE_WND parameter returns a handle to the dialog box created by MVCS service. If calculations of Ultimate Limit States are switched on, then there is a third tab showing results of member calculations for this calculation type. If MVCS module does not implement codes for this type of calculations, then they are performed by RDIM module in a standard manner and presented in a typical dialog box implemented in this module. If a code service contains its own regulation on calculations of Serviceability Limit States - typical only of a given code, then it is this module that performs calculations and generates an appropriate dialog box. Then RDIM module activates the CreateResWnd metod of the IRDimMembRes interface with the I_DMRWT_DETAILED parameter.
The third tab ”Detailed Results” is a sheet showing intermediate results, references to paragraphs and formulas applied in calculations. This sheet includes results of calculations of both Ultimate Limit States and Serviceability Limit States. Standard parts of this sheet are generated by RDIM module. All other items of code-dependent information are created in MVCS module through sequential activation of the relevant methods from the IRDimMembRes interface. These methods include:

- GetDefMaxLineNo
- IsDefLineActive
- GetDefLineComponent
- GetMaxLineNo
- IsLineActive
- GetLineType
- GetLineComponent

Below is shown the dialog box presenting in detail results of calculations according to LRFD for Ultimate Limit States:



W 8x48

Bar: 5
Point / Coordinate: 1 / x=0.00 L
Load Case: 2 WIND1

Section OK

MEMBER PARAMETERS
Ly = 22.50 ft KLy/ry = 74.74 Lb = 22.50 ft
Lz = 22.50 ft KLz/rz = 129.92 Cb = 1.00

INTERNAL FORCES
Pu = 0.17 kip
Muy = 79.95 kip*ft Vuy = -0.07 kip
Muz = -0.55 kip*ft Vuz = -3.94 kip

CAPACITY
Pn = 208.75 kip
Mny = 127.58 kip*ft Vny = 215.99 kip
Mnz = 68.70 kip*ft Vnz = 73.44 kip

COEFFICIENTS
Fi b = 0.90 Fi c = 0.85 Fi v = 0.90

SECTION ELEMENTS
UNS = Compact STI = Compact

RESULTS
Pu/(2*Fic*Pn) + (Muy/(Fib*Mny) + Muz/(Fib*Mnz)) = 0.71 < 1.00 LRFD (H1-1B)
Vuy/(Fiv*Vny) = 0.00 < 1.00 Vuz/(Fiv*Vnz) = 0.06 < 1.00 LRFD (F2-2)

There is the 'Calculation Note' button provided in the above dialog box, which when pressed generates a calculation note in an RTF file format. A calculation note is generated based on a template whose variables are replaced with the relevant values. RDIM module employs the IRDimMembRes interface for this purpose, activating appropriate methods sequentially. These are:

- BlockCount
- IsStatement
- ReplaceMark
- RecognizedPQ
- CurrentBlock

The dialog box contains, as well, the 'Forces' button, which when pressed activates the ModalEditManualParams method from the IRDimCodeService interface implemented in MVCS module. This function displays the dialog box for manual calculations. For communication and data transfer from this dialog box to RDIM module and vice versa the IRDimManCalcPar interface is used.

In addition, MVCS module may implement calculations of **detailed analysis** and then it has to inform RDIM module about such a situation, since this module has to display the button activating this option in the dialog box with detailed presentation of calculations results. This information is passed by means of the IRDimMembRes interface. RDIM module calls up the ResOfCalc method with the I_DMRCT_DETAILED parameter. If it returns the I_DMCRV_CORRECT value, it means that the dialog boxes and calculation codes of the detailed analysis are implemented. If that is the case, then after clicking on the detailed analysis button RDIM module activates the CreateResWnd method with the I_DMRWT_DETAILED_WND parameter which should result in creating an appropriate dialog box. This dialog box and its derivatives are the implementation of the detailed analysis option visible to a user. Once the relevant calculations are carried out, MVCS service may instruct RDIM module to generate a calculation note for this analysis. Such a note is generated similarly to that discussed earlier, on the basis of an appropriate template and with the use of the IRDimMembDef interface. To generate a calculation note, MVCS module activates the SendMessage method from the IRDimConnection interface whose implementation is included in RDIM module. This method is called up with the I_DCM_DETAILED_ANALYZE_RTF_PRINT parameter and with the name of RTF file containing the template.

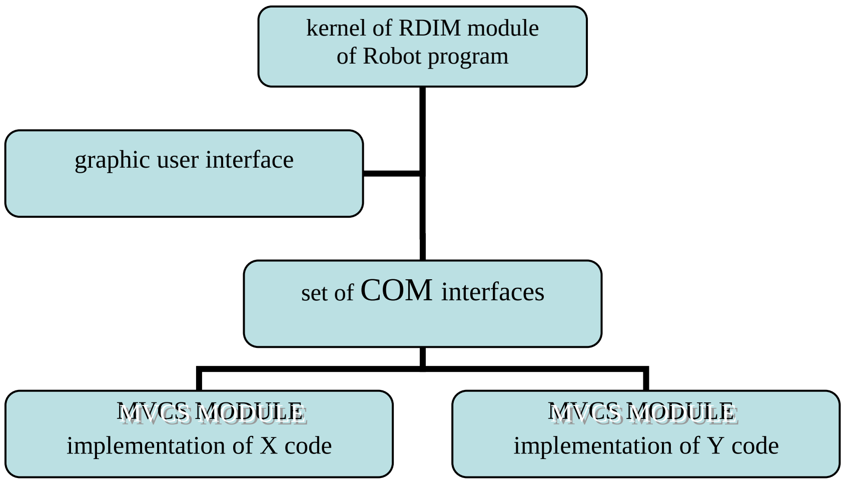
Other aspects of result presentation will be discussed in the chapter concerned with implementation of MVCS module.

3. Implementation of the Code Module

For the purposes of implementation of the MVCS code service, a framework of such a module written in C++ has been generated. This chapter presents source files which together with a project for Microsoft Visual C++ enable generation of a fully functional model of MVCS module. Here is a description of how to modify the source code as well as how and where to add a new one to achieve implementation of a required code.

3.1 Introduction

RDIM module together with servers implementing code services forms a closely cooperating structure of components shown below.



3.2 General Structure of MVCS Module

MVCS module is in fact implementation of six basic interfaces. These are:

- IRDimClient
- IRDimCodeService
- IRDimMembDefData
- IRDimMembDef
- IRDimMembCalc
- IRDimMembRes

Moreover, to enable external applications to apply MVCS module properties, it is necessary to define, implement and make accessible two additional interfaces to support non-standard member parameters specific of a given code and to get numeric calculation results. The first one is a parameter of the CodeParams attribute of the IRDimMembDefData interface, whereas the second one is returned by the CodeResults of the IRDimMembRes interface and is accessible to external applications from the IRDimDetailedResults interface.

3.3 General Description of COM Interfaces

Below is a description of all the interfaces used during interactive cooperation of MVCS module with RDIM module which, as it has already been mentioned, is an integral part of the Robot system.

IRDimStreamType

Definition of the set of identifiers for the IRDimStream interface.

Attributes

I_DST_LONG : = 1
Determines Long-type data.

I_DST_DOUBLE : = 2
Determines the double-type data.

I_DST_TEXT : = 3
Determines the text data.

IRDimStream

- **Implementation in RDIM module.**
  This interface implements any data stream so that it allows a free data exchange between RDIM and MVCS modules. Data saving and reading is the same as saving to a file but there are three separate files (streams), one for each type of data.

Attributes:

<GET>EoL : bool
Identifies the end of the data stream (long type)

<GET>EoD : bool
Identifies the end of the data stream (double type)

<GET>EoT : bool
Identifies the end of the text data stream

Functions:

Clear ()
This method removes all the data from three streams.

Size (_type : IRDimStreamType) : long
The method gives the stream size with the given type of data.

SeekSet (_type : IRDimStreamType, pos : long)
Sets the current stream position with the given type of data.

WriteLong (val : long)
Saves the data of long-type and changes stream's position by 1 value (one step) upwards

ReadLong () : long
Reads data of long type from current position and changes stream's position by 1 value (one step) upwards.

WriteDouble (val : double)
Saves the double-type data and changes stream's position by 1 value (one step) upwards.

ReadDouble () : double
Reads the double-type data from the current position and changes stream's position by 1 value (one step) upwards.

WriteText (val : string)
Saves text data and changes stream's position by 1 value (one step) upwards.

ReadText () : string
Reads text data from current position and changes stream' s position by 1 value (one step) upwards.

IRDimMembDefGuidType

Definition of identifiers defining the type (given by ClientID method) of the component identifier which implements IRDimClient interface.

Attributes:

I_DMDGT_LACK : = 0
Identifier of the component which implements the IRDimClient interface
does not exist. This means the code-defined code embodied in the RDIM module.

I_DMDGT_CLSID : = 1
The form of IRDimClient interface implementing component is CLSID.

I_DMDGT_PROGID : = 2
The form of IRDimClient interface implementing component is ProgID.

IRDimMembDefType

Definition of the set of identifiers determining member type for IRDimMembDef interface.

Attributes:

I_DMDT_USER : = 1
Determines user's member with any characteristics.

I_DMDT_MEMBER : = 2
Determines default member parameters.

I_DMDT_BEAM : = 3
Determines default beam parameters.

I_DMDT_COLUMN : = 4
Determines default column parameters.

IRDimMembDefMatType

Definition of the set of identifiers defining member material (timber or steel) for IRDimMembDef interface.

Attributes:

I_DMDMT_STEEL : = 1
Determines the steel member.

I_DMDMT_TIMBER : = 2
Determines the timber member.

IRDimMembDefLengthDataType

Definition of the set of identifiers defining the direction of the member length in given planes (for buckling calculations) for IRDimMembDef interface.

Attributes:

I_DMDLDT_LENGTH_Y : = 1
Defines member length – buckling in Y axis plane.

I_DMDLDT_LENGTH_Z : = 2
Defines length in the Z axis plane.

I_DMDLDT_LENGTH_U : = 3
Defines the length in the U axis plane (maximum inertia)

I_DMDLDT_LENGTH_V : = 4
Defines the length in the V axis plane (minimum inertia)

IRDimMembDefBucklingDataType

Definition of the set of identifiers defining the plane of the member buckling for IRDimMembDef interface.

Attributes:

I_DMDBDT_BUCKLING_Y : = 1
Defines buckling in the Y-axis plane.

I_DMDBDT_BUCKLING_Z : = 2
Defines buckling in the Z axis plane.

I_DMDBDT_BUCKLING_U : = 3
Defines buckling in the U axis plane (maximum inertia)

I_DMDBDT_BUCKLING_V : = 4
Defines buckling in the V axis plane (minimum inertia).

IRDimMembDefDispDataType

Definition of the set of identifiers defining member nodes displacement (in global coordinate system) for IRDimMembDef interface.

Attributes:

I_DMDDDT_DISP_X : = 1
Defines displacement in the X axis plane.

I_DMDDDT_DISP_Y : = 2
Defines displacement in the Y axis plane.

IRDimMembDefDeflDataType

Definition of the set of identifiers defining the direction of member deflection for IRDimMembDef interface.

Attributes:

I_DMDDDT_DEFL_Y : = 3
Defines deflection in the Y axis plane.

I_DMDDDT_DEFL_Z : = 4
Defines deflection in the Z axis plane.

IRDimMembDefIntPsDataType

Definition of the set of identifiers defining the plane of buckling or the level (flange) lateral buckling in the intervals limited by intermediate points for IRDimMembDef interface.

Attributes:

I_DMDIPDT_BUCKLING_Y : = 1
Defines buckling in the Y axis plane.

I_DMDIPDT_BUCKLING_Z : = 2
Defines buckling in the Z axis plane.

I_DMDIPDT_LBUCKLING_U : = 3
Defines the lateral buckling of the upper flange.

I_DMDIPDT_LBUCKLING_L : = 4
Defines the lateral buckling of the lower flange.

IRDimMembDefInitDeflType

Definition of the set of identifiers which define the way of taking into account cambers during displacements control.

Attributes:

I_DMDIDT_LACK : = 1
Means that camber is not taken into account.

I_DMDIDT_USER : = 2
Means that camber is taken into account as a numerical value assigned by user.

I_DMDIDT_AUTO : = 3
Means that camber is automatically taken into account (additional parameters are defined in the calculation configuration interface).

IRDimMembDefUserInitDeflType

Definition of the set of identifiers defining the direction of numerically assigned camber used during displacements control.

Attributes:

I_DMDUIDT_Y : = 1
Defines camber in Y direction.

I_DMDUIDT_Z : = 1
Defines camber in Z direction.

IRDimMembDef

- **Implementation in MVCS module**

  In Robot system codes characteristics defining a member are stored as data related to a certain label. A specific member in the structure is perceived as an object with attributed particular label identifying data with code characteristics. A specific label defined by its name may be attributed to many members in the structure but not inversely. IRDimMembDef interface has been defined so that it is possible to define code characteristics of the member, save them in the Robot system and attribute it to any member in the structure and carry out code calculations using these data. This interface inherits from the IRDimMembDefData interface.

Attributes:

<GET>GuidTye : IRDimMembDefGuidType

Returns the information whether the identifier of the IRDimClient interface implementing component given by ClientID method has the form of CLSID (like in C++ for example) or if it's form is ProgID (like in Basic for example).

<GET>ClientID : string

Returns a unique identifier of the IRDimClient interface implementing component. This identifier (such as GUID generated by the standard tool furnished with Visual C++ packet in the form {530C6482-BD05-11D3-9325-0050DA767C8D}) is necessary to RDIM module to run an appropriate application being MVCS module and being able to interpret specific data characteristic of a member.

<GET>MatType : IRDimMembDefMatType

Returns the type of member material.

Length : double

Defines the member length. If this interface defines the type of the member (label) this attribute is not fixed and its default value is 1.0. But if the RDIM module creates such an interface to calculate a specific member in the structure, it calls the "Retrieve" method and besides that it defines the Length attribute as the value corresponding to the real length of the member.

Functions:

IsBuckCoefConst (_type : IRDimMembDefBucklingDataType) : bool

The value given by this method informs the RDIM module whether, during the calculation of bar which code characteristics are defined with data transferred by this interface, the method: “CalculBuckling.” Should be called in appropriate iterations from IRDimMembCalc interface. Calling the above-mentioned method causes the update of buckling length coefficients for calculated member after, for instance, changing the profile during dimensioning. This update occurs in the MVCS module in which the IRDimMembCalc interface is implemented.

IsClientBuckCoefServiceYZ (_type : IRDimMembDefBucklingDataType) : bool

The value returned by this method informs the RDIM module whether the MVCS module implements the code providing support for updating values of the buckling length coefficients for the calculated member after e.g.: changing the section during design procedure.

AreAdjoinParamsYZ (_type : IRDimMembDefBucklingDataType) : bool

The value returned by this method informs RDIM module whether MVCS module implements the code providing support for updating values of the buckling length coefficients for the calculated member after e.g.: changing the section during design procedure.

IRDimMembDefData

- **Implementation in MVCS module**

  Interface defining code characteristics assigned to a label. It is a base for IRDimMembDef interface which, additionally, defines some implementation characteristics for the given code.

Attributes:

Name : string

Defines the name of label or member. If the type of member is defined as a label, this attribute defines its name. And if the RDIM module creates the mentioned interface for the needs of calculation of a specific member in the structure, this attribute defines the name of the member.

Type : IRDimMembDefType

Defines member type.

InitDeflType : IRDimMembDefInitDeflType

Defines the way of taking into account camber during displacements control.

CantileverMode : bool

Defines additional member characteristic. – the cantilever. This characteristic is used in calculations of serviceability.

PipeMode : bool

Defines the additional member characteristic. – pipe-shaped. This characteristic is used in the ULS calculations.

CodeParams : IDispatch
Defines member parameters in a specific code.

Functions:

Store (str : IRDimStream)
This method copies code data about the type of the member (label) from the IRDimStream to internal data structures of the IRDimMembDefData interface class.

Retrieve (str : IRDimStream)
This method copies code data about the type of the member (label) from internal data structures of the IRDimMembDefData interface class to the IRDimStream.

SetLengthYZUV (_type : IRDimMembDefLengthDataType,val double) : double
Sets the absolute or relative length of the member in the assigned buckling direction (assigned by the user as one of the code parameters). If the number given is negative its value is considered as relative length in respect of the real member length. Whereas the positive value is considered as assigned absolute length.

LengthYZUV (type : IRDimMembDefLengthDataType) : double
Returns the value fixed by the SetLengthYZUV method.

SetDisplacementXY (type : IRDimMembDefDispDataType, val : bool)
Starts / stops calculations of displacement verification in the assigned direction. The verification of mode set by this method is performed by IsDisplacementXY method.
IsDisplacementXY (type : IRDimMembDefDispDataType) : bool
The value returned by this method informs the RDIM module whether the displacements in the assigned direction should be verified during the calculations for a specific member which code characteristics are described by means of data transferred by this interface.

SetDisplXYRelLimit (type : IRDimMembDefDispDataType, val: double)
Sets the limit displacement values in the assigned direction.

DisplXYRelLimit (type : IRDimMembDefDispDataType) : double
Returns displacements limit values in the assigned direction.

SetDeflectionYZ (type : IRDimMembDefDeflDataType, val : bool)
Starts/stops the calculation of deflection the assigned direction. The verification of mode set by this method is performed by IsDeflectionXY method.

IsDeflectionYZ (type : IRDimMembDefDeflDataType) : bool
The value given by this method informs the RDIM module whether deflections in the assigned direction should be verified during analysis for a specific member which code characteristics are described by data transferred by this interface.

SetDeflYZRelLimit (type : IRDimMembDefDeflDataType, val : double)
Sets deflection limit values in the assigned direction.

DeflYZRelLimit (type : IRDimMembDefDeflDataType) : double
Returns the limit deflection values in the assigned direction.

IntPsClear(type : IRDimMembDefIntPsDataType)
Removes all intermediate points (defined by type parameter) defined on the member (bracing).

IntPsCoordNum (_type : IRDimMembDefIntPsDataType) : long
Gives the number of defined intermediate points (braces) on the member.

SetIntPsCoordValueType (type : IRDimMembDefIntPsDataType, isRelValue : bool)
Sets the way of assigning intermediate points (bracing). This setting is read by IntPsCoordValueType method.

IntPsCoordValueType (type : IRDimMembDefIntPsDataType) : bool
The returned value informs the RDIM module how intermediate points (brace) were assigned on the member. TRUE value means that "IntPsCoordValue" method gives values relative to member's length. Otherwise this method gives absolute values.

SetIntPtValues (type : IRDimMembDefIntPsDataType, no : long, coord : double, coeff : double)
Sets the coordinate of the intermediate point and the coefficient of buckling / lateral buckling length for the range corresponding to this point.. Points' numbers are counted starting from 1 up to the value given by the "IntPsCoorNum" method.

IntPtValues (type : IRDimMembDefIntPsDataType, no : long, coord : double*, coeff : double*)
Returns the coordinates of the intermediate point and the coefficient of buckling / lateral buckling length for the range corresponding to the returned point. Points' numbers are counted starting from 1 up to the value given by the "IntPsCoorNum" method.

SetStructureSwayYZ (type : IRDimMembDefBucklingDataType, is_sway : long)
Sets the flag of structure type – sway / non-sway.

IsStructureSwayYZ (type : IRDimMembDefBucklingDataType) : long
Returns the flag of structure type – sway / non-sway.

StoreAdjoinParamsYZ (type : IRDimMembDefBucklingDataType, in : IDispatch*)
This method copies code data about adjoining bars from an appropriate interface, presently it is IRDimAdjoinParams, to internal structures of class data for IRDimMembDef Data interface.

RetrieveAdjoinParamsYZ (type : IRDimMembDefBucklingDataType, in : IDispatch*)
This method copies code data about adjoining bars from internal structures of class data for IRDimMembDefData interface to an appropriate interface. At present it is the IRDimAdjoinParams interface.

IRDimAdjoinMembNo

Definition of a set of identifiers indicating successive structure members being adjoining members.

Attributes:

I_DAMN_NO_1: = 1
First adjoining member.

I_DAMN_NO_2: = 2
Second adjoining member.

I_DAMN_NO_3: = 3
Third adjoining member.

I_DAMN_NO_4: = 4
Fourth adjoining member.

I_DAMN_NO_5: = 5
Fifth adjoining member.

I_DAMN_NO_6: = 6
Sixth adjoining member.

IRDimAdjoinMembPos

Definition of a set of identifiers determining position of a member in a structure.

Attributes:

I_DAMP_POSITION_NORMAL: = 1
Vertical position.

I_DAMP_POSITION_90ROTATED: = 2
Horizontal position.

IRDimAdjoinMembSuppCond

Definition of a set of identifiers determining the support method on the other end of an adjoining member.

Attributes:

I_DAMSC_SUPPCOND_PINNED: = 1
Pinned support.

I_DAMSC_SUPPCOND_STIFFENED: = 2
Fixed support.

I_DAMSC_SUPPCOND_MIXED: = 3
Combined support.

IRDimAdjoinParamsEditWndType

Definition of a set of identifiers determining a type of the dialog box for definition of adjoining memebers.

Attributes:

I_DAPEWT_WNDTYPE_1MEMBER: = 1
Dialog box used for definition of one adjoining member.

I_DAPEWT_WNDTYPE_3MEMBER: = 2
Dialog box used for definition of three adjoining members.

I_DAPEWT_WNDTYPE_6MEMBER: = 3
Dialog box used for definition of six adjoining members.

IRDimAdjoinParamsEqMInertia

Definition of a set of identifiers determining equivalent moment of inertia for both the main member and adjoining bars.

Attributes:

I_DAPEMI_MINIMUM: = 1
Equivalent moment of inertia defined as the member’s minimal moment of inertia.

I_DAPEMI_AVERAGE: = 2
Equivalent moment of inertia defined as the member’s medium moment of inertia.

I_DAPEMI_MAXPERCENT: = 3
Equivalent moment of inertia defined as a percentage of the member's maximal moment of inertia.

IRDimAdjoinParams

- **Implementation in RDIM module**

  This interface is used to determine parameters of adjoining members and as a set of data for the module of automation of calculation of the main member buckling coefficient.

Attributes:

MainMembEqMInertia : IRDimAdjoinParamsEqMInertia
The attribute determines equivalent moment of inertia for the main member.

MainMembEqMInertiaPercentVal : double
The attribute determines equivalent moment of inertia for the main member defined as a percentage of the member's maximal moment o inertia.

AdjoinMembEqMInertia : IRDimAdjoinParamsEqMInertia
The attribute determines equivalent moment of inertia for adjoining members.

AdjoinMembEqMInertiaPercentVal : double
The attribute determines equivalent moment of inertia for adjoining members defined as a percentage of the member's maximal moment o inertia.

Functions:

Clear ()
This method deletes entire data.

GetMemberUserNo (no : IRDimAdjoinMembNo, ret : long*)

SetMemberUserNo (no : IRDimAdjoinMembNo, val : long)

SetMomLenByMembList (no : IRDimAdjoinMembNo, in : IRDimStream)

GetMomentOfInertia (no : IRDimAdjoinMembNo, ret : double*)

SetMomentOfInertia (no : IRDimAdjoinMembNo, val : double)

GetLength (no : IRDimAdjoinMembNo, ret : double*)

SetLength (no : IRDimAdjoinMembNo, val : double)

GetMemberPosition (no : IRDimAdjoinMembNo, ret : IRDimAdjoinMembPos*)

SetMemberPosition (no : IRDimAdjoinMembNo, pos : IRDimAdjoinMembPos)

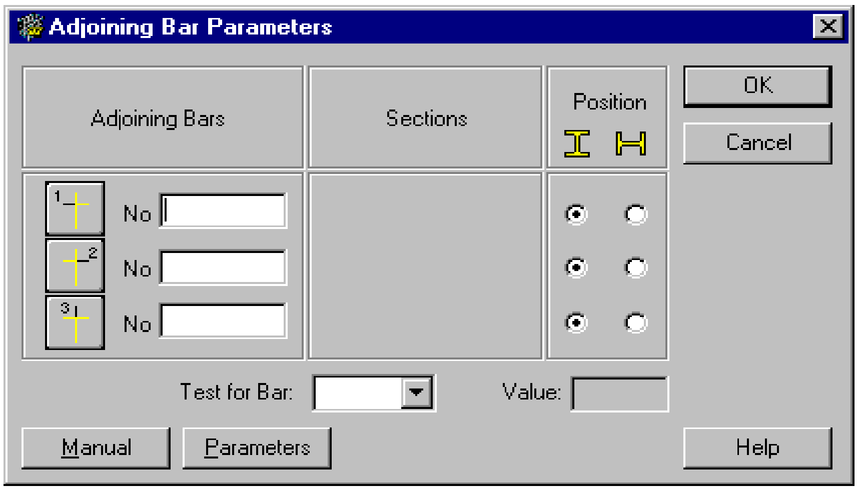
GetMemberSuppCond (no : IRDimAdjoinMembNo, ret : IRDimAdjoinMembSuppCond *)

SetMemberSuppCond (no : IRDimAdjoinMembNo, pos : IRDimAdjoinMembSuppCond)

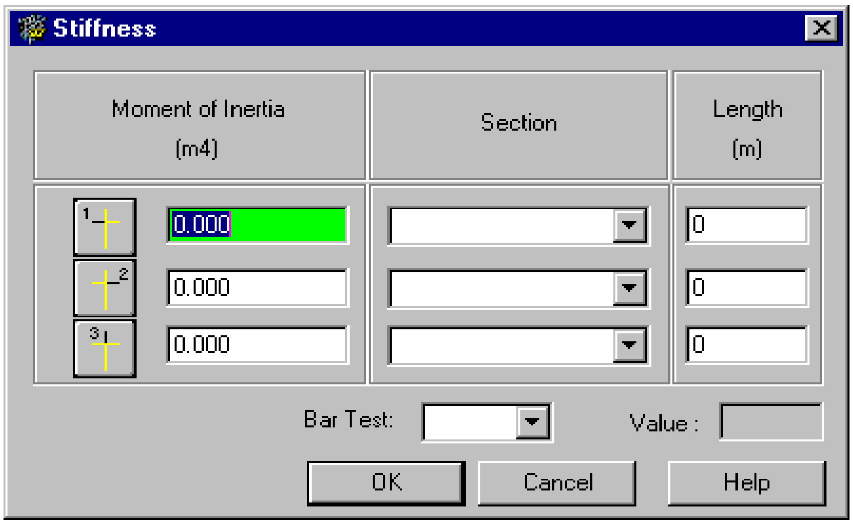
Edit (type : IRDimAdjoinParamsEditWndType, in : IDispatch)
Displays the dialog box used for definition of adjoining members.

The dialog boxes for three adjoining members are shown below. At present, the parameter is not in use.

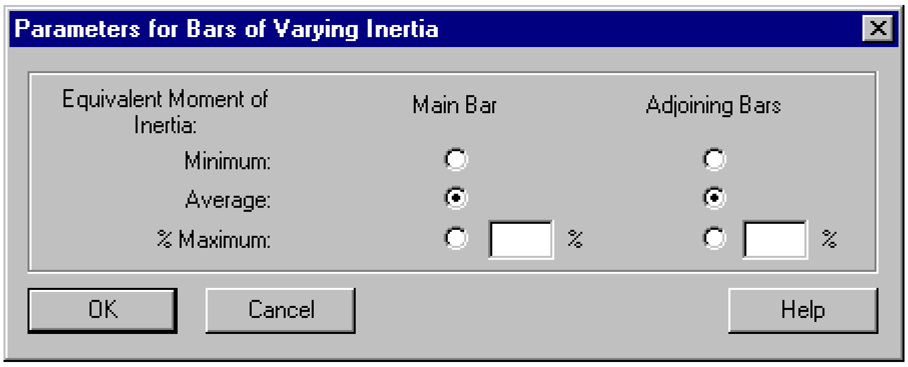
The main dialog box:



Clicking on the ”Manual” button opens the Stiffness dialog box:



The ”Parameters” button in the main dialog box opens the dialog box for definition of the equivalent moments of inertia:


Parameters for Bars of Varying Inertia
Equivalent Moment of Inertia:
Main Bar
Adjoining Bars
Minimum:
Average:
% Maximum:
%
%
OK
Cancel
Help

The exact description of how these dialog boxes work is provided in the user manual for the Robot program.

IRDimEffDefParamType

Definition of the set of identifiers defining the type of parameters related to the internal forces.

Attributes:

I_DEDPT_ACC : = 1
Defines an accidental combination.

I_DEDPT_ELEM_NO : = 2
Defines the internal number of the currently analyzed calculation element (if necessary, the member is automatically divided into so-called calculation elements).

I_DEDPT_MEMB_NO : =3
Defines the internal number of the currently analyzed member.

I_DEDPT_CASE_NO : = 4
Defines the internal number of the load case.

I_DEDPT_COMP_NO : = 5
Defines the successive component number in the code combination case (otherwise it takes the value 1)

I_DEDPT_POINTS_NUM : =6
Defines the number of calculation points assigned by the user (so-called equal intervals points)

I_DEDPT_POINT_NO : = 7
Defines the successive number of the point in which calculation is carried out (in the case of the equal interval point).

I_DEDPT_LOADCLASS : = 8
Defines the load class.

I_DEDPT_MEMB_USER_NO : = 9
Defines user's number of the currently calculated bar.

IRDimEffDefDirType

Definition of the set of identifiers defining the axis.

Attributes:

I_DEDDT_Y : = 1
Defines the Y axis.

I_DEDDT_Z : = 2
Defines the Z axis.

IRDimEffDefIntPsType

Definition of the set of identifiers defining the type of buckling and lateral buckling.

Attributes:

I_DEDIPT_BUCKLING_Y : = 1
Defines buckling in the Y axis plane.

I_DEDIPT_BUCKLING_Z : = 2
Defines buckling in the Z axis plane.

I_DEDIPT_LBUCKLING_U : = 3
Defines the lateral buckling of the upper flange level.

I_DEDIPT_LBUCKLING_L : = 4
Defines the lateral buckling of the lower flange level.

IRDimSimEffDef

- **Implementation in the RDIM module.**
  In case of a manual calculation of a single member the RDIM module (and also MVCS module) uses a simplified interface to support data concerning the internal forces. It includes only the information about the internal forces and additional moments which can possibly be taken into account during this type of calculations.

Functions:

Clear ()
This method removes all the data.

WriteForces (N : double, QY : double, QZ : double, MX : double, MY : double, MZ : double)
Saves the set of basic internal forces.

- N - axial force
- QY, QZ - shear forces in appropriate directions
- MX, MY,MZ - moments in appropriate directions

ReadForces (N : double*, QY : double*, QZ : double*, MX : double*, MY : double*, MZ : double*)
The method allows to read the set of basic internal forces entered by the "WriteForces" function.

WriteValuesSet1 (_type : IRDimEffDefDirType, M1 : double, M2 : double, M12 : double)
Saves the set of moments at the member's origin and end for the given direction.

- M1 - moment at the origin of the member
- M2 - moment at the end of the member
- M12 - relation of moments at the member' s origin and end

ReadValuesSet1 (_type : IRDimEffDefDirType, M1 : double*, M2 : double*, M12 : double*)
The method allows to read the set of moments at the member's origin and end

- M1 -- moment at the origin of the member
- M2 -- moment at the end of the member
- M12 -- relation of moments at the member origin and end

WriteValuesSet2 (_type : IRDimEffDefDirType, MP_MAX : double, MN_MAX : double, M_MID : double, M_1P4L : double, M_3P4L : double)
Saves the set of member's moments for the given direction.

- MP_MAX -- the maximum positive value of the moment for the given member
- MN_MAX -- the minimum positive value of the moment for the given member
- M_MID -- moment in the middle of the member
- M_1P4L -- moment in the 1/4 of the member's length
- M_2P4L -- moment in the 3/4 of the member's length

ReadValuesSet2 (_type : IRDimEffDefDirType, MP_MAX : double*, MN_MAX : double*, M_MID : double*, M_1P4L : double*, M_3P4L : double*)
The method allows to read the set of member's moments for the given direction

- MP__MAX - the maximum positive value of the moment for the given member
- MN_MAX -- the maximum negative value of the moment for the given member
- M_MID -- a moment in the middle of the member
- M_1P4L -- a moment in the 1/4 of member's length
- M_3P4L -- moment in the 3/4 of member's length

IRDimEffDef

- **Implementation in RDIM module**

  For automatic verification, dimensioning and optimization of members and groups, for data support concerning internal forces the RDIM module (and also MVCS module) uses a complex interface which contains information about internal forces and the complementary moments which are used for code calculations.

Attributes:

N : double
Axial force.

QY : double
Shear force in the Y axis plane .

QZ : double
Shear force in the Z axis plane .

MX : double
Moment in the X axis plane .

MY : double
Moment in the Y axis plane .

MZ : double
Moment in the Z axis plane .

Functions:

Clear ()
The method removes all the data.

Set_x_0_1 (val : double)
Sets a relative coordinate for the currently analyzed characteristic point (on the member)

Get_x_0_1 ( ) : double
Returns a relative coordinate of the member point in which the calculations are currently performed. If this value is negative it means that the analysis is currently carried out in the equal interval point and it is possible to recall the ReadParam function with I_DEDPT_POINTS_NUM and I_DEDPT_POINT_NO attributes.

WriteParam (_type : IRDimEffDefParamType, val : long)
Saves the parameters defined by the given attribute.

ReadParam (_type : IRDimEffDefParamType) : long
Returns the parameter defined by the given attribute and previously saved by the "WriteParam" method.

WriteForces (N : double, QY : double, QZ : double, MX : double, MY : double, MZ : double)
Saves the set of basic internal forces.

- N -- axial force

- QY, QZ -- shear forces in appropriate directions
- MX, MY, MZ -- moments in appropriate directions

WriteValuesSet1 (_type : IRDimEffDefDirType, M1 : double, M2 : double, M12 : double)

Saves the set of moments at member' s origin and end for the given direction.

- M1 -- moment at the member's origin
- M2 -- moment at the member' s end
- M12 -- relation of moments at member' s origin and end

Read_M1 (_type : IRDimEffDefDirType) : double

It gives the moment at the member' s origin for the given direction.

Read_M2 (_type : IRDimEffDefDirType) : double

It gives the moment at the end of the member for the given direction

Read_M12 (_type : IRDimEffDefDirType) : double

It gives the relation of moments at member' s origin and end for the given direction

WriteValuesSet2 (_type : IRDimEffDefDirType, MP_MAX : double, MN_MAX : double, M_MID : double, M_1P4L : double, M_3P4L : double)

It saves the set of moments in the member for the given direction.

- MP_MAX -- maximum positive value of the moment for the given member
- MN_MAX -- minimum positive value of the moment for the given member
- M_MID -- moment in the middle of the member
- M_1P4L -- moment in the 1/4 of the member' s length
- M_3P4L -- moment in the 3/4 of the member's length

Read_MP_MAX (_type : IRDimEffDefDirType) : double

Returns the maximum positive value of the moment in the given member for the assigned direction.

Read_MN_MAX (_type : IRDimEffDefDirType) : double

Returns the maximum negative value of the moment in the given member for the assigned direction.

Read_M_MID (_type : IRDimEffDefDirType) : double

Returns the value of the moment in the middle of the member for the assigned direction.

Read_M_1P4L (_type : IRDimEffDefDirType) : double

Returns the value of the moment in the 1/4 of the member length for the assigned direction.

Read_M_3P4L (_type : IRDimEffDefDirType) : double
Returns the value of the moment in the 3/4 of the member length for the assigned direction

WriteIntPsRangeCoeff (_type : IRDimEffDefIntPsType, x_beg : double, x_end : double, coeff : double)
Saves current coordinates of range limit values and the corresponding value of buckling / lateral buckling length coefficient (for bracing in a given direction). The current range is defined by RDIM module as the one inside which there is the current calculation point.

ReadIntPsRangeCoeff (_type : IRDimEffDefIntPsType, x_beg : double*, x_end : double*, coeff : double*)
Reads current coordinates of range limit values and the corresponding value of buckling / lateral buckling length coefficient (for bracing in a given direction).

WriteIntPsEffSet1 (_type : IRDimEffDefIntPsType, M1 : double, M2 : double, M12 : double)
Saves the set of moments at the origin and end of the interval limited by the intermediate points (brace) assigned on the member for the given direction. The current intervals defined by the RDIM module as the one the current calculation point is included in.

- M1 -- moment at the origin of the section
- M2 -- moment at the end of the section
- M12 -- relation of moments at the section origin and end

Read_IntPsEff_M1 (_type : IRDimEffDefIntPsType) : double
Returns the value of the moment for the assigned direction at the origin of the current interval between the given intermediate points. This value is saved by the "WriteIntPsEffSet1" function.

Read_IntPsEff_M2 (_type : IRDimEffDefIntPsType) : double
Returns the value of the moment for the assigned direction at the end of the current interval between the given intermediate points. This value is written by the "WriteIntPsEffSet1" function.

Read_IntPsEff_M12 (_type : IRDimEffDefIntPsType) : double
Returns the value of the moments at the origin and end of the current interval between the given intermediate points (for the assigned direction). This value is written by the "WriteIntPsEffSet1" function.

WriteIntPsEffSet2 (_type : IRDimEffDefIntPsType, MP_MAX : double, MN_MAX : double, M_MID : double, M_1P4L : double, M_3P4L : double)
Enters the set of the moments in the section limited by the assigned intermediate points (brace) on the member for the given direction.

- MP_MAX -- the maximum positive value of the moment for the given section
- MN_MAX -- the minimum positive value of the moment for the given section
- M_MID -- moment in the middle of the section
- M_1P4L -- moment in the 1/4 of the section' s length
- M_3P4L -- moment in the 3/4 of the section' s length

Read_IntPsEff_MP_MAX (_type : IRDimEffDefIntPsType) : double
Returns the maximum positive value of the moment for the assigned direction in the current interval between the given intermediate points. This value is entered by the "WriteIntPsEffSet2" function.

Read_IntPsEff_MN_MAX (_type : IRDimEffDefIntPsType) : double
Returns the maximum negative value of the moment for the assigned direction in the current interval between the given intermediate points. This value is entered by the "WriteIntPsEffSet2" function.

Read_IntPsEff_M_MID (_type : IRDimEffDefIntPsType) : double
Returns the value of the moment for the assigned direction in the middle of the current interval between given intermediate points. This value is entered by the "WriteIntPsEffSet2" function.

Read_IntPsEff_M_1P4L (_type : IRDimEffDefIntPsType) : double
It gives the value of the moment for the assigned direction in the quarter of the length of the current interval between given intermediate points. This value is entered by the "WriteIntPsEffSet2" function.

Read_IntPsEff_M_3P4L (_type : IRDimEffDefIntPsType) : double
It gives the value of the moment for the assigned direction in the 3/4 of the length of the current interval between given intermediate points. This value is entered by the "WriteIntPsEffSet2" function.

IRDimMatDefType

Definition of the set of identifiers defining the type of the member material.

Attributes:

I_DMDT_STEEL : = 1
Defines steel.

I_DMDT_ALUMINIUM : = 2
Defines aluminum.

I_DMDT_WOOD : = 3
Defines timber.

IRDimMatDefValType

Definition of the set of identifiers defining the type of numerical parameter typical for the given material.

Attributes:

I_DMDVT_CS : = 1
reduction factor for shear.

I_DMDVT_E : = 2
Young's modulus or axial elasticity modulus for timber.

I_DMDVT_G : = 3
Shear modulus

I_DMDVT_RE : = 4
Limit of elasticity.

I_DMDVT_RE_AX_COMR : = 5
Strength for axial compression.

I_DMDVT_NU : = 6
Poisson's ratio.

I_DMDVT_FU : = 7
Limit strength for tension.

I_DMDVT_LX : = 8
Thermal expansion ratio.

I_DMDVT_RO : = 9
Force density (unit weight)

I_DMDVT_RT : = 10
Tension strength

I_DMDVT_E_5 : = 11
5% module of axial elasticity (timber)

I_DMDVT_TRANS : = 12
Cross-sectional elasticity module (timber)

I_DMDVT_PN_E_TRANS : = 13
Cross-sectional elasticity module (timber)(Polish standards)

I_DMDVT_PN_E_ADITIONAL : = 14
Additional Young's module (Polish Standards)

I_DMDVT_RE_BENDING : = 15
Strength for bending

I_DMDVT_RE_AX_TENS : = 16
Strength for axial tension (timber).

I_DMDVT_RE_TR_TENS : = 17
Strength for transverse axial tension.

I_DMDVT_RE_TR_COMPR : = 18
Strength for transverse compression.

I_DMDVT_RE_SHEAR : = 19
Strength for shear (timber).

I_DMDVT_DAMPCOEF : = 20
Damping coefficient.

IRDimMatDefLongExValType

Definition of the set of identifiers defining the type of the parameter of the member material.

Attributes:

I_DMDLEVT_TIMB_TYPE : = 1
Defines the type of timber material (normal -- 0, glue-laminated) -- 1)

I_DMDLEVT_CATEGORY : = 2
Defines the additional characteristic for timber code -- 3 categories (I,II,III).

I_DMDLEVT_NATURE : = 3
Defines the additional characteristic for timber code
2 natures (resinous / non resinous)

IRDimMatDefDblExValType

Definition of the set of identifiers defining the type of additional parameters of member material.

Attributes:

I_DMDDEVT_RETRAIT : = 4
Additional parameter for timber code – shrinkage in %

I_DMDDEVT_HUMIDITY : = 5
Additional parameter for timber code - Humidity in %.

IRDimMatDef

- **Implementation in RDIM module**
Interface for member's material data support.

Attributes:

Type : IRDimMatDefType
The attribute defines the type of material.

<GET>Name : string
Returns the name of material.

<GET>SecondName : string
Returns the additional brief description of the given material.

Functions:

SetNames (_name : string, second_name : string)
The method saves the name of the material.

WriteValue (_type : IRDimMatDefValType, val : double)
It saves the value of the numerical parameter of the given type.

ReadValue (_type : IRDimMatDefValType) : double
It gives the value of the numeric parameter of the given type.

WriteLongExtraValue (_type : IRDimMatDefLongExValType, val : long)
The method saves the additional characteristics of the material.

ReadLongExtraValue (_type : IRDimMatDefLongExValType) : long
Returns the additional characteristics of the material.

WriteDoubleExtraValue (_type : IRDimMatDefDblExValType, val : double)
Saves the additional numeric parameters of the given type.

ReadDoubleExtraValue (_type : IRDimMatDefDblExValType) : double
Returns the values of the additional numerical parameters of the given type.

IRDimProfDefType

Definition of the set of identifiers defining the set of profiles type analyzed by the RDIM module.

Attributes:

The last part of identifier's name defines the type of the profile.

I_DPDT_NONE : = 0
I_DPDT_CAE : = 1
I_DPDT_CAEP : = 2
I_DPDT_CAI : = 3
I_DPDT_CAIP : = 4
I_DPDT_DCEC : = 5
I_DPDT_DCED : = 6
I_DPDT_DCEP : = 7
I_DPDT_DCIG : = 8
I_DPDT_DCIP : = 9
I_DPDT_HEA : = 10
I_DPDT_HEAA : = 11
I_DPDT_HEB : = 12
I_DPDT_HEC : = 13
I_DPDT_HEM : = 14
I_DPDT_HER : = 15
I_DPDT_HHEA : = 16
I_DPDT_HHEB : = 17
I_DPDT_HHEM : = 18
I_DPDT_IIPE : = 19
I_DPDT_IPE : = 20
I_DPDT_IPEA : = 21
I_DPDT_IPEO : = 22
I_DPDT_IPER : = 23
I_DPDT_IPEV : = 24
I_DPDT_IPN : = 25
I_DPDT_MHEA : = 26
I_DPDT_MHEB : = 27
I_DPDT_MHEM : = 28
I_DPDT_MIPE : = 29
I_DPDT_PRS : = 30
I_DPDT_TCAR : = 31
I_DPDT_TEAE : = 32
I_DPDT_TEAI : = 33
I_DPDT_THEX : = 34
I_DPDT_TREC : = 35
I_DPDT_TRON : = 36
I_DPDT_UAP : = 37
I_DPDT_UPN : = 38
I_DPDT_UUAP : = 39
I_DPDT_UUPN : = 40

I_DPDT_OSIE : = 41
I_DPDT_CAISSON : = 42
I_DPDT_RECT : = 43
I_DPDT_DRECT : = 44
I_DPDT_TUBE : = 45
I_DPDT_ISYM : = 46
I_DPDT_INSYM : = 47
I_DPDT_TUSER : = 48
I_DPDT_CUSER : = 49
I_DPDT_FRTG : = 50
I_DPDT_CROSS : = 51
I_DPDT_UPAF : = 52
I_DPDT_RECS : = 53
I_DPDT_CIRC : = 54
I_DPDT_X : = 55

IRDimProfDefItemType

Definition of the set of identifiers defining the type of data for the given type of profile analyzed by the RDIM module.

Attributes:

I_DPDIT_STANDARD : = 1
Defines data describing the profile in any member point (constant inertia profile).

I_DPDIT_VAR_IN_POINT : = 2
Defines the data describing the profile in the member point in which calculations are currently carried out (variable inertia profile)

I_DPDIT_VAR_MIDDLE : = 3
Defines the data describing the profile in the middle of the member (variable inertia profile).

I_DPDIT_VAR_BEGEND : = 4
Defines the data describing the profile at member's origin and end (variable inertia profiles).

IRDimProfDefValType

Definition of the set of identifiers defining the numerical parameter type of the given profile.

Attributes:

I_DPDVT_H : = 1
Cross-section height.

I_DPDVT_HW : = 2
Cross-section web height (without fillet)

I_DPDVT_HD : = 3
The distance between internal flange edges

I_DPDVT_B : = 4
The width of the cross-section upper flange.

I_DPDVT_B2 : = 5
The width of the cross-section lower flange.

I_DPDVT_BF : = 6
The width of the cross-section upper flange for class section calculations(without fillet)

I_DPDVT_BF2 : = 7
The width of the cross-section lower flange for class section calculations (without fillet).

I_DPDVT_BD : = 8
The distance between internal web edges (for example square tube)

I_DPDVT_EA : = 9
Web thickness.

I_DPDVT_ES : = 10
flange thickness.

I_DPDVT_EA2 : = 11
Web thickness.

I_DPDVT_ES2 : = 12
Lower flange thickness for INSYM.

I_DPDVT_DIS : = 13
The distance between the double rectangle section elements.

I_DPDVT_R : = 14
Fillet radius.

I_DPDVT_R2 : = 15
Fillet radius.

I_DPDVT_VZ : = 16
The distance between the extreme fibers (+) along the axis Z

I_DPDVT_VPZ : = 17
The distance of the extreme fibers (-) along the Z axis.

I_DPDVT_VY : = 18
The distance of the extreme fibers (+) along the Y axis.

I_DPDVT_VPY : = 19
The distance of the extreme fibers (-) along the Y axis.

I_DPDVT_I : = 20
Moment of inertia (torsion)

I_DPDVT_IY : = 21
Moment of inertia around the Y axis.

I_DPDVT_IZ : = 22
Moment of inertia around the Z axis.

I_DPDVT_S : = 23
Cross- section area .

I_DPDVT_SY : = 24
Reduced section area for the XY - calculations of the medium shear stress

I_DPDVT_SZ : = 25
Reduced section area XZ - calculations of the medium shear stress

I_DPDVT_MSY : = 26
Plastic Section modulus (axis Y)

I_DPDVT_MSZ : = 27
Plastic Section modulus (axis Z)

I_DPDVT_MASSE : = 28
Nominal weight per length unit.

IRDimProfDef

- **Implementation in the RDIM module**
  Interface to support the data concerning the profile of the member.

Attributes:

Type : IRDimProfDefType
The attribute defines the type of the profile.

Name : string
The attribute defines the full name of the profile.

<GET>IsVar : bool

If the TRUE value is assigned to this attribute, it’s the variable inertia profile.

Functions

Clear ()

This method removes all the data.

WriteValue (item : IRDimProfDefItemType, _type : IRDimProfDefValType, val : double)

The method saves particular values of the profile parameter. If it is a variable inertia profile it's particular parameters are considered as the values in the calculation point, at the member origin, middle or end (depending on the variable value “item”).
For the profile with constant inertia, the “item” parameter of the discussed function has always the I_DPDIT_STANDARD value.

ReadValue (item : IRDimProfDefItemType, _type : IRDimProfDefValType) : double

The method reads particular values of the profile parameters previously written by means of the “WriteValue” function. If it is a variable inertia profile, it’s particular parameters are considered as the values in the calculation point, at the member’s origin, middle and end (depending on the value of the “item”variable). For the constant inertia profile, the " item" parameter of the discussed function has always the I_DPDIT_STANDARD value.

IRDimCalcStateFlagType

Definition of the set of identifiers defining the information flags for the IRDimCalcState interface.

Attributes:

I_DCSFT_CHECKSLEND : = 1

The attribute forces the information whether the maximum slender calculation option has been selected in the calculations configuration window.

I_DCSFT_POINTMIDDLE : = 2

The attribute forces the information whether the current calculation point is in the middle of the member or on one of it’s ends.

I_DCSFT_FIRE : = 3

The attribute forces the information whether the calculations are carried out taking into account the possibility of fire impact.

IRDimCalcStateCalcType

Definition of the set of identifiers defining the type of calculations which are carried out for the IRDimCalcState interface.

Attributes:

I_DCSPV_MANUAL_VERIF : = 1
The attribute informs that the calculations are carried out in the manual mode.

I_DCSPV_MEMBERS_VERIF : = 2
The attribute informs that the current calculations are part of the members verification.

I_DCSPV_GROUP_VERIF : = 3
The attribute informs that the current calculations are part of the groups verification.

I_DCSPV_DIMENSIONING : = 4
The attribute informs that the current calculations are part of members dimensioning in groups.

I_DCSPV_OPTIMIZATION : = 5
The attribute informs that the current calculations are part of the members optimization in the groups.

IRDimCalcStateValueType

Definition of the set of identifiers defining the type of numerical parameters transferred to the MVCS module during the calculation by means of the IRDimCalcState interface, defined by the calculation configuration window.

Attributes:

I_DCSPV_MAXSLEND : = 1
Defines the maximum slenderness (the parameter is set by the user in the calculation configuration window).

I_DCSPV_EFFRATIO : = 2
Defines the efficiency ratio limit value (the parameter is set by the user in the calculation configuration window).

IRDimCalcState

- **Implementation in the RDIM module**

The auxiliary interface to support the data defining general parameters of the performed calculations. It includes the data about the calculation type and the general parameters which are to be taken into account during partial member calculations in any point.

Attributes:

CalcType () : IRDimCalcStateCalcType
Defines the type of the calculations.

Functions:

SetFlag (_type : IRDimCalcStateFlagType, val : bool)
Sets the selected parameter (flag) on the appropriate value (TRUE or FALSE).

IsFlagSet (_type : IRDimCalcStateFlagType) : bool
Verifies what logical value is the selected parameter (flag) set on.

SetParamValue (_type : IRDimCalcStateValueType, val : double)
Sets the value of the selected numerical parameter.

GetParamValue (_type : IRDimCalcStateValueType) : double
Returns the value of the numerical parameter previously assigned by means of the “SetParamValue” function.

WriteClientData (_val : IRDimStream)
Saves the unique, arbitrary data of the MVCS module which are used for calculations by the module.

ReadClientData (_val : IRDimStream)
Reads the unique, arbitrary data of the MVCS module previously saved by the WriteClientData function which are used for calculations by this module.

IRDimMembSrv

- **Implementation in RDIM module**

The interface intended for complex support of member parameters using data supported by the IRDimMembDef interface.

Functions:

CheckLabelName (_name : string, is_defined : bool*, can_be_saved : bool*)
The method verifies whether the label with the given name already exista in the Robot system (if it’s defined in the system) and whether it is possible to update the code characteristic of the member by the “Save” method with such a label. If such label already exists and if

there are specific code characteristic assigned to it (which means that the member “Type”attribute is defined) the variable “can_be_saved”be set on the TRUE value only when the attribute will have the I_DMDT_USER value. Otherwise the MVCS module should display an appropriate message. If the “Type” attribute has the I_DMDT_USER value and the label already exists in the system, before the saving by the “Save” method the MVCS module should display a message asking for permission to overwrite the existing data.

Save (val : IRDimMembDef)
Saves the member definition with the label name in accordance with the “Name” attribute in the IRDimMembDef. Interface

EditAdjoinParams (in : IDispatch*)
Starts the mechanism of adjoining bars parameters edition for bar definition (IRDimMembDefData interface).

IRDimMembResCalcType

Definition of the identifiers defining the type of calculations.

Attributes:

I_DMRCT_ULTIMATE : = 0
Defines the ULS calculations

I_DMRCT_SERVICE : = 1
Defines the SLS calculations.

IRDimMembResUltimateValueType

Definition of the set of identifiers defining the basic ULS calculations results parameters.

Attributes:

I_DMRVT_SLEND_Y : = 0
Defines the slenderness in respect of the buckling in Y axis plane.

I_DMRVT_SLEND_Z : = 1
Defines the slenderness in respect of the Z axis.

I_DMRVT_RATIO : = 2
Defines the efficiency ratio value.

I_DMRVT_EFFRATIO : = 3
Defines the efficiency ratio limit value.

IRDimMembResServiceType

Definition of the set of identifiers defining the basic SLS results parameters.

Attributes:

I_DMRST_DEFL_Y : = 0
Defines deflections in Y axis direction.

I_DMRST_DEFL_Z : = 1
Defines deflections in respect of the Z axis.

I_DMRST_DISP_X : = 2
Defines the displacement in respect of the X axis.

I_DMRST_DISP_Y : = 3
Defines the displacement in respect of the Y axis.

IRDimMembResWndType

Definition of the set of identifiers defining the type of the window including the calculation results generated in the MVCS module on the RDIM module demand.

Attributes:

I_DMRWT_ULTIMATE_WND : = 0
Indicates the dialog box for ULS calculation results.

I_DMRWT_SERVICE_WND : = 1
Indicates the dialog box for SLS calculation results.

I_DMRWT_DETAILED_WND: = 2
Indicates the dialog box of detailed analysis.

IRDimMembResTableLineType

Definition of the set of identifiers defining the type of the line having it's own character and look which is displayed in the detailed results table of the RDIM module.

Attributes:

I_DMRTLT_PRINTCAPTION : = 0
The template visible only on the printout of the detailed results table.

I_DMRTLT_CAPTION : = 1
Title

I_DMRTLT_SUBCAPTION : = 2
Subtitle

I_DMRTLT_1PARAM : = 3
Straight line without columns.

I_DMRTLT_4HEADER : = 4
Four parameters (four column) template of the whole table.
I_DMRTLT_5HEADER : = 5
Five parameters (five columns) template of the whole table.

I_DMRTLT_4PARAM : = 6
Straight line with 4 columns.

I_DMRTLT_5PARAM : = 7
Straight line with 5 columns.

IRDimMembResTableComp

Definition of the set of identifiers defining the field type of the line displayed in the detailed results table of the RDIMmodule. The straight line with 4 divisions doesn't have the I_DMRTC_PARAGRAPH attribute.

Attributes:

I_DMRTC_NAME : = 1
The area (columns) including the name of the calculations results.

I_DMRTC_VALUE : = 2
The area (columns) including the value of the calculations results.

I_DMRTC_UNIT : = 3
The area (columns) including the unit in which the calculations results are displayed.

I_DMRTC_DESCRIPTION : = 4
The area (columns) including the description of the displayed calculations result.

I_DMRTC_PARAGRAPH : = 5
The number of the code paragraph which has been applied to the final calculations.

IRDimMembResTableDefType

Definition of the set of identifiers defining the type of the block (interval) of the parameters displayed in the detailed results table of the RDIM module.

Attributes:

I_DMRTDT_DEFSECT_PROFILE : = 1
Defines the block including the profile parameters.

I_DMRTDT_DEFSECT_MATERIAL : = 2
Defines the block including the material parameters.

IRDimMembResTableDefProf

Definition of the set of identifiers defining particular parameters displayed in the block (section) concerning the profile, displayed in the detailed results table of RDIM module.

Attributes:

I_DMRTDP_LINE_PROF_PRINTCAPTION : = 0
I_DMRTDP_LINE_PROF_CAPTION : = 1
Display of headings.

I_DMRTDP_LINE_PROF_S : = 2
Displays cross-section area

I_DMRTDP_LINE_PROF_SY : = 3
Displays reduced area of section for XY.

I_DMRTDP_LINE_PROF_ SZ: = 4
Displays reduced area of section for XZ.

I_DMRTDP_LINE_PROF_I : = 5
Displays moment of inertia.

I_DMRTDP_LINE_PROF_IY : = 6
Displays moment of inertia about Y axis.

I_DMRTDP_LINE_PROF_IZ : = 7
Displays moment of inertia about Z axis.

I_DMRTDP_LINE_PROF_WEL_Y : = 8
Displays elastic section modulus about Y axis.

I_DMRTDP_LINE_PROF_WEL_Z : = 9
Displays elastic section modulus about Z axis.

I_DMRTDP_LINE_PROF_H : = 10
Displays section height.

I_DMRTDP_LINE_PROF_B : = 11
Displays top flange width.

I_DMRTDP_LINE_PROF_B2 : = 12
Displays bottom flange width.

I_DMRTDP_LINE_PROF_ES : = 13
Displays flange thickness.

I_DMRTDP_LINE_PROF_ES2 : = 14
Displays bottom flange thickness.

I_DMRTDP_LINE_PROF_EA : = 15
Displays web thickness.

I_DMRTDP_LINE_PROF_RY : = 16
Displays fillet radius with respect to Y axis.

I_DMRTDP_LINE_PROF_RZ : = 17
Displays fillet radius with respect to Z axis.

I_DMRTDP_LINE_PROF_MASSE : = 18
Displays nominal weight per length unit.

IRDimMembResTableDefMater

Definition of the set of identifiers defining particular parameters displayed in the block (section) concerning the material and displayed in the detailed results table of the RDIM module.

Attributes:

I_DMRTDM_LINE_MATER_CAPTION : = 1
I_DMRTDM_LINE_MATER_NAME : = 2
Display of headings.

I_DMRTDM_LINE_MATER_RE : = 3
Material resistance.

IRDimMembRes

- **Implementation in MVCS module**
  The interface supports calculation results of the member.

Attributes:

<PUT>Language : long
Defines the number of the language in which results are displayed.

<PUT>Units : IRDimUnits
Sets the interface for units support.

<GET>RtfFileName : string
Returns the name of the RTF-type file which will be used as a template for calculation note generation.

<GET>IsClientLimitStateService () : bool
Returns the information whether in the customer's module (MVCS) SLS calculations are implemented. If they are, the method returns the TRUE value. Otherwise the RDIM module will use for such calculations a default implementation included in it.

<GET>CodeResults () : IDispatch *
Returns the interface allowing to get numerical values of all important code parameters used in the algorithm of member verification according to a given code.

Functions:

RefreshUnits (areROBOTUnits : bool)
Commands interface update for units support (Refresh () method). Method's parameter informs which set of units is currently valid (code units or the so-called Robot units that is units set in the system configuration module).

ResOfCalc (type : IRDimMembResCalcType) : IRDimMembCalcRetValue
Returns brief information about member calculation results in the current point.

GetUltimateValue (type : IRDimMembResUltimateValueType) : double
Returns the set of values for ULS calculation parameters which are displayed in the curtailed result window.

IsServiceActive (type : IRDimMembResServiceType) : bool
Returns information about the type of SLS calculations which have to be carried out. This information is included in the code definition of the member. This method is important only when SLS analysis is implemented in customer's module (MVCS).

GetServiceRatio (type : IRDimMembResServiceType) : double
Returns the value of the calculated "RATIO"for the specified SLS calculation type. This method is important only when SLS analysis is implemented in customer's module (MVCS).

GetServiceCaseName (type : IRDimMembResServiceType) : string
Returns the name of the result load case for the particular type of SLS calculations. This method is important (is to be taken in account) only when the SLS analysis is implemented in customer's module (MVCS).

Retrieve (str : IRDimStream)
Reads the data of the analysis results from the IRDimStream.

SetMembDef (memb_def : IRDimMembDef)
Sets the interface to support code parameters data of the member for which calculations have been carried out.

SetMatDef (mat_def : IRDimMatDef)
Sets the interface to support data concerning the material used in the analysis

SetProfDef (prof_def : IRDimProfDef)
Sets the interface to support data concerning the profile used in the analysis.

SetEffDef (eff_def : IRDimEffDef)
Sets the interface to support data concerning internal forces and additional moments for the member for which the analysis has been carried out.

GetMembDefAccess () : IRDimMembDef
Provides the data concerning code parameters of the member for which calculations have been carried out.

GetMatDefAccess () : IRDimMatDef
Provides the data concerning the parameters of the material used for member calculation.

GetProfDefAccess () : IRDimProfDef
Provides data concerning the parameters of the profile used for member calculation.
GetEffDefAccess () : IRDimEffDef
Provides data concerning the internal forces and additional moments for the member for which the calculations have been carried out.

CreateResWnd (type : IRDimMembResWndType, parent_hwnd : long) : long
Creates a window displaying the calculation results which will be used as a tab in the RDIM module results window.

GetDefMaxLineNo (type : IRDimMembResTableDefType, ret : long*, was_proc : bool*)
Returns the number of all the lines from the profile/material parameters block which can be displayed in the detailed results table of the RDIM module (depending on *type* parameter). If the MVCS module doesn't change the number of lines from the profile / material parameters block, it sets the "was_proc" parameter on FALSE. Otherwise a new number of lines to be displayed in the above mentioned blocks should be given in the "ret"parameter.

IsDefLineActive (type : IRDimMembResTableDefType, line_no : long, ret : bool*, was_proc : bool*)

Informs whether the given line from the profile/material parameters block (in the detailed results table of the RDIM module) should be displayed or not (lines are numbered starting from 1). If the MVCS module doesn't change the standard lines disposition in the profile / material parameters block, it sets the "was_proc" parameter on FALSE. Otherwise the information whether the given line should displayed or not should be given in ret parameter.

GetDefLineComponent (type : IRDimMembResTableDefType, line_no : long, cmpnt_no : IRDimMembResTableComp, defline_no : long*,ret : string, defline_no_proc: bool*)

Returns the content of the field (component, column) from the block line containing profile / material parameters in the detailed results table of the RDIM module. (Lines are numbered starting from 1).

If the MVCS module doesn't modify the standard lines disposition in the profile / material parameters block, it sets the "defline_no_proc" parameter on TRUE and the "defline_no" parameter on "line_no" value. If the last parameter is set on a value different than "line_no" (but from the domain by the list IRDimMembResTableDefProf lub IRDimMembResTableDefMat), the component (defined by the "cmpnt_no" parameter) of the line number "line_no"will be replaced by the analogical component corresponding to the "defline_no"line (formatted in RDIM module). If the variable "defline_no_proc"is set on FALSE, the "defline_no"parameter is not taken into account and the RDIM module as a component identified by the "cmpnt_no" parameter (column) in the line number "line_no_ will display the "ret" variable.

GetMaxLineNo () : long

Returns the number of all the lines to be displayed (in the detailed results table of the RDIM module results window).

IsLineActive (line_no : long) : bool

Informs whether the given line should be displayed (lines are numbered starting from 1)

GetLineType (line_no : long) : IRDimMembResTableLineType

Returns the type of the line (lines are numbered starting from 1)

GetLineComponent (line_no : long, cmpnt_no : IRDimMembResTableComp) : string

Returns the content of the field (column) of the line with the given type (lines are numbered starting from 1)

BlockCount (_name : string , ret : long* , was_proc : bool*)

The method supports the calculation note generated on the basis of a template in RTF format .

This function returns the number of repetitions of the given block (template's fragment) with a name situated in the variable "_name" generated on the basis of this block.
The template in RTF file can look like this:

```
@BEGIN(Loop)@
 ............................................
 ...................................................
@END(Loop)@
```

Code in C++ language supporting this template has the following form:

```
{*ret=1;
  *was_proc = FALSE;
  if (_name == "Loop")
       {
        *was_proc = FALSE;
         *ret = 3;
        }
}
```

The block named "Loop" will be processed three times.
For a multiple processing of the same template, by default the RDIM module supports the name of the "MemberNo" block which allows to print information about all the analyzed structure members in one note. The template included in the RTF file should be situated in the block enclosed in this variable.
Actually the MVCS module shouldn't use this method and it's implementation should only set the variable was_proc on FALSE while to the "ret" variable it should attribute the value 1.

IsStatement (_name : string , ret : bool* , was_proc : bool*)

The method serves the calculation note generated on the basis of the template in the RTF format.
This function is considered as a logical condition. For example, if the variable "_name" sent following template's content :

```
@IF(Buckling=ON)@
...................................................
............................................
@END()@
```

contains the inscription “Buckling=ON”, basing on the analysis results, the MVCS module (if it supports this condition, it defines the "was_proc" variable as TRUE) defines the value of the variable "ret" as TRUE when the block of the template included in this condition is to be processed. Otherwise this variable has to be defined as FALSE. If the MVCS module doesn't serve the described condition defined by parameter's content "_name", it defines the variable "was_proc"as FALSE. This enables the use of the default mode of (developing) (if such a condition is implemented in the RDIM module). Here is a brief example of implementation in the C++ language.

```
{*was_proc = FALSE;
  int i = Name.Find('=');
if(i >= 0)
  {
  CString VarName = Name.Left(i);
   if(VarName == "Buckling")
      {
      *was_proc = TRUE;
      CString Value = Name.Mid (i + 1);
      CString Val = ( IsBuckling()  ? "ON" : "OFF" );
      if( Value == Val )
          *ret =  TRUE;
      }
  }
else  *ret = FALSE; }
```

ReplaceMark (mark : string, mark_out : string* , was_proc : bool *)

The method supports the calculation note generated on the basis of the template in RTF format.

The function supports the variables included in the template. The "mark" parameter contains the name of such a variable. If the MVCS module supports the given variable, it assigns the TRUE value to the variable "was_proc" and enters to the "mark_out" variable a text. This text should replace the display of the inscription included in the "mark" variable in the template.

If the MVCS module doesn't support the variable it sets the "was_proc" parameter on FALSE. As a result, a default way of replacement will be used for such a variable (if the support is implemented in the RDIM module).

The template in the RTF file looks as follows:

@Mat@

Here is a brief example of implementation in the C++ language.

```
{ *was_proc = FALSE;
  if(mark == "Mat")
     {
     *mark_out = “Material name:”;
      *was_proc = TRUE;
     }
}
```

RecognizedPQ (_name : string, vstr : string*, ustr : string*, was_proc : bool*)

The method supports the calculation note generated on the basis of the template in the RTF format.

This function supports the variable values included in the template. The "_name" parameter contains the name of such a variable. If the MVCS module supports the given variable, it assigns to the "was_proc"variable the TRUE value and enters into the "mark_out" variable a text. This text should replace in the template the display of the inscription included in the "mark" variable. If the MVCS module doesn't support the given variable it assigns the FALSE value to the

"was_proc" parameter. As a result a default way of substitution (replacement) is used for such a variable (if the support is implemented in the RDIM module).
The template in the RTF file looks as follows:

```
@VAL(MatVal)@
```

Here is a brief example of implementation in the C++ language.

```
{ *was_proc = FALSE;
  *vstr = “”;
  *ustr = “”;
  if(mark == "MatVal")
     {
     *vstr = “Steel”;
      *was_proc = TRUE;
     }
}
```

If numeric values are processed, it is possible to introduce an inscription in the "ustr" parameter with the unit abbreviation. Information about the unit can of course be introduced in the variable "vstr" together with the numerical value.

IRDimDeflDispType

Definition of the set of identifiers defining the type of displacements and deflections related to a single member for the IRDimDeflDisp interface.

Attributes:

I_DDDT_DEFL_Y : = 0
Deflection in respect of the axis Y

I_DDDT_DEFL_Z : = 1.
Deflection in respect of the axis Z.

I_DDDT_DEFL_MAX_Y : = 2
Maximum deflection for the member in respect of the axis Y.

I_DDDT_DEFL_MAX_Z : = 3
Maximum deflection for the member (member) in respect of the Z axis.

I_DDDT_DISP_X : = 4
Displacement in respect of X axis.

I_DDDT_DISP_Y: = 5
Displacement in respect of Y axis.

IRDimDeflDispInitType

The definition of identifiers defining the member and the load case. The set of identifiers has been defined for the IRDimDeflDisp interface.

Attributes:

I_DDDIT_MEMBER_NO : =1
Defines member's internal number.

I_DDDIT_CASE_NO : = 2
Defines the internal number of the load case.

IRDimDeflDispCaseType

Definition if the set of identifiers characterizing the load case type for the IRDimDeflDisp interface.

Attributes:

I_DDDCT_NONE : = 0
Defines an unidentified load case.

I_DDDCT_SIMPLE : = 1
Defines a simple load case.

I_DDDCT_COMBINATION : = 2
Defines the combination.

I_DDDCT_CODECOMBINATION : = 3
Defines the code combination

IRDimDeflDispSimpleCaseNature

Definition of the set of identifiers defining the load nature (character). The set of identifiers has been defined for the IRDimDeflDisp interface.

Attributes:

I_DDDSCN_PERM : = 0
Defines the dead load type.

I_DDDSCN_EXPL_Z : = 1
Defines live load type.

I_DDDSCN_TEMP : = 2
Defines thermic load type.

I_DDDSCN_WIND : = 3
Defines wind load type.

I_DDDSCN_SNOW : = 4
Defines snow load type.

I_DDDSCN_ACCI: = 5
Defines accidental load type.

I_DDDSCN_SEIS : = 6
Defines seismic load type.

I_DDDSCN_NON: = 7
Defines unidentified load type.

IRDimDeflDisp

- **Implementation in RDIM module**

  Interface assigned for SLS calculations

  Generally, the SLS calculations as not depending on the code, are totally carried out by the RDIM module. Nevertheless, some of the codes introduce a specific approach of such calculations. That's why it's necessary to provide to the customer's MVCS module an interface which allows to implement these calculations according to the standard recommendation.

Attributes:

<GET>MemberNo : long
Defines the internal number of the member for which the MVCS module has to carry out the Serviceability (Limit State) calculations.

<GET>CaseNo : long
Defines the internal number of the load case for which Serviceability (Limit State) calculations have to be carried out by the MVCS module.

<PUT>PointsNum : long
Sets the number of calculation points on the member. If the method is not used (which is recommended), a default number of calculation points will be assigned (= 11).

<PUT>CurPointNo : long
Sets the number of the calculation point necessary for the MVCS module to carry out the SLS calculations.

<PUT>CurCaseNo : long
Sets the internal number of the load case necessary for the MVCS module to carry out the SLS calculations.

<PUT>CurCompIndex : long
Sets the number of the code combination case component necessary to the MVCS module to carry out the SLS calculations.

<PUT>CantileverMode : bool

Sets the mode in which the member is considered as a cantilever.

Functions:

Init (type : IRDimDeflDispInitType , val : long)

Sets member's and load case internal number for which the MVCS module has to carry out SLS calculations. The method is called by the RDIM module after it has created the discussed interface.

CaseUserNo (case_no : long) : long

Returns the number of the given load case user ("case_no" parameter is the internal number of this case).

CaseType (case_no : long) : IRDimDeflDispCaseType

Returns the type of the given load case ("case_no" parameter is the internal number if this case)

CaseName (case_no : long , comp_index : long , with_def : bool) : string

Returns the name of the given load case ("case_no" parameter is the internal number of this case). In case of the code combination the "comp_index" parameter is the number of it's component. If the parameter "with_def" is set on TRUE, the definition substitution is added to the name (for the combination and the code combination code combination case component).

CaseNature (case_no : long) : IRDimDeflDispSimpleCaseNature

Returns the character of the given load case ("case_no" parameter is the internal number of this case).

CaseClass (case_no : long) : long

Returns the class of the given Load case ("case_no" parameter is the internal number of this case).

CodeCombCompNum (case_no : long) : long

Returns the number of components that is combinations for the code combination cases ("case_no" parameter is the internal number of this case).

CombCompNum (case_no : long , comp_index : long) : long

Returns the number of simple cases of which the combination or the component of the code combination code combination case is made with a following number defined by the "comp_index" parameter (The "case_no" parameter is the internal number of the combination or the component of the code combination case).

Value (type : IRDimDeflDispType) : double

Returns the numerical value for the given displacement or deflection case. This value is given taking into account the attributes:

“PointsNum”, “CurPointNo”, “CurCaseNo”, “CurCompIndex” and “CantileverMode”.

CombCompParam (index : long , case_no : long* , cv : double*)

Returns the "index" defined by a parameter, the internal number of a simple case of which a combination or a code combination case component have been created and also the combination coefficient for this simple case. This value is returned taking in account the attributes: “CurCaseNo” and “CurCompIndex”. "Index" parameter is the following component number returned by the “CombCompNum” method.

IRDimMembCalcRetValue

Definition of the set of identifiers characterizing calculation results. The attributes have been defined for the IRDimMembCalc interface.

Attributes:

I_DMCRV_INCORRECTDATA : = -2

Defines the data for which the MVCS module cannot carry out calculations.

I_DMCRV_INSTABILITY : = -3

Defines the instability.

I_DMCRV_INCORRECT : = 0

Means that the results of the current calculations carried out for the member are negative. The profile doesn't meet the requirements.

I_DMCRV_CORRECT : = 1

Means that current calculations carried out for the member were successful. The profile meets the requirements.

IRDimMembCalcBuckType

Definition of identifiers defining the buckling direction. These identifiers have been defined for the IRDimMemCalc interface.

Attributes:

I_DMCBC_Y : = 1

Defines the direction in respect of Y axis.

I_DMCBC_Z : = 2

Defines the direction in respect of the Z axis.

IRDimMembCalcFinishType

Definition of identifiers defining the moment when particular calculations are finished.

Attributes:

I_DMCFT_ULTIMATE : = 1
Defines the moment when the ULS calculations are finished.

I_DMCFT_SERVICE : = 2
Defines the moment when the SLS calculations are finished.

I_DMCFT_FINISH_ALL: = 3
Defines the moment when all the calculations are finished.

IRDimMembCalcMessageType

Definition of the identifiers defining the type of information which should be displayed in the appropriate tab control in the short results window.

Attributes:

I_DMCMT_MESSAGE : = 1
Defines the information about any untypical incidents which occurred during the calculation.

I_DMCMT_ERROR : = 2
Defines the information about errors which occurred during the calculation.

IRDimMembCalc

- **Implementation in MVCS module**
  The basic interface defined to support the ULS and SLS calculations for the member. It allows to carry out code calculations out of the RDIM module and to transfer them to the customer's MVCS module. By this interface partial calculations for the given member are carried out, and all the data necessary to these calculations are managed by the RDIM module. That is why the customer's module can implement only the code taking in account national standards and doesn't have neither to determine data for a given calculation point nor how to carry out following iterations for the analyzed structure.

Attributes:

<GET>IsExtraMomentsSet1 : bool

Returns information whether the calculation code of the customer's MVCS module uses for calculation the values of parameters in respect of Y or Z axis from the following set of moments:

- M1 -- moment at the member's origin
- M2 -- moment at the member's end
- M12 -- relation of the moments at the member's origin and end

If the MVCS module does not use any of these values, the method should return the FALSE value. It would cause the RDIM module not to support the above mentioned set of data (and they won't be accessible by the interface) which would considerably accelerate the calculations.

<GET>IsExtraMomentsSet2 : bool

Returns the information whether the calculation code of the customer's MVCS module uses for calculations the values of the parameters in regard of the Y or Z axis from the following set of moments:

- MP_MAX -- maximum positive value of the moment for the given member
- MN_MAX -- minimum positive value of the moment for the given member
- M_MID -- moment in the middle of the member
- M_1P4L -- moment in the 1/4 of the member's length
- M_3P4L -- moment in the 3/4 of the member's length

If the MVCS module doesn't use any of these values the method should return the FALSE value. As a result the RDIM module will not support the above mentioned set of data (and they won't be available by the interface) which would considerably accelerate the calculations.

<GET>IsIntPointsMomentsSet1 : bool

Returns the information whether the calculation code of the customer's MVCS module uses the values of parameters in respect of the Y or Z axis for the calculations taking into account intermediate points on the member (brace).

- M1 -- moment at the interval origin (between intermediate points)
- M2 -- moment at the end of the interval (between intermediate points)
- M12 -- relation of moments at the interval's origin and end (between intermediate points)

If the MVCS module doesn't use any of these values, the method should return the FALSE value. As a result the RDIM module won't support the above mentioned set of data (and they won't be available by the interface) which would considerably accelerate the calculations.

<GET>IsIntPointsMomentsSet2 : bool

Returns the information whether the calculation code of the customer's MVCS module uses for calculations taking into account intermediate points on the member (brace)
the values of parameters in respect of the Y or Z axis from the following set of moments:

- MP_MAX -- maximum positive value of the moment for the given interval (between intermediate points)
- MN_MAX -- minimum positive value of the moment for the given interval (between intermediate points)
- M_MID -- moment in the middle of the interval (between intermediate points)
- M_1P4L -- moment in the 1/4 of the member's length (between intermediate points)
- M_3P4L -- moment in the 3/4 of the member's length (between intermediate points)

If the MVCS module doesn't use any of these values the method should return the FALSE value. As a result, the RDIM module won't support the above mentioned set of data (and they won't be available by the interface) which would considerably accelerate the calculations.

<GET>IsClientLimitStateService : bool

Returns the information whether the SLS calculations are implemented in the MVCS customer's module. If that is the case the method returns the TRUE value. Otherwise, the RDIM module will use a default implementation for such calculations.

<GET>MessagesNum : long

Returns the number of messages which should be displayed by the RDIM module in the appropriate tab control in the short results window.

Functions:

SetMembDef (memb_def : IRDimMembDef)

Sets the interface to support the code data of the currently analyzed member.

SetMatDef (mat_def : IRDimMatDef)

Sets the interface to support the material data for the currently analyzed member.

SetProfDef (prof_def : IRDimProfDef)

Sets the interface to support the data characterizing the profile of the currently analyzed member.

SetEffDef (eff_data : IRDimEffDef)

Sets the interface to support the data about internal forces of the currently analyzed member.

SetCalcState (st : IRDimCalcState)
Sets the interface to support the data characterizing the general parameters of the calculations carried out. This interface includes the data about their type and about general parameters which have to be taken into account during partial calculations of the member in any given point.

CalculBuckling ()
The method is called by the RDIM module if the buckling coefficient update is necessary (to call this method, the "IsBuckCoefConst" method of the IRDimMembDef interface should return the FALSE value).

CalculMember () : IRDimMembCalcRetValue
The method called by the RDIM module and forcing the ULS calculation.

GetRatio () : double
Returns the Ratio (ULS) value after the calculations carried out by the “CalculMember” method.

SetAutoInitDeflections (defl_y : double, defl_z : double)
The method called by the RDIM module. If, for the given member, the values of initial cambers from a given load case have to be automatically calculated and if the user activates (starts) the option of taking this cambers into account, the RDIM module would transfer the calculated values to the MVCS module during the member calculations by means of the discussed method by calling it.

CalculDeflDisp (dd : IRDimDeflDisp) : bool
The method called by the RDIM module and forcing user's calculations.

ReCalculDeflForProf (coefY : double , coefZ : double) : bool
The method called by the RDIM module in order to update the SLS calculations results after the substitution of the member's profile during dimensioning. The parameters of this method are the values of the ratios of the moments of inertia (Y and Z) for the first profile and the new one.

GetDeflDispMaxRatio ( ) : double
Returns the value of the maximum Ratio calculated separately for displacement and deflections.

GetResultsInterface () : IRDimMembRes
Returns the interface to support the calculations results.

Finish (fin : IRDimMembCalcFinishType) : bool

The method called by the RDIM module after a given stage of calculations or the whole of calculations have been finished (all the iterations).

GetMessages (message_no : long) : string

The method called by the RDIM module after all the calculation stages have been finished (all the iterations). The method is called repeatedly (number of calls is determined by the MessagesNum attribute) and returns the messages that the RDIM module displays in the appropriate tab control in the short results window. The "Messages_no" parameter accepts values between 1 and the MessagesNum attribute value.

GetMessagesType (message_no : long) : IRDimMembCalcMessageType

The method called by the RDIM module after all the calculation stages have been finished (all the iterations). The method is called repeatedly (the number of calls is determined by the MessagesNum attribute) and returns the type of message (message, error message). The type of message determines the appearance of the icon which will be displayed beside the message. The"messages_no" parameter accepts the values between 1 and the MessagesNum attribute value.

IRDimCodeService

- **Implementation in MVCS module**

The interface implementing the basic services necessary for the RDIM module to support the external code implementations.

Functions:

GetDefaultMembDef (_type : IRDimMembDefType) : IRDimMembDef

Returns the default definition of the code characteristics for the member of the given type. Such definition will be used by the RDIM module to create the label, the name of which would be transferred by the IRDimMembDef interface.

EditMembDef (inModalWindow : bool, parentHWND : long, val : IRDimMembDef)

Method called by the RDIM module in order to edit all the code characteristics of the member. After this method has been called, the MVCS module should display a dialog box in which the user could edit all the parameters that he needs.
If a TRUE value is assigned to the ‘inModalWindow’ parameter, the above mentioned dialog box has to be displayed in the modal mode. Otherwise the dialog box should be displayed in the non modal mode. The ‘parentHWND’ parameter defines the window handle which is the parent for the code data edition dialog box of the member created by the MVCS module.

GetMembCalc () : IRDimMembCalc

Returns the interface to support the SLS and ULS.

ModalEditManualParams (parentHWND : long, val : IRDimManCalcPar) : IRDimManParRetValue

Method called by the RDIM module to edit the data for the manual member calculations. After this method has been called, the MVCS module should display a modal dialog box in which the user could edit parameters supported by the IRDimManCalcPar interface. The ‘parentHWND’ parameter defines the window handle which is the parent for the code data edition dialog box of the member created by the MVCS module.

ModalEditCalcConfigParams (parentHWND : long, val : IRDimCalcConf) : bool

Method called by the RDIM module to edit the data defining the calculations configuration. After this method has been called, the MVCS module should display the modal dialog box in which the user could edit the parameters supported by the IRDimCalcConf interface. The ‘parentHWND’ parameter defines the window handle which is the parent for the code data edition dialog box of the member created by the MVCS module.

IRDimClientCalcConfDefaultWndType

Definition of the set of identifiers defining the type of dialog boxes of the calculations configuration implemented in the RDIM module and accessible by the COM interface.

Attributes:

I_DCCCDWT_BASIC_TYPE : = 1

Defines the basic dialog box of the calculation configuration which enables the edition of all the basic parameters of calculation configurations.

I_DCCCDWT_WITH_DURATION_OPT_TYPE : = 2

Defines the dialog box of the calculation configuration which has been extended in respect of the basic dialog box with a load case classification option according to their duration.

IRDimClient

- **Implementation in the MVCS module**

  The main interface providing supports implemented in the user's MVCS module. In the present version this support concerns only the actions directly related to the timber and steel calculation standard implemented in the customer's module.

Functions:

Connect () : IRDimConnection
Sets the connection of MVCS module with RDIM module.

GetRDimCodeService () : IRDimCodeService
Returns the interface implementing the basic supports which are necessary for the RDIM module to support the external code implementations.

GetCodeUnitName (type : IRDimUnitType) : string
Returns the names of the standard units that is units that have to be used to present the results when the units settings are not considered in the Robot program configuration module. That is the case when in the calculations configuration window of the RDIM module the "standard" button is active in the results unit area.

GetCodeUnitCoef (type : IRDimUnitType) : double
Returns the code factors that is factors by which should be multiplied the numerical values written in the internal units of the RDIM module (N, m). They should represent the data in code units used by the standard implemented in the MVCS module and the names of which are returned by the GetCodeUnitName method.

GetCalcConfDefaultWndType() : IRDimClientCalcConfDefaultWndType
Returns the message about the dialog box (among those implemented in the RDIM module) which has to be used to set the calculations configuration parameters, in case the customer's MVCS module doesn't implement this dialog box.

IRDimUnitType

Definition of the set of identifiers defining the units defined for the IRDimUnits interface.

Attributes:

I_DUT_SECDIMEN : = 1
Defines the unit of the section dimensions (for example "m")

I_DUT_SECSUR : = 2
Defines the unit of the section characteristics (for example "cm2")

I_DUT_SECVOL : = 3
Defines the unit of the section characteristics (for example "cm3")

I_DUT_SECMI : = 4
Defines the unit of the section characteristics (for example "cm4")

I_DUT_LENGTH : = 5
Defines the unit of the structure dimensions.

I_DUT_FORCE : = 6
Defines the unit of the forces.

I_DUT_DISPL : = 7
Defines the unit of displacements.

I_DUT_MOMENT : = 8
Defines the unit of moments.

I_DUT_STRESS : = 9
Defines the unit of stresses.

I_DUT_NONE : = 10
Defines the non dimensional dimension.

IRDimUnits

- **Implementation in the RDIM module.**

Interface providing information about units.

Attributes:

<GET>AreRobotUnits : bool
Returns the information whether the interface supports the units set in the task preference window or the default units. The default units are:

- m -- for length
- N - for forces
- Nm - for moments
- m - for displacements
- N_m2 - for stresses.

Functions:

Refresh (areRobotUnits : bool)
The method causes the internal data update. The returned information have to be conform with the settings in the task preference window.

ReadToUserCoef (type : IRDimUnitType) : double
Returns the factor for the user units. The factor is a numerical value by which should be multiplied the number given in default unit to obtain it in the user unit.

ReadUserName (type : IRDimUnitType) : string
Returns the user units names.

Format (type : IRDimUnitType, val : double) : string
Returns the format in which numerical values should be displayed (in accordance with the settings in the task preferences window).

IRDimCalcConfFlagType

Definition of the set of identifiers defining the type of the calculation configuration option.

Attributes:

I_DCCFT_CHECKSLEND : = 1
Defines the option considering the maximum slenderness.

I_DCCFT_FIRE : = 2
Defines the option considering the possibility of fire impact

I_DCCFT_INACTIVECOMP : = 3
Defines the option not considering the components of the complex members during the calculations

I_DCCFT_CHARPOINTS : = 4
Defines the option of the calculation in characteristic points.

I_DCCFT_N_AND_CHAR_POINTS : = 5
Defines the option considering the equal intervals points while the calculation option in characteristic points is selected.

I_DCCFT_MAX_FX_POINT : = 6
Defines the option of calculation in the characteristic point of the FX force maximum value.

I_DCCFT_MAX_FY_POINT : = 7
Defines the option of calculations in the characteristic point of the FY force maximum value.

I_DCCFT_MAX_FZ_POINT : = 8
Defines the option of calculation in the characteristic point of the FZ force maximum value.

I_DCCFT_MAX_MY_POINT : = 9
Defines the option of calculations in the characteristic point of the MY moment maximum value.

I_DCCFT_MAX_MZ_POINT : = 10
Defines the option of calculations in the characteristic point of the MZ moment maximum value.

I_DCCFT_USERPOINTS : = 11
Defines the option taking into account characteristic points entered by the user.

I_DCCFT_USERPOINTS_ABS : = 12
Determines as absolute the values of the characteristic points coordinates entered by the user.

I_DCCFT_ROBOTUNITS : = 13
Defines the information whether the unit support should consider the units set in the task preference window or whether there should be default units.

I_DCCFT_INITIAL_DEFLECTIONS : = 14
Causes camber to be taken into account during the analysis.

IRDimCalcConfValueType

Definition of the set of identifiers defining the numerical values of the calculation configuration parameters.

Attributes:

I_DCCPV_MAXSLEND : = 1
Defines the maximum slenderness.

I_DCCPV_EFFRATIO : = 2
Defines the efficiency ratio limit value.

SetAutoDeflectionsLoadCase (case_no : long, comp_index : long)
Sets the load case which will be used to define initial values of cambers.

GetAutoDeflectionsLoadCase (case_no : long*, comp_index : long*)
Returns the load case which will be used to define initial values for cambers.

IRDimCalcConf

- **Implementation of the RDIM module**

Interface providing the code calculation configuration parameter. It allows to modify them and, additionally, enables the set extension.

Attributes:

NPointsNum : long

The attribute defines the number of the equal interval points.

<GET>CasesNum : long

The attribute defines the number of the load cases available for the calculations.

Functions:

SetFlag (_type : IRDimCalcConfFlagType, val : bool)

Sets the calculation option with the given attribute.

IsFlagSet (_type : IRDimCalcConfFlagType) : bool

Verifies whether the calculation option with the given attribute has been set.

SetParamValue (_type : IRDimCalcConfValueType, val : double)

Sets the value of the calculation configuration numerical parameter with the given attribute.

GetParamValue (_type : IRDimCalcConfValueType,) : double

Returns the value of the calculation configuration numerical parameter with the given attribute.

ClearUserCharPoints ()

Removes all the characteristic points entered by the user.

GetUserCharPointsNum () : long

Returns the value of all the characteristic points entered by the user.

WriteUserCharPoint (val : double)

Saves the successive characteristic point with the given coordinate.

ReadUserCharPoint (no : long) : double

Reads the successive characteristic point (it's coordinate). The "no" method parameter is the next characteristic point number.

WriteClientData (val : IRDimStream)

Saves any calculation configuration data which would afterwards be necessary for the customer's MVCS module to carry out the calculations. Afterwards, the RDIM module will copy those data to the IRDimCalcState interface during the verification of the member calculation support, groups verification, dimensioning or optimization. As a result they would be available for the IRDimMembCalc calculation interface which would allow the MVCS module to use them in the strict code calculations.

ReadClientData (val : IRDimStream)

Reads any calculations configuration data previously entered by the customer's MVCS module. The RDIM module copies those data to the IRDimCalcState interface while supporting the member verification calculations, groups verification, dimensioning or optimization. As a result, they are available for the IRDimMembCalc interface which allows the MVCS module to use them for the strict code calculations.

ModalEditBasicParams ()

Allows the edition of some basic parameters of the calculation configuration (in the special dialog box implemented in the RDIM module). The method should be used (in case the edition of the basic calculation configuration data is necessary) in the implementation of the support of the manually entered data for the manual calculations (the IRDimManCalcPar interface).

GetCaseNo (index : long , case_no : long*)

Returns the internal number of the load case defined by the successive number included in the " index" parameter (the " index" parameter takes the values between 1 and the value returned by the CasesNum attribute).

GetCaseCompNum (case_no : long , ret : long*)

Returns the number of the components of the given load case defined by the internal number included in the " case_no" parameter. For the simple cases and combinations the value is always 1.

GetCaseName (case_no : long , comp_index : long , ret : string)

Returns the name of the given load case or the code combination component. For the simple cases or combinations the " comp_index" parameter should be 1.

IRDimManParRetValue

Definition of the set of identifiers defining the command selected by the user in the manual calculations parameters dialog box. This dialog box is implemented in the customer's MVCS module.

Attributes:

I_DMPRV_OK : = 1

Indicates acceptance.

I_DMCFT_CANCEL : = 2

Indicates canceling.

I_DMCFT_CALCULATION: = 3

Indicates activation of calculations.

IRDimManCalcPar

- **Implementation in the RDIM module.**
  Interface that allows data transmission to and from the modal dialog box (supporting the manual calculations parameters) implemented in the customer's MVCS module in which the user will be able to edit the parameters supported by this interface.

Attributes:

ManMembNo : long
The attribute defines the internal member number for manual calculations.

Functions:

SetManMembDef (memb_def : IRDimMembDef)
Sets the definition of the code parameters for the member with the ‘ManMembNo’ attribute.

GetManMembDefAccess () : IRDimMembDef
Returns the definition of code parameters for the member with the ‘ManMembNo’ attribute.
SetManEffDef (eff_def : IRDimSimEffDef)
Sets the data about the internal forces.

GetManEffDefAccess () : IRDimSimEffDef
Provides data concerning the internal forces and additional moments.

SetCalcConf (conf_data : IRDimCalcConf)
Sets the interface providing o the code calculations configuration parameters.

GetCalcConfAccess () : IRDimCalcConf
Provides the interface giving access to the code calculations configuration parameters.

GetMembNum () : long
Returns the number of structure members available for code calculations.

GetMembNo (index : long) : long
Returns the internal member number with the successive number (between 1 and the value returned by the ‘GetMembNum’ method.

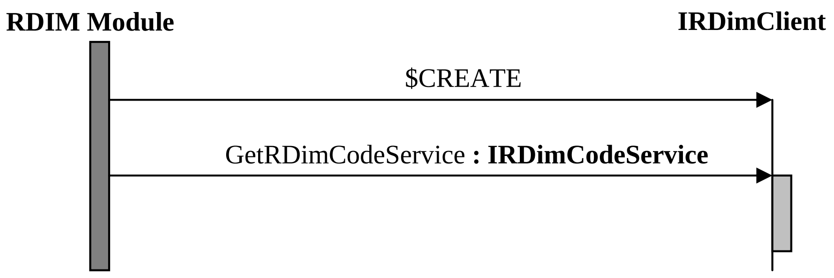
GetMembUserNo (memb_no : long) : long
Returns the user member number with its the internal number.

GetMembName (memb_no : long) : sting
Returns the name of member. ‘memb_no’ parameter is the internal member number.

GetMembDef (member_no : long) : IRDimMembDef
Returns the definition of the code parameters for the member with the internal number defined by the ‘member_no’ parameter.

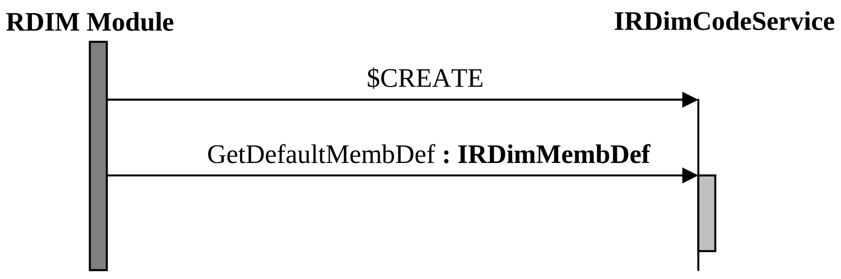
3.4 Communication between Robot and MVCS module

RDIM module communicate with MVCS code module activating appropriate methods of the interfaces implemented there. There is the guid of the class implementing the **IRDimClient** interface saved in the register and this is the creation of this interface where the communication with MVCS module starts. Once this interface is created, RDIM module passes to MVCS component – the indicator to the IRDimConnection interface calling up the Connect method. It gets information about the units applied in the implemented code and through activation of the GetRDimCodeService method obtains the IRDimCodeService interface implementing the code service.

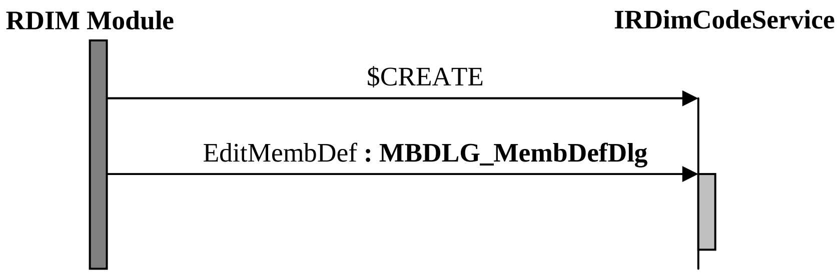


Being given access to the code service such as IRDimCodeService, RDIM module may perform its regular tasks:

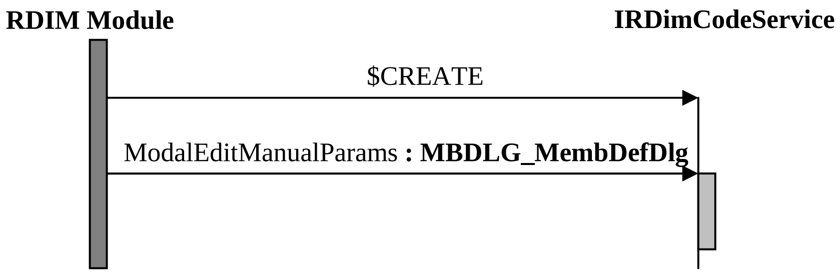
- generation of basic sets of member code parameters, labels – beam, column



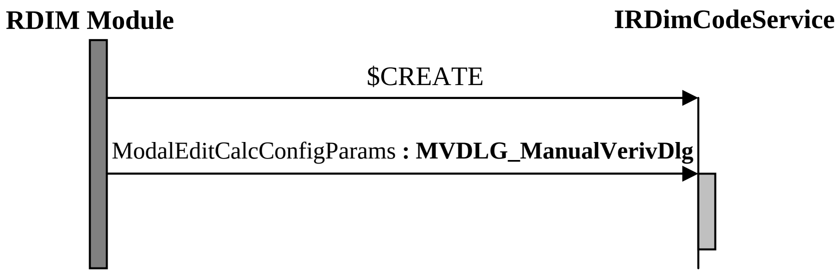
- edition of member code parameters



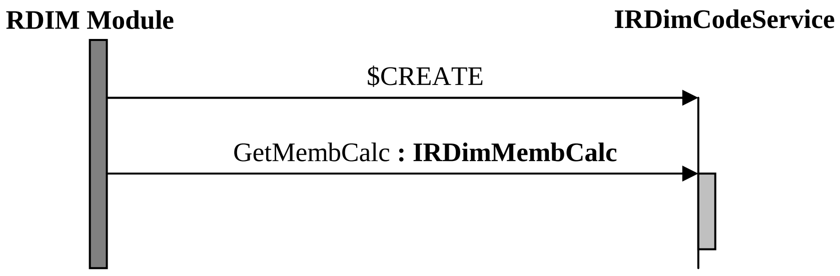
- edition of data for member manual calculations



- edition of data for configuration of calculations



- member code calculations

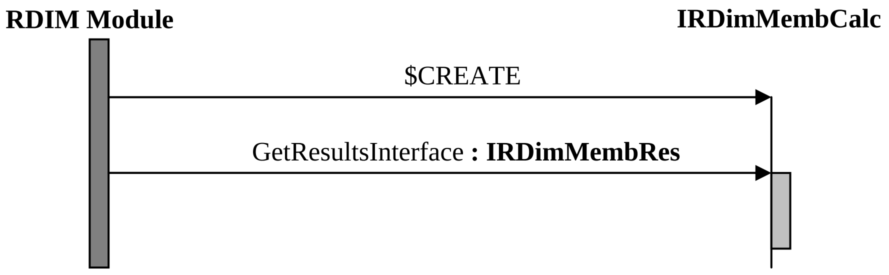


Having obtained the object of the IRDimMembCalc interface, RDIM module may carry out any calculations of member verification, group verification or design. It sets input parameters at any calculation point by means of the following:

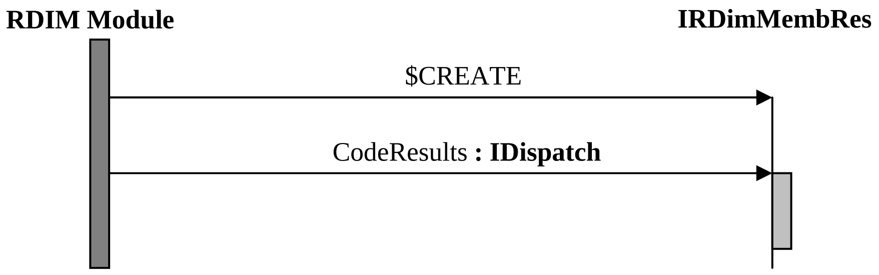
- SetMembDef method and IRDimMembDef interface
- SetMatDef method and IRDimMatDef interface
- SetProfDef method and IRDimProfDef interface

- SetEffDef method and IRDimEffDef interface
- SetCalcState method and IRDimCalcState interface

Member calculation for a given point are run by means of the CalculMember method for Ultimate Limit States and by means of the methods CalculDeflDisp and ReCalculDeflForProf for Serviceability Limit States. RDIM module obtains access to calculation results using the GetResultsInterface method.

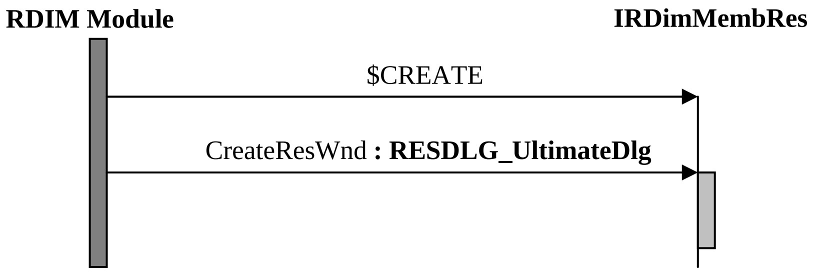


From the interface acquired by means of this method standard calculation results are taken. Access to specific, typically code-dependent numeric results is gained by RDIM module by calling up the CodeResults method.

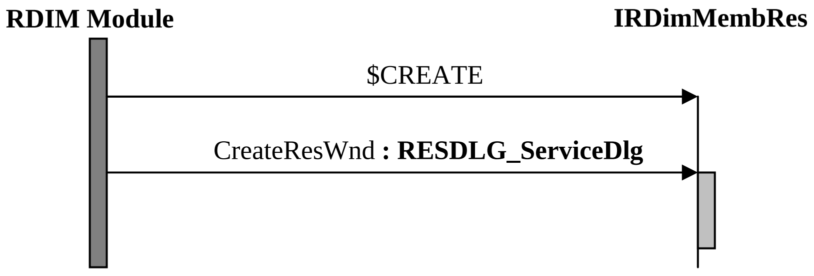


Tabs in the dialog box presenting in detail calculation results are made accessible by means of the CreateResWnd method.

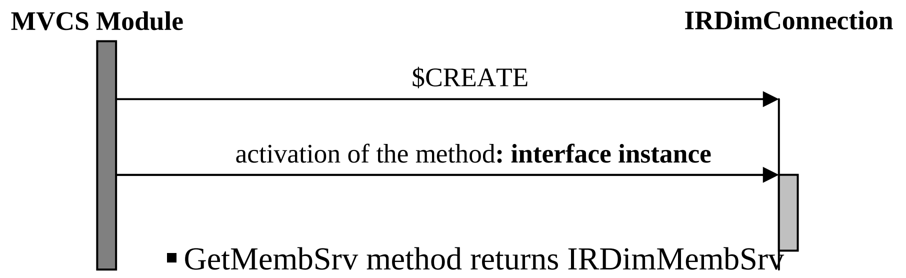
- for calculations of Ultimate Limit States



- for calculations of Serviceability Limit States



MVCS component has the indicator to the RdimConnection interface (it has been passed by means of the Connect method of the IRDimClient interface), using which it may communicate with RDIM module calling up appropriate methods. In this way it may also acquire indispensable instances of some interfaces.



- GetMembSrv method returns IRDimMembSrv
- GetUnits method returns IRDimUnits
- GetStream method returns IRDimStream
- GetAdjoinParams method returns IRDimAdjoinParams
- GetKernel method returns Idispatch
- GetGrpProfs method returns IRDimGrpProfs

3.5 Registration of MVCS Module

MVCS module is registered in the standard manner using for this purpose the regsvr32.exe program delivered together with the operating system in the system32\ folder.

```
system32\regsvr32.exe  norm.dll
```

This is an example name of dll library file here; the name of MVCS framework module presented in the next chapter is an example, as well.
Apart from registration of all the implemented interfaces, the code module has to record a ”new” component in the register so that the Robot program is able to ”spot” it. For the framework project described in the next chapter this record takes the following form:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Autodesk\Structural\COM
Services\Norm]
"Code Name"="Full norm code name"
"Group ID"=dword:00000000
"GUID"="{64490C83-D0BF-444E-A8F5-CE7BDA73A788}"
"GUID Type"=dword:00000001
```

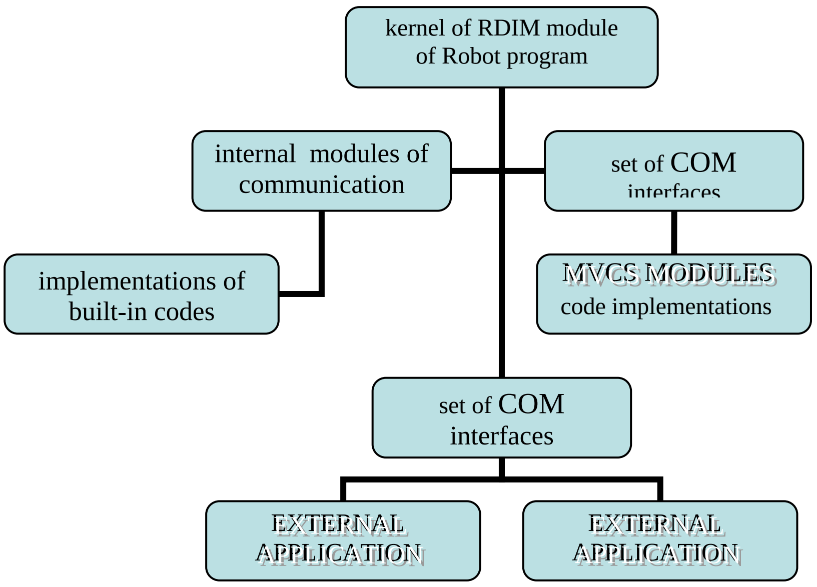
For the already existing component of BS59 code this record takes the following form:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Autodesk\Structural\COM
Services\BS59_2000]
"Code Name"="BS 5950:2000"
"Group ID"=dword:00000000
"GUID"="{A8C7D4FF-9285-4E54-9951-E4695EFD6F43}"
"GUID Type"=dword:00000001
```

4. Communication of RDIM Module with External Applications

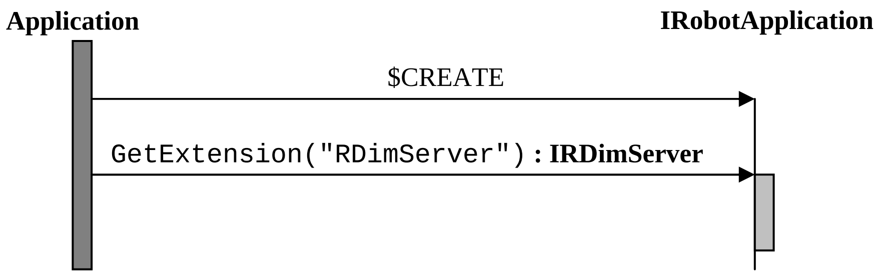
4.1 Introduction

RDIM module being a part of the Robot program implements COM interfaces which enable external applications to take advantage of its computational capabilities. It is illustrated by the diagram below:


kernel of RDIM module of Robot program
internal modules of communication
set of COM interfaces
implementations of built-in codes
MVCS MODULES code implementations
set of COM interfaces
EXTERNAL APPLICATION
EXTERNAL APPLICATION

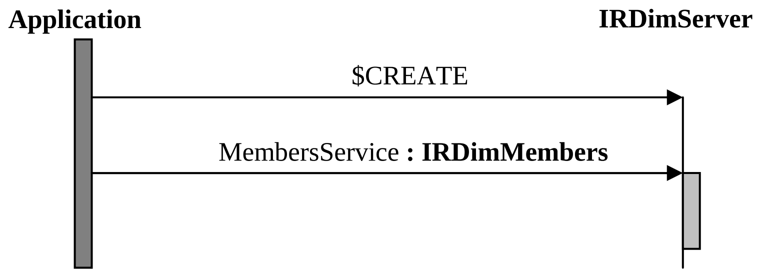
4.2 Rules of Communication with RDIM Module

An external application acquires access to RDIM module via the **IRDimServer** interface**.** To create it, this application applies the main interface of the Robot program. It is presented in the drawing below:

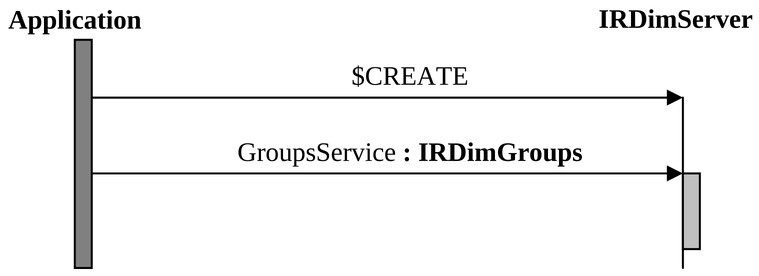


Having gained the access to the RDIM module server, which is the IRDimServer interface, an external application is able – by obtaining the relevant interface – to
carry out all the operations needed by calculations:

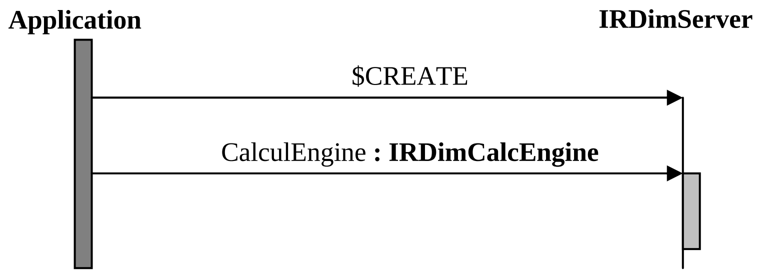
- it may manipulate with members and code parameter labels assigned to it, creating or modifying them, if need be



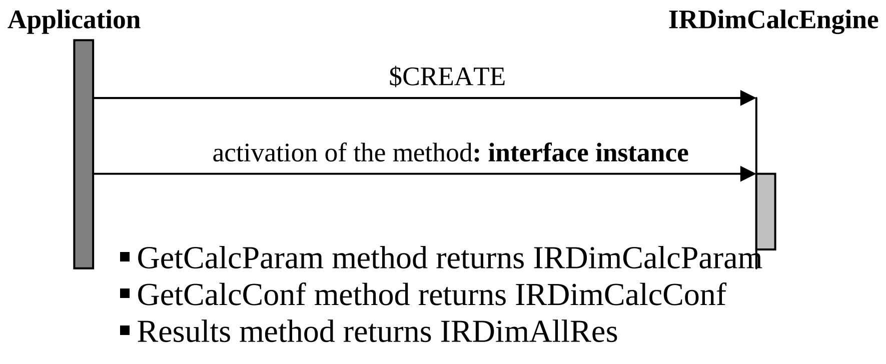
- it may create member groups, modify and delete them



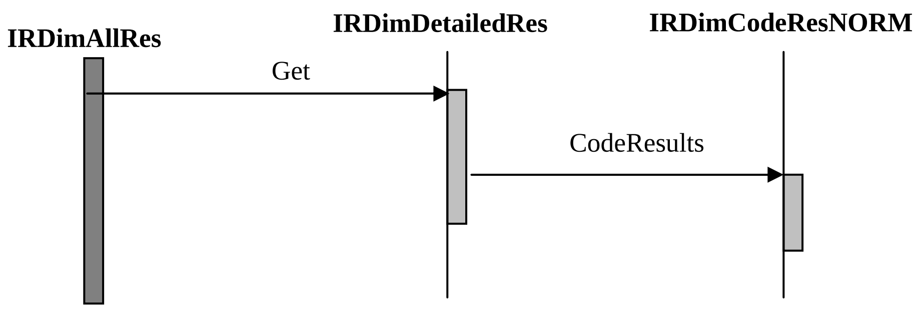
- it may perform determined calculations and get their results



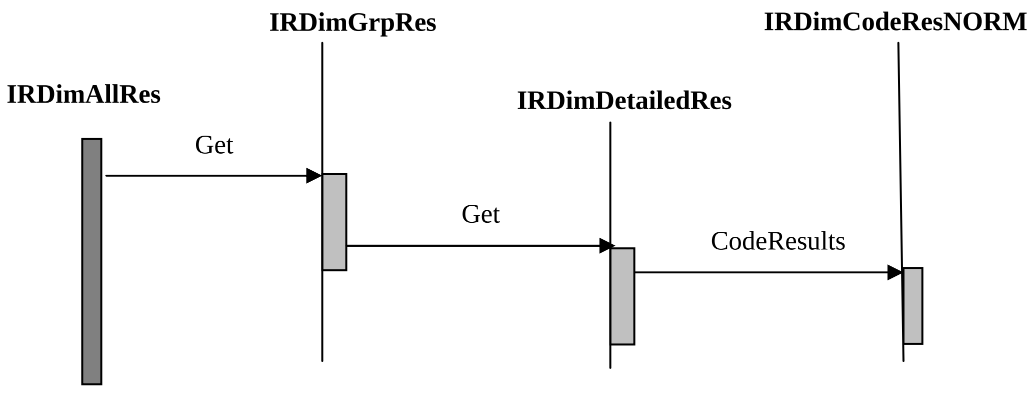
Before an external application runs calculations, with the use of the IRDimCalcEngine interface it modifies default settings of configuration and calculation parameters. It is able to do it after obtaining the access to appropriate interfaces.
Calculations are run by the Solve method. Once they are completed, an external aplication gets access to their results using the IRDimAllRes interface.


Application
IRDimCalcEngine
$CREATE
activation of the method: interface instance
▪ GetCalcParam method returns IRDimCalcParam
▪ GetCalcConf method returns IRDimCalcConf
▪ Results method returns IRDimAllRes

In case of verification of members or groups, access to calculation results of various type and level of detail proceeds as shown in the diagram below:



For group design the sequence of calls becomes longer and looks as follows:



4.3 General Description of COM Interfaces

Below is a description of all the interfaces used by external applications to communicate with RDIM module.

IRDimServerMode

Definition of a set of identifiers for the IRDimServer interface.

Attributes:

I_DSM_STEEL : = 1
Sets the work mode - steel.

I_DSM_TIMBER : = 2
Sets the work mode - timber.

IRDimServer

- **Implementation in RDIM module**

This interface is a basis for communication with RDIM module. Practically all the operations derive from this interface, since for a given instance of Robot it returns specific interfaces supporting individual tasks, such as support of members and groups or control of the process of calculation and getting results.

Attributes:

<PUT>Mode : IRDimServerMode
Sets the work mode – steel or timber

<GET> MembersService : IRDimMembers
Returns an interface used for complex support of members.

<GET> GroupsService : IRDimGroups
Returns an interface used for complex support of groups.

<GET> CalculEngine: IRDimCalcEngine
Returns a support interface used to perform calculations of verification and design/optimization.

<GET> Connection: IRDimConnection
Returns an auxiliary interface for a given instance of the Robot program.

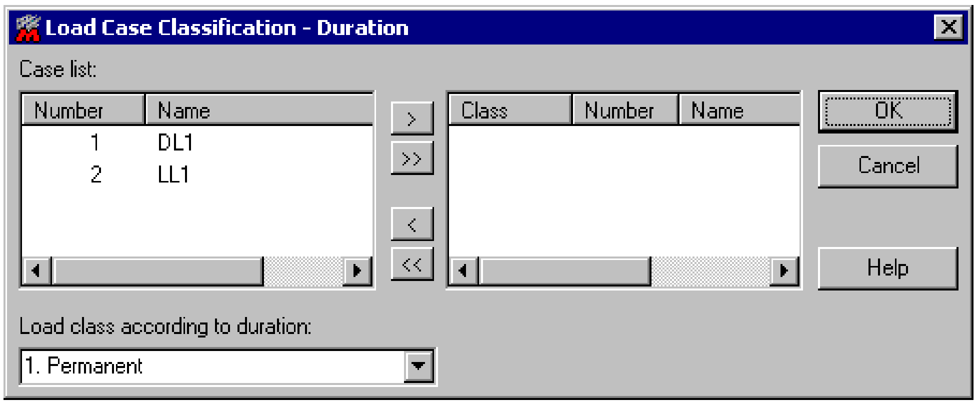
IRDimConnectionMsg

Definition of a set of identifiers for the IRDimConnection interface.

Attributes:

I_DCM_DETAILED_ANALYZE_RTF_PRINT: = 1
Runs the mechanism of file generation in the rtf format on the basis of a template file.

I_DCM_SHOW_LOADCASES_DIALOG: = 2
Shows the ”Load Case Classification – Duration” dialog box implemented in RDIM module which includes options that enable ascribing an appropriate class of load duration to simple load cases defined earlier. This dialog box is presented below:


Load Case Classification - Duration

Case list:

Number	Name
1	DL1
2	LL1

Class	Number	Name

Load class according to duration:

1. Permanent

IRDimConnection

- **Implementation in RDIM module**
  This interface is an auxiliary object for a given Robot instance which makes manipulation with other interfaces and their creation more efficient. It also returns a few parameters of Robot's installation and configuration.

Atrybuty:

<GET> Language : long
Returns a number of the language currently used.

<GET> ResPath : string
Returns a full path to the Robot's folder containing resources.

<GET> BinPath : string
Returns a full path to the Robot's folder containing binary files.

<GET> HelpPath : string
Returns a full path to the Robot's folder containing help files.

<GET> CfgPath : string
Returns a full path to the Robot's folder containing configuration files.

Functions:

GetMembSrv (ret : IRDimMembSrv)
The method returns an interface intended for complex support of member parameters, applies data supported by the IRDimMembDef interface.

GetUnits (ret : IRDimUnits)
Sets an interface for unit support

GetStream (ret : IRDimStream)
The method returns an interface implementing any data stream. Data saving and reading looks similar to saving in a file, but there are three files(streams) separate for each data type.

GetAdjoinParams (ret : IRDimAdjoinParams)
The method returns an interface which enables automatization of calculations in which adjoining bars are considered.

GetKernel (ret : IDispatch)
The method returns an interface of the engine of the Robot system.

SendMessage (msg : IRDimConnectionMsg, IDispatch *dat)
Sends information – to RDIM module – about the state of a client or instructs it to perform a specific task.

GetGrpProfs (ret : IRDimGrpProfs)
The method returns an interface which enables definition of a section family for calculations of member group design.

IRDimMembDefDataSrv

- **Implementation of RDIM module**

The interface supports labels of member parameters. This interface enables their creation, saving and reading.

Attributes:

<GET> Count : long
Returns a number of existing labels of member parameters.

Function:

Name (index : long, ret : string)
Returns names of labels for successive indices in a collection.

Get (name : string, ret : IRDimMembDefData)
Returns an interface being a definition of a member parameter label.

Check (name : string, is_defined : bool*, can_be_saved : bool*)
The method checks if a label with the specified name already exists in the system and if code properties of the member with such a label can be saved (updated) by means of the “Save” method. If the label already exists and determined code properties are ascribed to it and consequently, the member ‘Type’ attribute is defined, then the ‘can_be_saved’ variable is given the TRUE value only when this attribute is given the I_DMDT_USER value.

Save (val : IRDimMembDefData)
Saves a label of member parameters.

IRDimMember

- **Implementation in RDIM module**
  This interface represents entire data concerned with a member. By default, all the existing structure members are the members of RDIM module. Every other member defined in this module is a ”superbar”. A superbar is an abstract object ascribed the same attributes as member, but, additionally a list of members is assigned to it. Thus it is a compound member. Superbars are defined, generated and deleted only in RDIM module.

Attributes:

<GET> UserNo: long
Returns member’s number.

<GET> IsComplex : bool
Returns information about that, if the bar is a ”superbar”.

Name : string
Represents member’s name.
DefDataName : string
The attribute determines name of the ascribed label.

Functions:

GetComplexList (in : IRDimStream)
Returns a list of members that a superbar is composed of. For a ”regular” member the method returns a number of this member.

GetComplexList (in : IRDimStream)
Sets the list of members that a superbar is composed of.

IRDimMembers

- **Implementation of RDIM module**
  This interface enables manipulation with members. It is a collection of members.

Attributes:

<GET> Count : long
Returns a number of existing members and superbars.

<GET> DefDataService : IRDimMembDefDataSrv
Returns an interface supporting member parameter labels. This interface enables their creation, saving and reading.

Functions:

UserNo (index : long, no : long*)
Returns member's number for successive indices in a collection.

Get (no : long, ret : IRDimMember)
Returns an interface representing entire data concerned with a member.

Save (val : IRDimMember)
Saves entire data concerned with a member directly in RDIM module.

New (like_no : long, new_no : long, ret : IRDimMember)
Generates a new superbar and returns an interface containing entire data connected with it. A new member is generated based on a definition of member with the number given in the 'like_no' parameter. A new member number is assigned on the basis of the 'new_no' parameter. If it equals zero, a new member is given a default number – the first free one.

Delete (no : long, succ : bool*)
Deletes a superbar with the specified number.

Generate (in : IDispatch)
The method implements a specifc superbar generator.

IRDimGrpProfs

- **Implementation in RDIM module**
  This interface is a set of collections of names for sections used in calculations of design / optimization. For each existing database name there is a collection of section names.

Attributes:

<GET> Count : long
Returns a number of used names of section databases.

Functions:

GetBase (index : long, base_name : string)
Returns names of section databases.

Clear ()
Deletes entire data.

ClearProfs (base_name : string)
Deletes all the section names from the collection connected with a given database name.

GetProfs (base_name : string, in : IRDimStream)
Returns in the IRDimStream stream – all the section names connected with the database name given in the 'base_name' parameter.

SetProfs (base_name : string, in : IRDimStream)
Creates a collection of section names (or extends the existing one) associating it to the database name given in the 'base_name' parameter. Full section names are included in the 'in' parameter (in a text stream), e.g.: "IPE 360".

SetFamilies (base_name : string, in : IRDimStream)
Creates a collection of section names (or extends the existing one) associating it to the database name given in the 'base_name' parameter. Full names of section families are included in the 'in' parameter (in a text stream), e.g.: "IPE","HEA". Names of families are expanded into collections of section names based on the physical contents of the databases of the Robot system.

IRDimGroup

- **Implementation in RDIM module**
  This interface represents entire data connected with a group of members, used in the process of design/optimization.

Attributes:

<GET> UserNo: long
Returns a number of a member group.

Name : string
This attribute represents name of a member group.

Material : string
This attribute determines name of the material that will be used in calculations.

Functions:

GetMembList (in : IRDimStream)
Returns a list of members that a group is composed of.

SetMembList (in : IRDimStream)
Sets a list of members that a group is composed of.

GetProfs (in : IRDimGrpProfs)
Returns a collection of sections for design/optimization calculations.

SetProfs (in : IRDimGrpProfs)
Sets a collection of sections for design/optimization calculations.

GetParamProfs (in : IRDimStream)
Returns a list of parametrized sections for design/optimization calculations.

SetParamProfs (in : IRDimStream)
Sets a list of parametrized sections for design/optimization calculations.

IRDimGroups

- **Implementation in RDIM module**

This interface enables manipulation with members. It is a collection of members.

Attributes:

<GET> Count : long
Returns a number of the existing groups.

Functions:

UserNo (index : long, no : long*)
Returns group's number for the successive indices in a collection.

Get (no : long, ret : IRDimGroup)
Returns an interface representing entire data concerned with a group.

Save (val : IRDimGroup)
Saves entire data concerned with a group directly in RDIM module.

New (like_no : long, new_no : long, ret : IRDimMember)
Creates a new group and returns the interface containing entire data concerned with it. A new group is generated based on a definition of the group with a number given in the 'like_no' parameter. A number of new group is ascribed on the basis of the 'new_no' parameter. If it equals zero, a new group is given a default number – the first free one.

Delete (no : long)
Deletes a group with the specified number.

Generate (in : IDispatch)
The method implements a specific group generator.

GetParamProfsPattern (in : IDispatch)
The method returns templates of parameterized sections.

SetParamProfsPattern (in : IDispatch)
The method sets templates of parameterized sections.

IRDimOptimParamOptionType

Definition of a set of identifiers for the methods SetOption and GetOption of the `IRDimOptimParam` interface. The identifiers determine the kind and type of optimization.

Attributes:

I_DOPOT_WEIGHT: = 1
Optimization due to weight.

I_DOPOT_MAX_SECTION_HEIGHT: = 2
Section height with the top limit.

I_DOPOT_MIN_SECTION_HEIGHT: = 3
Section height with the bottom limit.

I_DOPOT_MIN_FLANGE_THICKNESS: = 4
Flange thickness with the bottom limit.

I_DOPOT_MIN_SECTION_HEIGHT: = 5
Web thickness with the bottom limit.

I_DOPOT_ENTIRE_SET_SECTIONS: = 6
Calculations for a full section set. If this option is switched on, then during calculations – in order to find the most optimal section, the whole available section family is searched through (it is significant, particularly if sections in the database are not arranged in the ascending order, i.e. if the next section is not ”larger” than the previous one).

IRDimOptimParamLimitType

Definition of a set of identifiers for the methods SetLimit and GetLimit of the `IRDimOptimParam` interface.

Attributes:

I_DOPLT_MAX_SECTION_HEIGHT: = 2
The maximum section height cannot be greater than the specified limit.

I_DOPLT_MIN_SECTION_HEIGHT: = 3
The minimum section height cannot be lesser than the specified limit.

I_DOPLT_MIN_FLANGE_THICKNESS: = 4
The minimum flange thickness cannot be lesser than the specified limit.

I_DOPLT_MIN_WEB_THICKNESS: = 5
The minimum web thickness cannot be lesser than the specified limit.

IRDimOptimParam

- **Implementation in RDIM module**

The interface enabling definition of optimization parameters.

Attributes:

Optimization : bool
Switches on/off the optimization.

Functions:

SetOption (option : IRDimOptimParamOptionType, val : bool)
Switches on or off a given kind, type of optimization (e.g. optimization by section weight).

GetOption (option : IRDimOptimParamOptionType, ret : bool*)
Returns information about that if a given kind, type of optimization is switched on or not.

SetLimit (option : IRDimOptimParamLimitType, val : double)
Sets the limit for a given kind, type of optimization (e.g. the minimum section height).

GetLimit (option : IRDimOptimParamLimitType, ret : double*)
Returns the limit for a given kind, type of optimization (e.g. the maximum section height).

IRDimCalcParamVerifType

Definition of a set if identifiers for the VerifType attribute and the SetObjsList method of the `IRDimOptimParam` interface.

Attributes:

I_DCPVT_MEMBERS_VERIF: = 1
Verification of members.

I_DCPVT_GROUPS_VERIF: = 2
Verification of groups.

I_DCPVT_GROUPS_DESIGN: = 3
Design of groups.

IRDimCalcParamLimitStateType

Definition of a set if identifiers for the methods SetLimitState and GetLimitState of the IRDimCalcParam interface.

Attributes:

I_DCPLST_ULTIMATE: = 1
Ultimate Limit State.

I_DCPLST_SERVICEABILITY: = 2
Serviceability Limit State.

IRDimCalcParam

- **lmplementation in the RDIM module**

The interface enabling definition of calculation parameters.

Attributes:

<GET>VerifType : IRDimCalcParamVerifType
Sets calculation type, member verification, group verification or group design.

Functions:

SetObjsList (type : IRDimCalcParamVerifType, in : IRDimStream)
Sets the list of members or groups used in calculations depending on the type of these calculations. If the 'in' parameter equals NULL (nothing), a full list of members or groups is taken into account in calculations.

GetObjsList () : IRDimCalcParamVerifType
Returns a list of members or groups used in calculations.

SetLimitState (type : IRDimCalcParamLimitStateType, val : bool)
Determines how calculations should be performed, for the Ultimate Limit State, Serviceability Limit State or for both of them at a time (for verification).

GetLimitState (type : IRDimCalcParamLimitStateType) : bool
Returns limit state settings for calculations.

SetLoadList (in : IRDimStream)

Sets a list of load cases considered in calculations. . If the ‘in’ parameter equals NULL (nothing), all the load cases are taken into account in calculations.

GetLoadList () : IRDimStream

Returns a list of load cases considered in calculations.
in] IRDimStream *in);

SetOptimParam (val : IRDimOptimParam)

Sets calculation parameters.

GetOptimParam () : IRDimOptimParam

Returns settings of calculation parameters.

IRDimCalcEngine

- **Implementation in RDIM module**

  This interface is the basis for all the calculations concerned with member verification, group verification, design and optimization.

Functions:

GetCalcParam () : IRDimCalcParam

Returns the interface supporting settings of calculation parameters.

SetCalcParam (val : IRDimCalcParam)

Saves defined calculation parameters.

GetCalcConf () : IRDimCalcConf

Returns the interface of calculation configuration.

SetCalcConf (val : IRDimCalcConf)

Saves the determined calculation configuration.

Solve (observer : IDispatch)

Starts calculations. The ‘observer’ parameter is an object receiving events which inform about the course of calculations. The observer object also enables running calculations in a separate thread. At present, this parameter is not considered and should be ascribed the value ‘nothing’.

Terminate ()

Stops calculations.

Results () : IRDimAllRes

The method returns a collection of all calculation results.

IRDimAllResObjectType

Definition of a set of identifiers for the ObjectsType attribute of the IRDimAllRes interface.

Attributes:

I_DAROT_VERIFIED_MEMBER: = 1
Indicates objects of member verification results.

I_DAROT_VERIFIED_GROUP: = 2
Indicates objects of group verification results.

I_DAROT_DESIGNED_GROUP: = 3
Indicates objects of group design results.

IRDimAllRes

- **Implementation in RDIM module**
  The interface is a collection of all the results of a given calculation process. This collection is a set of objects. Each such object stores calculation results of a member – for member verification or of a group – for verification or design of groups.

Attributes:

<GET> CodeName : string
Returns a full code name.

<GET> ObjectsType: IRDimAllResObjectType
Returns a type of result objects.

<GET> ObjectsCount : long
Returns a number of result objects.

Functions:

ObjectUsrNo (index : long) : long
Returns a number of member or a number of group to which the results stored in successive objects of the collection of all the results of given calculations refer.

Get (obj_user_no : long) : IDispatch

Returns an object which stores calculation results for a member or group with the number specified in the obj_user_no parameter. For verification of members or groups this object is the IRDimDetailedRes interface. In case of group desing, this is the IRDimGrpRes interface.

MakeNote (obj_list : IRDimStream , rtf_file : string , out_file : string)

Generates a calculation note for a given list of objects (members or groups). The ‘rtf_file’ parameter contains a full name of an RTF format file being a template, which will be filled out as appropriate according to the obtained calculation results. The template filled out appropriately is saved to a file, whose full name should be contained in the ‘out_file’ parameter.

IRDimDetailedRes

- **Implementation in RDIM module**

The object stores calculation results for one member or superbar.

Attributes:

<GET> MemberUsrNo: long

Returns a number of member for which calculation results are stored.

<GET> ProfileName : string

Returns a section name.

<GET> MaterialName : string

Returns a material name.

<GET> ResOfCalc : IRDimMembCalcRetValue

Returns information about a result of member calculations.

<GET> IsLimitStateUltimate : bool

Returns information about that if calculation results take account of the Ultimate Limit State.

<GET> IsLimitStateService_uy : bool

Returns information about that if calculation results take account of the Serviceability Limit State – member deflection in the direction of y axis (local system).

<GET> IsLimitStateService_uz : bool

Returns information about that if calculation results take account of the Serviceability Limit State – member deflection in the direction of z axis (local system).

<GET> IsLimitStateService_vx : bool
Returns information about that if calculation results take account of the Serviceability Limit State – node displacements in the direction of x axis (global system).

<GET> IsLimitStateService_vy : bool
Returns information about that if calculation results take account of the Serviceability Limit State – node displacements in the direction of y axis (global system).

<GET> Lay : double
Returns a value of slenderness in the y direction for the Ultimate Limit State.

<GET> Laz : double
Returns a value of slenderness in the z direction for the Ultimate Limit State.

<GET> Ratio : double
Returns a value of the ratio 'Ratio' for the Ultimate Limit State.

<GET> GovernCaseName : string
Returns a name of the governing load case for the Ultimate Limit State.

<GET> Uy : double
Returns a value of member deflection in the direction of y axis (local system) – calculations of Serviceability Limit States.

<GET> Uz : double
Returns a value of member deflection in the direction of z axis (local system) – calculations of Serviceability Limit States.

<GET> Vx : double
Returns a value of node displacements in the direction of x axis (global system) – calculations of Ultimate Limit States.

<GET> Vy : double
Returns a value of node displacements in the direction of y axis (global system) – calculations of Ultimate Limit States.

<GET> RelLimit_uy : double
Returns the limit value of member deflection in the direction of y axis (local system) – calculations of Serviceability Limit States.

<GET> RelLimit_uz : double
Returns the limit value of member deflection in the direction of z axis (local system) – calculations of Serviceability Limit States.

<GET> RelLimit_vx : double
Returns the limit value of node displacements in the direction of x axis (global system) – calculations of Serviceability Limit States.

<GET> RelLimit_vy : double
Returns the limit value of node displacements in the direction of y axis (global system) – calculations of Serviceability Limit States.

<GET> GovernCaseName_uy : string
Returns a name of the governing load case for member deflection in the direction of y axis (local system) – calculations of Serviceability Limit States.

<GET> GovernCaseName_uz : string
Returns a name of the governing load case for member deflection in the direction of z axis (local system) – calculations of Serviceability Limit States.

<GET> GovernCaseName_vx : string
Returns a name of the governing load case for node displacements in the direction of x axis (global system) – calculations of Serviceability Limit States.

<GET> GovernCaseName_vy : string
Returns a name of the governing load case for node displacements in the direction of y axis (global system) – calculations of Serviceability Limit States.

<GET> MembDef : IRDimMembDef
Returns an interface storing code parameters of a member.

<GET> MatDef : IRDimMatDef
Returns an interface storing parameters of a member material.

<GET> ProfDef : IRDimProfDef
Returns an interface storing parameters of a member section.

<GET> EffDef : IRDimEffDef
Returns an interface storing information about internal forces at the calculation point.

<GET> CodeResults: IDispatch
Returns an interface which enables getting numeric values of all important code parameters used in the algorithm of member verification according to a given code.

Functions:

GetGovernCase_uy (case_usr_no : long* , comp_index : long*) : long
Returns a number of the governing load case and if it is a composed case, it also returns a number of the component for member deflection in the direction of y axis (local system) - calculations of Serviceability Limit States.

GetGovernCase_uz (case_usr_no : long* , comp_index : long*) : long
Returns a number of the governing load case and, if it is a composed case, it also returns a number of the component for member deflection in the direction of z axis (local system) - calculations of Serviceability Limit States.

GetGovernCase_vx (case_usr_no : long* , comp_index : long*) : long
Returns a number of the governing load case and if it is a composed case, it also returns a number of the component for node displacements in the direction of x axis (global system) - calculations of Serviceability Limit States.

GetGovernCase_vy (case_usr_no : long* , comp_index : long*) : long
Returns a number of the governing load case and if it is a composed case, it also returns a number of the component for node displacements in the direction of y axis (global system) - calculations of Serviceability Limit States.

IRDimGrpResCurrProf

Definition of a set of identifiers for the methods Check and Get of the IRDimGrpRes interface.

Attributes:

I_DGRCP_PREVIOUS: = 1
Indicates a cross section which does not fulfill code conditions any more. It is called the preceding section here.

I_DGRCP_GOVERNING: = 2
Indicates a section which fulfills code conditions. It is called the designing section here.

I_DGRCP_NEXT: = 3
Indicates a cross section which fulfills code conditions, but too large a ratio is obtained.
It is called the next section here.

IRDimGrpRes

- **Implementation in RDIM module**
  The interface is a collection of all calculation results for one member group. This collection is a set of objects. Each such object stores calculation results for a member with a section chosen from the relevant family. For each section family there are maximally three such objects in the collection, for the designing section, for the one preceding it and the next one after it.

Attributes:

<GET> UsrNo : long
Returns a group's number.

<GET> FamiliesCount : long
Returns a number of section families in a collection.

<GET> IsOptimization : bool
Returns information about that if calculation results include optimization results.

<GET> OptFamilyIndex : long
Returns the index of a section family in which an optimal section has been found. It has been a designing section in this family.

Functions:

FamilyName (family_index : long) : string
Returns successive names of section families in a collection.

Check (family_index : long , val : IRDimGrpResCurrProf) : bool
Checks, if there exist appropriate results for the specified family, e.g.
if there are results for a section which is the next after the designing one. They may not exist if the designing section is the last section in a family.

Get (family_index : long , val : IRDimGrpResCurrProf) : IRDimDetailedRes
Returns an object which stores calculation results for the specified section family.
Calculation results are returned for the designing section, for the preceding one (if it exists) or for the next one (if it exists).

4.4 Examples of Application of RDIM Module Interfaces

Below are presented a few examples of application of the RDIM module interfaces presented in the previous chapter. The language used in these examples is Visual Basic, since most probably, this one or a different – similiar to it script language will be used within other applications to communicate with RDIM module.

4.4.1 Generation of a New Label Describing Member Parameters

```
‘....
‘......
‘.........

Dim RobApp As RobotApplication
Set RobApp = CreateObject("Robot.Application")
RobApp.Project.Open(“file name”)

Dim RobStr As IRobotStructure
Set RobStr = RobApp.Project.Structure

Dim RobLab As IRobotLabel
Dim RdmMembDefData As IRDimMembDefData

Set RobLab =
RobStr.Labels.Create(I_LT_MEMBER_TYPE,”example”)
Set RdmMembDefData = RobLab.Data

With RdmMembDefData
  .SetStructureSwayYZ I_DMDBDT_BUCKLING_Y, 1
  .SetLengthYZUV I_DMDLDT_LENGTH_Y, 1#
  .SetStructureSwayYZ I_DMDBDT_BUCKLING_Z, 1
  .SetLengthYZUV I_DMDLDT_LENGTH_Z, 1#
  .SetDeflectionYZ I_DMDDDT_DEFL_Y, 1
  .SetDeflectionYZ I_DMDDDT_DEFL_Z, 1
  .SetDeflYZRelLimit I_DMDDDT_DEFL_Y, 10#
  .SetDeflYZRelLimit I_DMDDDT_DEFL_Z, 10#
  .CantileverMode = 1
  .SetDisplacementXY I_DMDDDT_DISP_X, 1
  .SetDisplacementXY I_DMDDDT_DISP_Y, 1
  .SetDisplXYRelLimit I_DMDDDT_DISP_X, 10#
  .SetDisplXYRelLimit I_DMDDDT_DISP_Y, 10#

  ‘.........
  ‘......
  ‘...

End With

‘...
‘......
‘ addition of code settings
‘......
‘...

RdmMembDefData.type = I_DMDT_USER

RobApp.Structure.Labels.Store Lab

‘.........
‘......
‘...
```

4.4.2 Addition of Code Settings to a Label of Member Parameters

```
‘....
‘......
‘.........

Dim RdmCodeMembDefPar As Object
Set RdmCodeMembDefPar = RdmMembDefData.CodeParams

With RdmCodeMembDefPar
  .BuckLenghtCoeffY = 1#
  .BucklingDiagramY = I_DBD_CM66_NO
  .BuckLenghtCoeffZ = 0.8
  .BucklingDiagramZ = I_DBD_CM66_PINNED_STIFF_0_7
  .LoadTypeY =  I_DLT_CM66_NO_MID_SPAN_FORCE_LOAD
  .LoadTypeZ = I_DLT_CM66_USR_DEF_MOM_MC
  .LoadTypeDistY = 2#
  .LoadTypeMomMcZ = 26#
  .LatBuckType = I_DLBT_CM66_CANTILEVER
  .LoadLevel = I_DLL_CM66_UPP_SECT_PAR_LOADED
  .LoadLevelValue = 1.2
  .LatBuckCoef_LowFlan = I_DLBCD_CM66_USER_DEFINED
  .LatBuckCoef_UppFlan = I_DLBCD_CM66_USER_DEFINED
  .UserDefLatBuckCoef_LowFlan = 0.7
  .UserDefLatBuckCoef_UppFlan = 0.9
  .TensAreaNetGros = 1.1
  .TubeControl = 1

  ‘.........
  ‘......
  ‘...

End With

RdmMembDefData.CodeParams = RdmCodeMembDefPar

‘.........
‘......
‘...
```

4.4.3 Generation of a New Group for the Design Module

```
  ‘....
  ‘......
  ‘.........

  Dim RobApp As RobotApplication
  Dim RDmServer As IRDimServer
  Dim RDmStream As IRDimStream
  Dim RDmGrps As IRDimGroups          ‘ collection of groups
  Dim RDmGrp1 As IRDimGroup           ‘ object of the 1st
group
  Dim RDmGrpProfs As IRDimGrpProfs    ‘ collection of
sections
                                      ‘ for design
  Dim GroupNo As Long                 ‘ group’s number

  Set RobApp = CreateObject("Robot.Application")
  Set RDmServer = RobApp.GetExtension("RDimServer")
  RDmServer.Mode = I_DSM_STEEL
  Set RDmGrps = RDmServer.GroupsService
  GroupNo = 10
  Set RDmGrp1 = RDmGrps.New(0, GroupNo) ‘ creates a new
object
                                        ‘ of the group with
                                        ‘ default settings
  ‘ Set RDmGrp1 = RDmGrps.New(3, GroupNo) - creates a new
  ‘ object of the group with initial settings as in 3rd
group
  RDmGrp1.Material = "Material Name"
  RDmG1.Name = "Group name"
  Set RDmStream = RDmServer.Connection.GetStream
  RDmStream.Clear
  RDmStream.WriteText ("1 2")           'list of members
  RDmGrp1.SetMembList RDmStream
  Set RDmGrpProfs = RDmServer.Connection.GetGrpProfs
  RDmGrpProfs.Clear
  RDmStream.Clear
  RDmStream.WriteText ("IPE")            ' section family
  RDmGrpProfs.SetFamilies "Rcat", RDmStream
  RDmGrp1.SetProfs RDmGrpProfs
  RDmGrps.Save RDmGrp1     ‘ adds a group to group
collection
                           ‘ of RDIM module
  ‘.........
  ‘......
  ‘...
```

4.4.4 Setting of Configuration and Calculation Parameters

```
  ‘....
  ‘......
  ‘.........

  Dim RDmEngine As IRDimCalcEngine
  Set RDmEngine = RDmEngine Server.CalculEngine

  Dim RDmCalPar As IRDimCalcParam
  Dim RDmCalCnf As IRDimCalcConf
  With RdmEngine
    Set RdmCalPar = .GetCalcParam
    Set RdmCalCnf = .GetCalcConf
  End With
  Dim RdmStream As IRDimStream
  Set RdmStream = rdimserver.Connection.GetStream
  RdmStream.Clear
  RdmCalCnf.SetParamValue I_DCCPV_EFFRATIO, 0.9
  ‘.........................  if needed, set different
                              configuration parameters

  RdmStream.Clear
  RdmStream.WriteText (“all”)
  ‘ for verification of members
  RdmCalPar.SetObjsList I_DCPVT_MEMBERS_VERIF, RdmStream
  ‘ or for verification of groups
  RdmCalPar.SetObjsList I_ DCPVT_GROUPS_ VERIF, RdmStream
  ‘ or for design of groups
  RdmCalPar.SetObjsList I_ DCPVT_GROUPS_DESIGN, RdmStream
  RdmCalPar.SetLimitState I_DCPLST_SERVICEABILITY, 1
  RdmCalPar.SetLimitState I_DCPLST_ULTIMATE, 1
  RdmStream.Clear
```

```
   RdmStream.WriteText "1 2"
   RdmCalPar.SetLoadsList rdimStream

   ‘ for design of groups, if need be, optimization
 parameters
     are set,
   Dim RdmOptPar As IRDimOptimParam
   Set RdmOptPar = RdmCalPar. GetOptimParam
   RdmOptPar.Optimization = 1
   RdmOptPar.SetOption I_DOPOT_WEIGHT, 1
   RdmOptPar.SetOption I_DOPOT_MAX_SECTION_HEIGHT, 1
   RdmOptPar.SetLimit I_DOPLT_MAX_SECTION_HEIGHT, 0.5
   RdmOptPar.SetOption I_DOPOT_MIN_SECTION_HEIGHT, 1
   RdmOptPar.SetLimit I_DOPLT_MIN_SECTION_HEIGHT, 0.1
   RdmOptPar.SetOption I_DOPOT_MIN_FLANGE_THICKNESS, 1
   RdmOptPar.SetLimit I_DOPLT_MIN_FLANGE_THICKNESS, 0.01
   RdmOptPar.SetOption I_DOPOT_MIN_WEB_THICKNESS, 1
   RdmOptPar.SetLimit I_DOPLT_MIN_WEB_THICKNESS, 0.05
   RdmOptPar.SetOption I_DOPOT_ENTIRE_SET_SECTIONS, 1
   RdmCalPar. SetOptimParam RdmOptPar

   With RdmEngine
     RdmEngine.SetCalcConf RdmCalCnf
     RdmEngine.SetCalcParam RdmCalPar
   End With

   ‘.........
   ‘......
    ‘...
```

4.4.5 Start of Calculations and Getting a Result Collection

```
   ‘....
   ‘......
   ‘.........

   RdmEngine.Solve Nothing   ' start of calculations

   Dim RdmAllResObjType As IRDimAllResObjectType
   Dim RdmAllRes As IRDimAllRes
   Dim RdmDetRes As IRDimDetailedRes
   Dim ObjCnt As Long
   Dim UsrNo As Long
   Dim Index As Long

   Set RdmAllRes = RdmEngine.Results
   RdmAllResObjType = RdmAllRes.ObjectsType
   ObjCnt = AllRes.ObjectsCount

   Dim Index As Long
   For Index = 0 To ObjCnt - 1
     UsrNo = RdmAllRes.ObjectUsrNo(Index)
     Select Case AllResObjType
       Case I_DAROT_VERIFIED_MEMBER
         Dim RDmDetRes As IRDimDetailedRes
         Set RDmDetRes = RdmAllRes.Get(UsrNo)

         ‘....
         ‘......  RDmDetRes is a collection of verification
```

```
                'results for a member with a number
contained
                'in UsrNo variable
        '.........

      Case I_DAROT_VERIFIED_GROUP
        Dim RDmDetRes As IRDimDetailedRes
        Set RDmDetRes = RdmAllRes.Get(UsrNo)

        '....
        '......  RDmDetRes is a collection of verification
                ' results for a group with a number
contained
                ' in UsrNo variable

        '.........

      Case I_DAROT_ DESIGNED_GROUP
        Dim RDmGrpRes As IRDimGrpRes
        Set RDmGrpRes = RdmAllRes.Get(UsrNo)
        '....
        '......  RDmGrpRes is a collection of design results
               ' for a group with a number contained in
               ' UsrNo variable
        '.........

    End Select

    '.........
    '......
    '...

  Next Index

  '.........
  '......
  '...
```

4.4.6 Getting Calculation Results from a Collection of Verification Results for a Specific Member

```
  '....
  '......
  '.........

  Dim RDmRetValue As IRDimMembCalcRetValue
  Dim RDmDetRes As IRDimDetailedRes
  Set RDmDetRes = RdmAllRes.Get(1)   ' for member with
number 1

  RDmRetValue = RdmDetRes.ResOfCalc
  '..
  '.....
  ........... = RdmDetRes.ProfileName
  ........... = RdmDetRes.MaterialName

  If RdmDetRes.IsLimitStateUltimate Then
     ........... = RdmDetRes.Lay
     ........... = RdmDetRes.Laz
     ........... = RdmDetRes.Ratio
```

```
      ........... = RdmDetRes. GovernCaseName
   End If
   If RdmDetRes.IsLimitStateService_uy Then
     ........... = RdmDetRes.Uy
     ........... = RdmDetRes.RelLimit_uy
     ........... = RdmDetRes.GovernCaseName_uy
   End If

   If RdmDetRes.IsLimitStateService_uz Then
     ........... = RdmDetRes.Uz
     ........... = RdmDetRes.RelLimit_uz
     ........... = RdmDetRes.GovernCaseName_uz
   End If
   If RdmDetRes.IsLimitStateService_vx Then
     ........... = RdmDetRes.Vx
     ........... = RdmDetRes.RelLimit_vx
     ........... = RdmDetRes.GovernCaseName_vx
   End If
   If RdmDetRes.IsLimitStateService_vy Then
     ........... = RdmDetRes.Vy
     ........... = RdmDetRes.RelLimit_vy
     ........... = RdmDetRes.GovernCaseName_vy
   End If
   ‘.........
   ‘......
   Dim RdmCodeRes As Object
   Set RdmCodeRes = RdmDetRes. CodeResults
   ‘...
   ‘ getting detailed – code-determined results of
 calculations
   ‘.........
   ‘......
   ‘...
   Dim RDmMembDef As IRDimMembDef
   Dim RDmMatDef As IRDimMatDef
   Dim RDmProfDef As IRDimProfDef
   Dim RDmEffDef As IRDimEffDef

   Set RDmMembDef = RdmDetRes.MembDef ‘ member code
 parameters
   Set RDmMatDef = RdmDetRes.MatDef    ‘ material parameters
   Set RDmProfDef = RdmDetRes.ProfDef ‘ member section
                                      ‘ parameters
   Set RDmEffDef = RdmDetRes.EffDef   ‘ loads at the point
                                      ‘ for which results
 have
                                      ‘ been obtained
   ‘.........
   ‘......
   ‘...
```

4.4.7 Getting Calculation Results from a Collection of Verification Results for a Specific Member Group

```
   ‘....
   ‘......
   ‘.........

   Dim UsrNo As Long
   Dim RDmRetValue As IRDimMembCalcRetValue
```

```
  Dim RDmDetRes As IRDimDetailedRes

  Set RDmDetRes = RdmAllRes.Get(3)    ‘ for group with number
3

  UsrNo = RdmDetRes.MemberUsrNo       ‘ number of verified
                                      ‘ member
  ‘....
  ‘......
  ‘.........
  ‘ getting calculation results - as for a collection of
  ‘ verification results for a member
  ‘.........
  ‘......
  ‘...
```

4.4.8 Getting Calculation Results from a Collection of Design Results for a Member Group

```
  ‘....
  ‘......
  ‘.........

  Dim RDmGrpRes As IRDimGrpRes
  Set RDmGrpRes = RdmAllRes.Get(2)    ‘ for group with number
2

  Dim RDmDetRes As IRDimDetailedRes
  Dim PrfFamCnt As Long
  Dim Index As Long
  Dim OptFamIndex As Long
  PrfFamCnt = RDmGrpRes.FamiliesCount
  For Index = 0 To PrfFamCnt – 1
    ‘....
    ‘......
    ‘.........
    If Index = 0 And RDmGrpRes.IsOptimization Then
       OptFamIndex = RDmGrpRes.OptFamilyIndex
    End If
    ‘.........
    ‘......
    ‘...
    If RDmGrpRes.Check(Index, I_DGRCP_PREVIOUS) Then
       Set DetRes = RDmGrpRes.Get(Index, I_DGRCP_PREVIOUS)
       ‘....
       ‘......
       ‘.........
       ‘ getting calculation results for the section which
       ‘ preceds the section designing a given family
       ‘ - as for a collection of member verification
results
       ‘.........
       ‘......
       ‘...
    End If
    If RDmGrpRes.Check(Index, I_DGRCP_GOVERNING) Then
       Set DetRes = RDmGrpRes.Get(Index, I_DGRCP_GOVERNING)
       ‘....
       ‘......
       ‘.........
       'getting calculation results for the section
designing
       'a given family - as a for a collection of member
       'verification results
       '.........
       '......
       '...
       If RDmGrpRes.IsOptimization And Index = OptFamIndex
Then
          '.........
          ' section is optimal
          '.........
       End If
       '.........
       '......
       '...
    End If
    If RDmGrpRes.Check(Index, I_DGRCP_NEXT) Then
       Set DetRes = RDmGrpRes.Get(Index, I_DGRCP_NEXT)
       '....
       '......
       '.........
       'getting calculation results for the section which is
       'the next after the section designing a given family
-
       'as for a collection of member verification results
       '.........
       '......
       '...
    End If
       '.........
       '......
       '...
  Next PrfFamIndex

  '.........
  '......
  '...
```

4.4.9 Getting Detailed Code-Determined Calculation Results

```
  '....
  '......
  '.........

  Dim RdmCodeRes As Object
  Set RdmCodeRes = RdmDetRes. CodeResults

  ' for LRFD code
  ........... = RdmCodeRes.BuckSlendY
  ........... = RdmCodeRes.BuckSlendZ
  ........... = RdmCodeRes.BuckColSlendParamLamcY
  ........... = RdmCodeRes.BuckColSlendParamLamcZ
  ........... = RdmCodeRes.SecFlangeClass
  ........... = RdmCodeRes.SecWebClass
  ........... = RdmCodeRes.SecUppUppFlangeLocSlendX1
  ........... = RdmCodeRes.SecLowUppFlangeLocSlendX2
  ........... = RdmCodeRes.SecUppFlangeWidthB
  ........... = RdmCodeRes.SecFlangeDepthInMidSpanB1
  ........... = RdmCodeRes.SecLowFlangeDepthB2
```

```
........... = RdmCodeRes.SecHightD
........... = RdmCodeRes.SecUppFlangeThickT
........... = RdmCodeRes.SecUppFlangeThickT2
........... = RdmCodeRes.SecWebHightH
........... = RdmCodeRes.SecHightAtMembBegH1
........... = RdmCodeRes.SecHightAtMembBegH2
........... = RdmCodeRes.SecWebThickTW
........... = RdmCodeRes.SecFlangeLocSlendX
........... = RdmCodeRes.SecWebLocSlendX
........... = RdmCodeRes.LimWidthThickParamKC
........... = RdmCodeRes.RedFactForSlendCompElemQ
........... = RdmCodeRes.LatBuckUnbracedLengthLB
........... = RdmCodeRes.LatBuckBendCoefCB
........... = RdmCodeRes.LatBuckLimPlastUnbracedLengthLpY
........... = RdmCodeRes.LatBuckLimPlastUnbracedLengthLpZ
........... = RdmCodeRes.LatBuckLimUnbracedLengthLrY
........... = RdmCodeRes.LatBuckLimUnbracedLengthLrZ
........... = RdmCodeRes.LatBuckLimMomentMrY
........... = RdmCodeRes.LatBuckLimMomentMrZ
........... = RdmCodeRes.StrengthAxTensPT
........... = RdmCodeRes.StrengthAxCompPA
........... = RdmCodeRes.StrengthAxTensCompPN
........... = RdmCodeRes.StrengthShearVnY
........... = RdmCodeRes.StrengthShearVnZ
........... = RdmCodeRes.StrengthFlexMnY
........... = RdmCodeRes.StrengthFlexMnZ
........... = RdmCodeRes.PlastBendMomMpY
........... = RdmCodeRes.PlastBendMomMpZ
........... = RdmCodeRes.CritElastMomMcrY
........... = RdmCodeRes.CritElastMomMcrZ
........... = RdmCodeRes.CompResidulStressFR
........... = RdmCodeRes.CritStressFCR
........... = RdmCodeRes.SecPlastModulusWplY
........... = RdmCodeRes.SecPlastModulusWplZ
........... = RdmCodeRes.SecElastModulusWelY
........... = RdmCodeRes.SecElastModulusWelZ
........... = RdmCodeRes.SecCompElastModulusWelYC
........... = RdmCodeRes.SecTensElastModulusWelYC
........... = RdmCodeRes.MaxEffRatio
........... = RdmCodeRes.EffShearRatioY
........... = RdmCodeRes.EffShearRatioZ
‘.........
‘......
‘...
```

The examples presented show precisely how the process of communication between an application and RDIM module should look like. Application of other interfaces used to get detailed code-determined calculation results, except for that shown in the example, has not been presented here and the process of getting these results has been demonstrated on the basis of the interface for LRFD code. All other interfaces of code-determined results differ only in names of attributes, which can be seen when viewing the contents of the RobotOM library. Names of attributes correspond to names of code parameters used in the algorithm of member verification according to a given code.