**Steel Connections**

**COM API**

**Reference Guide**

**Contents**

[Chapter 1 Introduction 7](#_Toc117525325)

[1.1 Joints 8](#_Toc117525326)

[1.1.1 JOINT summary 8](#_Toc117525327)

[Chapter 2 Joints 9](#_Toc117525328)

[2.1 Key concepts 10](#_Toc117525329)

[2.1.1 Rule 10](#_Toc117525330)

[2.2 Architecture 10](#_Toc117525331)

[2.2.1 Libraries 10](#_Toc117525332)

[2.2.2 Attributes and categories 11](#_Toc117525333)

[2.2.3 Roles 11](#_Toc117525334)

[2.2.4 User interface 11](#_Toc117525335)

[2.2.5 Query method 11](#_Toc117525336)

[2.2.6 CreateObjects method 11](#_Toc117525337)

[2.2.7 Attribute transfer 11](#_Toc117525338)

[2.2.8 InField/OutField methods 11](#_Toc117525339)

[2.2.9 GetUserPages method 11](#_Toc117525340)

[2.2.10 GetTableName 12](#_Toc117525341)

[2.2.11 Prompts and their numeric identifiers 12](#_Toc117525342)

[2.2.12 Database Structure 12](#_Toc117525343)

[2.3 Comments and Tips 12](#_Toc117525344)

[Chapter 3 Developing a joint 13](#_Toc117525345)

[3.1 Joint analysis 14](#_Toc117525346)

[3.1.1 Attributes 14](#_Toc117525347)

[3.1.2 Objects 14](#_Toc117525348)

[3.2 Coding 14](#_Toc117525349)

[3.3 Beam Transformations 20](#_Toc117525350)

[Chapter 4 API Reference 23](#_Toc117525351)

[4.1 Geometry 24](#_Toc117525352)

[4.1.1 Point3d 24](#_Toc117525353)

[4.1.2 Point3dArray 24](#_Toc117525354)

[4.1.3 Vector3d 25](#_Toc117525355)

[4.1.4 Vector3dArray 26](#_Toc117525356)

[4.1.5 Line3d 26](#_Toc117525357)

[4.1.6 LineSeg3d 26](#_Toc117525358)

[4.1.7 LineSeg3dArray 27](#_Toc117525359)

[4.1.8 Curve3d 27](#_Toc117525360)

[4.1.9 Arc3d 27](#_Toc117525361)

[4.1.10 Plane 28](#_Toc117525362)

[4.1.11 Matrix3d 29](#_Toc117525363)

[4.1.12 Matrix3dArray 29](#_Toc117525364)

[4.1.13 AugPolyline3d 29](#_Toc117525365)

[4.1.14 AugPolylinesArray 30](#_Toc117525366)

[4.1.15 AugPolygon3d 31](#_Toc117525367)

[4.1.16 AugPolygonsArray 32](#_Toc117525368)

[4.1.17 AugPolygonsArrayArray 32](#_Toc117525369)

[4.1.18 CS3d 33](#_Toc117525370)

[4.1.19 CS3dArray 33](#_Toc117525371)

[4.1.20 Extents 34](#_Toc117525372)

[4.1.21 PlaneArray 34](#_Toc117525373)

[4.1.22 VertexInfo 34](#_Toc117525374)

[4.1.23 VertexInfoArray 35](#_Toc117525375)

[4.1.24 GeomUtils 35](#_Toc117525376)

[4.2 Profiles access 35](#_Toc117525377)

[4.2.1 ProfType 35](#_Toc117525378)

[4.2.2 ProfSectVal 38](#_Toc117525379)

[4.2.3 ProfGapCollection 39](#_Toc117525380)

[4.3 Util facet 39](#_Toc117525381)

[4.3.1 AstModeler 39](#_Toc117525382)

[4.4 Database access 40](#_Toc117525383)

[4.4.1 OdbcUtils 40](#_Toc117525384)

[4.4.2 OdbcTable 40](#_Toc117525385)

[4.5 SteelAutomation 41](#_Toc117525386)

[4.5.1 AstObject 41](#_Toc117525387)

[4.5.2 EqualPartObject 42](#_Toc117525388)

[4.5.3 AstFeatObject 43](#_Toc117525389)

[4.5.4 PlateFeat 43](#_Toc117525390)

[4.5.5 PlateFeatVertex 43](#_Toc117525391)

[4.5.6 PlateFeatEdge – 43](#_Toc117525392)

[4.5.7 PlateFeatContour 43](#_Toc117525393)

[4.5.8 Chamfer 44](#_Toc117525394)

[4.5.9 Fillet 45](#_Toc117525395)

[4.5.10 PlateContourNotch 45](#_Toc117525396)

[4.5.11 BeamNotch 46](#_Toc117525397)

[4.5.12 BeamNotchStd 47](#_Toc117525398)

[4.5.13 BeamNotchEx 47](#_Toc117525399)

[4.5.14 BeamContourNotch 47](#_Toc117525400)

[4.5.15 BeamMultiContourNotch 48](#_Toc117525401)

[4.5.16 AstFeaturesArray 49](#_Toc117525402)

[4.5.17 BeamShortening 49](#_Toc117525403)

[4.5.18 Beam 50](#_Toc117525404)

[4.5.19 StraightBeam 57](#_Toc117525405)

[4.5.20 UnfoldedBeam 58](#_Toc117525406)

[4.5.21 BentBeam 58](#_Toc117525407)

[4.5.22 PolyBeam 59](#_Toc117525408)

[**Note**: This only works in Advance Steel. 60](#_Toc117525409)

[4.5.23 Plate 60](#_Toc117525410)

[4.5.24 PlateFolded 63](#_Toc117525411)

[4.5.25 PlateFoldRelation 64](#_Toc117525412)

[4.5.26 Bolt 65](#_Toc117525413)

[4.5.27 Weld 66](#_Toc117525414)

[4.5.28 StudShear 67](#_Toc117525415)

[4.5.29 Hole 67](#_Toc117525416)

[4.5.30 SpecialPart 68](#_Toc117525417)

[4.5.31 BoltHoleDefinition 69](#_Toc117525418)

[4.5.32 AstUI 69](#_Toc117525419)

[4.5.33 RulePage 78](#_Toc117525420)

[4.5.34 RulePageArray 78](#_Toc117525421)

[4.5.35 PropertySheetData 78](#_Toc117525422)

[4.5.36 HRLJoint 78](#_Toc117525423)

[4.5.37 Joint 79](#_Toc117525424)

[4.5.38 Rule 83](#_Toc117525425)

[4.5.39 Filer 84](#_Toc117525426)

[4.5.40 FilerItem 85](#_Toc117525427)

[4.5.41 DbCursor 85](#_Toc117525428)

[4.5.42 Role 85](#_Toc117525429)

[4.5.43 AstObjectsArr 86](#_Toc117525430)

[4.5.44 ClassFilter 86](#_Toc117525431)

[4.5.45 AstColor 87](#_Toc117525432)

[4.5.46 JointTransfer 87](#_Toc117525433)

[4.5.47 AstCreator 88](#_Toc117525434)

[Chapter 6 Appendix – Common data and types 89](#_Toc117525435)

[5.1 Profiles Access data 90](#_Toc117525436)

[5.2 SteelAutomation data 92](#_Toc117525437)

[5.3 Util Facet data 103](#_Toc117525438)

[5.4 Examples 107](#_Toc117525439)

[5.4.1 Calling a subrule 107](#_Toc117525440)

[Chapter 7 AstControls library 108](#_Toc117525441)

[6.1 AstUnitControl 109](#_Toc117525442)

[6.2 AstComboTableControl 109](#_Toc117525443)

[6.3 AstCheckBoxControl 110](#_Toc117525444)

[6.4 AstProfileControl 110](#_Toc117525445)

# Chapter 1 Introduction

## Joints

In Steel Connections, you can create your own modeling customizations and connections (referred to as **joints**) using a programming language which can implement COM.

This document describes the available Steel Connections COM objects and their usable properties and methods for joint implementation.

### JOINT summary

A *joint* is created through an object which implements the [**IRule**](#_Rule) interface and uses the [**IJoint**](#_Joint) interface to create steel Connection elements.

**Joint workflow**:

Input definition (*Query*) - creates objects (*CreateObjects*) - displays dialog box (*GetUserPages*).

Through a joint, the created objects depend on the selected input objects, as long as the joint object exists in the model. The created objects or their properties are modifiable at any time using the joint properties dialog box.

The *IRule* interface *Query* method is called when a joint is created. In this method you can select the input objects for your joint and initialize default parameters.

The *IRule* interface *CreateObjects* method must implement the logic of your connection.

The *IRule* interface *GetUserPages* is called when the joint properties window is opened.

The *IJointInfo* interface properties are interrogated to get information about the developer.

You need to add a record in **AstorRules**.*HRLDefinition* and **AstorRules**.*RulesDllSigned* tables after developing a joint to be able to be executed by Steel Connections in Revit.

# Chapter 2 Joints

## Key concepts

### Rule

A rule is a coclass which implements the *IRule* interface defined by the *AstSTEELAUTOMATIONLib* library. The rule must be implemented in an inprocess server.

## Architecture

Steel Connections creates and modifies joints by using rules. The *IRule* interface defines several methods/properties, each being responsible for a certain task.

The *joint* provides the link of *IRule* with the underlying *joint* object of Steel Connections. The joint object is set by Steel Connections before calling any method of the *Rule.*

The *Query* method describes the (input) parameters of the joint. Its implementation should ask the user for input with the help of *IAstUI* and add the necessary entities to the *InputObjects* of the *joint.*

The *CreateObjects* method describes the steps of the construction. Its implementation should create steel connection objects and add them to the *CreatedObjects* of the *joint.*

Persistency for the rule parameters is maintained with the use of *IAstFiler*.

*Rules* can be written by anybody and the Steel Connections framework is able to execute them at runtime. Thus, several services are provided - creation, modification, explode, and deletion of a joint object:

* A joint object is created along with its driven *objects*. A dialog box appears displaying all the parameters which apply to that joint*.*
* Editing a joint object leads to the appropriate dialog box.
* Exploding an object means deleting the ‘logical joint unit’ while maintaining the driven objects. These objects will live explicitly then and will not know anything about the joint anymore.
* Deleting an object means deleting the logical unit and the driven objects as well. This includes features at driving objects, e.g. beam notches, etc.

### Libraries

Steel Connections provides several libraries that are intended to be used when implementing a rule ( in C++ or C#):

* **AstSteelAutomation** / **Interop.AstSTEELAUTOMATIONLib5** library – Provides access to all Steel Connections model objects.
* **DSCGeomCom** / **Interop.DSCGEOMCOMLib** library – Geometry library, useful for geometric calculation (vectors, coordinates, points, parametric curves).
* **DSCProfilesAccesCom** / **Interop.DSCPROFILESACCESSCOMLib** library – Provides access to the defined profiles used by Steel Connections.
* **DSCUtilFacetCom** / **Interop.DSCUTILFACETCOMLib** library - Provides access to the geometric body of a Steel Connections object.
* **DSCOdbcCom** / **Interop.DSCODBCCOMLib** library - Provides access to data used by Steel Connections and stored in external databases.
* **AstControls** / **Interop.ASTCONTROLSLib** library – Provides several controls intended to be used for the GUI of the joints.

Mainly, this library has regular controls but uses database prompts (language dependent) instead of hardcoded prompts. Also, this library has a Bitmap control that uses bitmaps stored in **AstorBitmaps** database or in .dlls.

### Attributes and categories

### Roles

Whenever you need to create a Steel Connections object you should define a new *IRole* object. Using role objects you can customize groups of objects to have similar properties. See the [Role](#_Role) object.

### User interface

The joint with its attributes is controlled by a property sheet dialog box. This means, that this dialog box will appear during modification and will expose all the joint attributes.

The rule designer has to design all the other pages. This is done by implementing the *GetUserPages* method of the *IRule*.

### Query method

The *Query* method implementation should use the provided *AstUI* object and get user input regarding necessary parameters or objects. The objects must be added to the *Joint.InputObjects*. The selected objects will become *driver objects* for the joint. Modifying of the *driver objects* will prompt an update of the joint.

### CreateObjects method

The *CreateObjects* method implementation should create Steel Connections objects and add them to the *CreatedObjects* of the *Joint.* The objects can be created by using the methods of the Joint object. Once the objects are created they are automatically added to the model. The connection between them and the Joint object is done after they are added to *Joint.CreatedObjects.* Although it is possible to create objects without adding them to *Joint.CreatedObjects,* it is not advised since, in this case, the connection is not created and these object are “out of joint”. They will not be erased when updating the joint and other objects will be created.

### Attribute transfer

It refers to the joint capabilities to keep the properties of the objects created by the joints when updating the joint.

### InField/OutField methods

These methods are intended to provide a way for Steel Connections to retrieve/set persistent data. Steel Connections will call the *OutField* method every time it’s necessary to get the values and/or the names of the Rule attributes. Similarly Steel Connections will call the *InField* method every time it’s necessary for the Rule to change its current attribute values. For example, calls to these methods will happen when a model containing a joint created with this Rule is opened/saved, when you change the joint parameters in the joint properties dialog box, etc.

### GetUserPages method

By implementing this method you are able to specify the GUI of this Rule. This method should return handles of the created windows. The created windows will become property pages for the Rule dialog box. Also, here you should assign the corresponding prompts for the Rule dialog box title and for the PropertyPages titles.

### GetTableName

Returns the name of the table used by the rule.

### Prompts and their numeric identifiers

All the rules use prompt numbers referring to localized texts.These texts should be added to the prompts database, see *Database Structure* chapter.

### Database Structure

The following databases are used for joint development.

**AstCrtlDb.mdf** - used to store the localized texts that a joint uses to display in its dialog box pages.

**AstorRules.mdf** - used to store the joint "definition" records (*HRLDefinition* and *RulesDllSigned* tables) and to store any table the joint uses.

The definition record in *HRLDefinition* requires the specification of the coclass GUID on the ClassID column.

## Comments and Tips

This section covers specific topics which may be useful for practical work.

**Help system of the joint**

Each page of the joint can have a helpIndex. This helpIndex (is an integer) is mapped through a table from **AstorBase** database(**HelpButtons** table) using a string formatted as follows:

Rule\_RULENAME\_XXX1\_XXX2

Where:

* RULENAME is the name of the rule in question.
* XXX1 is the propertySheet prompt number.
* XXX2 is the propertyPage prompt number.

**Helpful Tips**

You are only using interfaces to objects and not objects. Whenever you need to create a Steel Connection object you should use the Rule object.

There can be two interfaces pointing to the same object. By changing one object using an interface you will also change the object beneath the other interface. This can happen because the interfaces are pointing to the same Steel Connections object. In order to avoid this, the functions for geometry interfaces that “copy” an object to another one were exposed.

**Debugging a joint**

A joint is created and kept in a .dll.

The release version of this .dll should be added in the **AddIns\SteelConnections** folder (Ex: C:\Program Files\Autodesk\Revit 2025\AddIns\SteelConnections\YourDllName.dll for Revit 2025) of your Revit installation.

Revit runs on two processes (Revit.exe and RevitWorker.exe). The joint creation in done only by RevitWorker.exe, so to debug your joint, attach it to the RevitWorker.exe.

Now you can start Visual Studio in debug mode, insert a breakpoint in the *Query* method the joint entry point.

# Chapter 3 Developing a joint

The detailed development process consists of the following:

## Joint analysis

An existing joint must be analyzed to know what kind of elements make up the joint,which elements are parametrical, which distances are fixed, etc.

### Attributes

The first essential question to answer is: Which are the attributes of the joint?

It must be possible to calculate / to develop the joint out of these attributes.

The attributes include the elements that respect the following:

* They exist before the joint is created.
* They deliver data necessary to calculate the joint (e.g., height of a beam) or they will get modified by the joint (i.e., by a feature).

**Note**: Elements connected by a joint which are not attributes of this joint may exist.

A joint can be calculated in different ways.For example, a base plate can be calculated based on the column cross section extents OR based on the (explicitly given) attributes length and width.

Keep in mind, that the user will be able to modify the joint by modifying the joint attributes, including the ‘element attributes’. The user will not be able to modify the joint any other way (except by exploding it).

### Objects

Identify the objects to create in a joint. This includes feature objects.

Are there any connections to create?

Which (Steel Connections) attributes of these objects are fixed, which ones are to be calculated, which ones are free (i.e. can be changed independently of the joint creation rule)?

## Coding

A Rule can be developed in any language that supports COM. The following is a description of how to create a joint using C#:

1. Start Visual Studio 2022 and choose Class Library as a working project.
2. Add the following assembly references to the project:

* **Interop.AstSTEELAUTOMATIONLib5.dll** library
* **Interop.DSCGEOMCOMLib.dll** library
* **Interop.DSCPROFILESACCESSCOMLib.dll** library
* **Interop.DSCUTILFACETCOMLib.dll** library
* **Interop.DSCODBCCOMLib.dll** library
* **Interop.ASTCONTROLSLib.dll** library

Each of these libraries has a well-defined role in the joint development process and will be described in the following pharagraphs. Also, for more information, there will be context sensitive help (available on F1) that will describe in detail each function from each library.

These libraries are described as follows:

* **AstSTEELAUTOMATIONLib5** handles all the “core” Steel Connections functionalities. It exposes interfaces to objects that can be created with Steel Connections and also handles joint, GUI and other functionalities.
* **DSCGEOMCOMLib** handles basic geometry classes. It exposes interfaces like point, vector, plane, etc.
* **DSCPROFILESACCESSCOMLib** is used to handle *ProfType* objects (beam profiles, user defined and general use profiles).
* **DSCUTILFACETCOMLib** mainly handles the body interface. It provides access to body intersection with lines and other basic geometry interfaces.
* **DSCODBCCOMLib** provides useful and easy access to Steel Connection databases. You can get prompts, defaults, or easily access tables through interfaces provided by this library.
* **ASTCONTROLSLib** is used mainly for GUI development of joints. Several ActiveX controls are provided with this library. For example, the joint developer can link one static prompt to an **AstCrtlDb.mdf** message through a *StaticDbTextControl*.
* At this stage of the project there should be only one class (named “Class1” by default ). This can be renamed to fit the name of the joint.
* You should write the class object in a way that it implements the *IRule* interface in order to have access to basic functionalities of a Rule:

public class CreateBeams : *IRule*

* There should also be a *Joint* object declared in the class:

pprivate Joint m\_Joint = null;

private Joint m\_Joint = null**;**

* This Joint object is automatically initialized by exposing it in a get/set property called Joint.

Please note that all future data members of the joint need to be declared globally in the class.

* Here are all the function names needed to implement a joint. Use the Visual Studio auto-complete functionality to create function bodies accordingly. They are as follows:
* **Query**
* **CreateObjects**
* **GetUserPages**
* **GetExportData**
* **GetTableName**
* **InField**
* **OutField**
* Joint [Property **get**]
* Joint [Property **set**]

**Query** is the function that is passed only once when the joint is run the first time. Its role is to ask the user for input objects and set defaults.

In this function it is mandatory to create, use and return an *InputObjects* array, as follows:



**CreateObjects** method contains the joint functionality. It uses the global variables declared in the declaration section and does the main work.

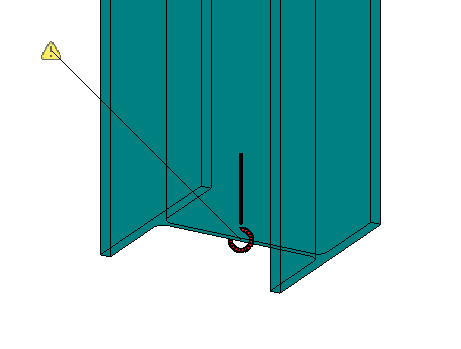
In this function it is also mandatory to create, use and return a CreatedObjects array, as follows:



Please create objects in this method at the first call of this method, do not insert tests that will allow objec creation only at a second execution of this method (e.g. do not create the joint objects only after the **GetUserPages** method is executed).

The normal behavior when creating a new joint is to create objects in the "*CreateObjects*" method and show the property pages. Remember - when you show property pages you should not force the update of the joint.

If the joint creation/update fails, you will be notified through a warning message and on the screen you will see only a connection handler object pointing out that there is a connection instead of the actual joint elements.



The joint should implement the CreateObjects method as follows( in C++):



C# sample:



**GetUserPages** provides the link between the joint C# Library class and GUI Forms.

You should keep the form objects inside the created Joint class as follows:

The joint developer is responsible for creating GUI forms for the joint. Each .NET form represents one property sheet in the Joint Properties tab. After the forms are ready the joint developer should do the following:

In the section declaration of the joint class, declare members of type Forms. For example:

Where m\_*Page1* is the name of the object and *Page1* is the name of the form class.

Declare an object for each created form.

In the declaration section of each form, declare a global variable (global for the current form) of type class1 (name of the Library class).

This variable should provide the link between the form and the Library class object.

In the *GetUserPages* function, write the following instruction for each form:

**Similar to the above code the joint, developer can create several pages for a joint.

In the end there should be a call for:



The parameter structure *AstSTEELAUTOMATIONLib*.*IPropertySheetData* pPropSheetData should contain the main prompt of the sheets, the bitmap index for the first property page, and the property pages size option.



The next two functions set and get the joint and they should contain:

**Joint [Property get]**

****

**Joint [Property set]**

*InField* should contain persistent data reading from the model.

In the InField function there should be calls to pFiler.readItem like:



Please note that the “Position” string is important and should be exactly the same in the reverse operation, *Outfield*:



*GetTableName* – this function should return the name of the table used by the rule to run.

The joint must also implement the *IJointInfo* interface (contains information about the developer). If this interface is not implemented the joint will fail.

This interface defines the next properties:

* + *Name* **[Property Get]**
  + *Email* **[Property Get]**
  + *CompName* **[Property Get]**
  + *ASVersion* **[Property Get]**
  + *VersionNumber* **[Property Get]**
  + *ApplicationName* **[Property Get]**
  + *LogoIconId* **[Property Get]**

Example:



## Beam Transformations

This section describes the beam transformations needed to get a beam from its profile local - 2D coordinate system to WCS. These transformations apply mainly to straight beams. Although the general principle is the same for all beams, it is obvious that the CS orientations and some transformation will be slightly different in the case of curved beams, poly beams, and so on.

**Start with an** *AstProfType* **(***IProfType***).**

* This is defined in the xOy plane, around the origin.
* The origin is the future location of the kSysSys reference axis.

**In order to get a beam from a** *ProfType* **do the following basic transformations:**

* Get the profile in the yOz plane.
* Extrude along – kXAxis (for straight beams, for curved beams, poly beams the extrusion is slightly more complicated).
* Transform the created beam in WCS.

More detailed:

* Start with a 2D xOy profile contour obtained directly from the *ProfType.*
* Transform with **“*GetMatSys2Phys2d*”**. Doing this, the profile refAxis is set as requested by the user (from the properties dialog box, or programmatically), apply the rod rotation (this is not the one which the user can set from the dialog box, that one is reflected directly in the ***getSysCSStart*** transformation) and transform with a translation matrix based on the rod offsets (***BeamOffsetX*** and ***BeamOffsetY***). Note that the offsets are represented by a 2D vector, and are applied to the profile contour which is still in 2D space.
* Transform by **“*getMatSysAsXY2Sys*”**. This matrix transforms the contour from the xOy plane with z normal into the yOz plane with x normal – this is the moment when the beam extrusion is performed.
* Transform by **“*getSysCSStart*”** which moves the beam to its final WCS position.

**Additional notes:**

* **Naming convention:** before the “***getSysCS***” transformation the beam is in its ‘local CS’**,** afterwards the beam is in WCS.
* All methods retrieving beams CS will return CS having the following property: the x-axis of each CS is oriented from the end towards the start of the beam (regardless of offsets, rotations, mirror, etc.).
* The **“*GetMatProf2WCS*”, “*GetMatProf2WCSMidCS*”, “*GetMatProf2WCSEndCS*”** methods will return the transformations which will transform the 2D contour obtained from ProfType from xOy, in a 3D position overlapping the physical beam body at start, mid, and end, respectively. These transformations include all the steps described above at point.
* The **“*getSysCSStart*”, “*getSysCSMid*”, “*getSysCSEnd*”** methods will return CS along the beam system line (this is the transformation corresponding to the last step).
* The **“*getPhysCSStart*”, “*getPhysCSStartMid*”, “*getPhysCSStartEnd*”** methods will return the transformations which will transform a 2D contour (obtained from ProfType) from the yOz position, in a 3D position overlapping the physical beam body at start, mid, and end, respectively. As you can see, these transformations are not part of the normal beam evolution process, but they may be useful in a number of situations and they also exist for historical reasons.
* The **“*GetCurrentSys2Phys3dOffset*”** returns the transformation in WCS between *getSysCSStart* and *getPhysCSStart*. This is the recommended way of getting this “offset”, because it takes into account all factors correctly (ref axes, rotations, offsets, and mirror).
* The first two steps above can also be obtained by only one transformation: ***GetMatProfToLocal***. So ***GetMatProf2Local* = *getMatSysAsXY2Sys* \* *GetMatSys2Phys2d***.

**Mirroring:**

* Mirroring only affects the last step of the beam evolution.
* When a beam is mirrored, the results of the “*getCS*” methods (sys or phys or prof2WCS) will be **left-handed** CS. The basic rule is that any CS of the mirrored beam could be calculated by just mirroring the corresponding CS of the original un-mirrored beam.
* Given this, never make the assumption that CS.x.crossproduct(CS.y) == CS.z, where CS is any of the beam CS obtained with the above explained methods.

# Chapter 4 API Reference

## Geometry

These are the basic geometric objects. In any joint that will be developed, you will have to use at least a few of these objects (***Point3d***, ***Vector3d***, ***Plane***, etc.). Below you will find a description for all of the objects and the methods and properties which they have, along with some code samples that show their usage.

### Point3d

|  |  |
| --- | --- |
| **Point3d – methods and properties** | |
| **X** (property) | Gets or sets the x coordinate of the point. |
| **Y** (property) | Gets or sets the y coordinate of the point. |
| **Z** (property) | Gets or sets the z coordinate of the point. |
| **DistanceTo( Point3d** toPoint **)** | Returns the distance from this point to the specified point. |
| **Subtract ( Point3d** subPoint **)** | Returns a vector, oriented from subPoint to this point: Ex: *p1.Subtract(p2)* – the vector is oriented from p2 to p1 |
| **Create ( double** dX, **double** dY, **double** dZ **)** | Creates the point with specified coordinates. |
| **Add ( Vector3d** vectorToAdd **)** | Adds the specified vector to this point. |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this point. |
| **Project ( Plane** projPlane, **Vector3d** projDir **)** | Projects the point on the specified plane, along the direction given by the vector. |
| **orthoProject ( Plane** projPlane **)** | Orthogonal projection of the point on the specified plane. |
| **getPointOnAxis ( Vector3d** axisDir, **Point3d** axisPoint **)** | Projects the point on the specified axis (by point and vector) and **returns** the resulted point. |
| **setFrom ( Point3d** asPoint **)** | Copies the coordinates of asPoint into this point. |
| **setFromVector ( Vector3d** asVector **)** | Similar to *setFrom*(). Instead of taking the coordinates from a point, it creates a point from x, y, z directions of the vector. |
| **IsEqualTo ( Point3d** ptIn**, double** dPrecision **)** | Checks with the specified precision if a specified point is the same as the current point. **Returns *True*** or ***False***. |

### Point3dArray

|  |  |
| --- | --- |
| **Point3dArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | Returns the point at the specified index in the array. |
| **Count ( )** (property – get) | Returns the number of elements in the array. |
| **Add ( Point3d** pointToAdd **)** | Appends a new point in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### Vector3d

|  |  |
| --- | --- |
| **Vector3d – methods and properties** | |
| **X** (property) | Gets or sets the x dimension of the vector. |
| **Y** (property) | Gets or sets the y dimension of the vector. |
| **Z** (property) | Gets or sets the z dimension of the vector. |
| **Length ( )** (property – get) | Returns the length of the vector. |
| **DotProduct ( Vector3d** withVector **)** | Returns the result of the scalar product of this vector and the specified vector. |
| **Add ( Vector3d** vectToAdd **)** | Returns the vector after adding the specified vector to this vector. |
| **Subtract ( Vector3d** toSubtract **)** | Returns the vector after subtracting the specified vector from this vector. |
| **IsParallelWith ( Vector3d** withVect **)** | Returns *True* or *False* depending on whether the vector is parallel with the specified vector. |
| **IsCodirectionalWith (Vector3d** withVect **)** | Returns *True* or *False* depending on whether the vector is pointing in the same direction as the specified vector. |
| **PerpVector ( )** | Returns a vector perpendicular to this vector. |
| **GetAngle ( Vector3d** withVect **)** | Returns the angle between the vectors (in radians). |
| **Create ( double** dX, **double** dY, **double** dZ **)** | Creates a vector with the specified coordinates. |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this vector. |
| **CrossProduct ( Vector3d** withVect **)** | Returns a vector – the result of the cross product of this vector and the specified vector. Useful when creating a CS. |
| **Multiply ( double** value **)** | Multiplies the vector with the specified value – the vector has a new length (and a new orientation if the value is negative). |
| **Normalize ( )** | Normalizes the vector (length = 1). Does not change in any way the orientation. |
| **Project ( Plane** onPlane **)** | Projects the vector on the specified plane. |
| **RotateBy ( Vector3d** aroundVect, **double** angle **)** | Rotates the vector around the specified vector with the specified angle (in radians). |
| **RotateX ( double** angle **)** | Rotates the vector around the X-axis with the specified angle value (in radians). |
| **RotateY ( double** angle **)** | Rotates the vector around the Y-axis with the specified angle value (in radians). |
| **RotateZ ( double** angle **)** | Rotates the vector around the Z-axis with the specified angle value (in radians). |
| **setFrom ( Vector3d** fromVector **)** | Creates a copy of the fromVector object into this vector. |
| **setFromPoint ( Point3d** fromPoint **)** | Similar to setFrom(), but copies in this vector’s dimensions the point coordinates. |
| **IsPerpendicularTo ( Vector3d** toVector **)** | Returns true if this vector is perpendicular to the specified vector, otherwise it returns false. |
| **IsEqualTo ( Vector3d** v**, double** dPrecision **)** | Checks with the specified precision if a specified vector is the same as the current vector. **Returns *True*** or ***False***. |
| **ProjectByDir ( Plane** onPlane, **Vector3d** v **)** | Projects the vector on the specified plane with the specified vector and returns a vector. |

### Vector3dArray

|  |  |
| --- | --- |
| **Vector3dArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | Returns the vector at the specified index in the array. |
| **Count ( )** (property – get) | Returns the number of elements in the array. |
| **Add ( Vector3d** pVal**)** | Appends a new vector in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### Line3d

|  |  |
| --- | --- |
| **Line3d – methods and properties** | |
| **Origin** (property – get) | **Returns** the origin point of this line. |
| **Direction** (property – get) | **Returns** the direction vector of this line. |
| **CreateFromVectorAndPoint ( Point3d** originPoint, **Vector3d** directionVector **)** | Creates a line with the specified origin and direction. |
| **CreateFromPoints ( Point3d** firstPoint, **Point3d** secondPoint **)** | Creates a line from the specified points. |
| **SetFrom (Line3d** targetLine**)** | Creates a copy of the targetLine object into this line. |

### LineSeg3d

|  |  |
| --- | --- |
| **LineSeg3d – methods and properties** | |
| **StartPoint** (property) | Gets or sets the segment start point. |
| **EndPoint** (property) | Gets or sets the segment end point. |
| **Length** (property – get) | Returns the segment length. |
| **Direction** (property – get) | Returns the segment direction (as *Vector3d*). |
| **CreateFromPoints ( Point3d** firstPoint, **Point3d** secondPoint **)** | Creates a line segment from the specified points. |
| **SetFrom (LineSeg3d** targetLineSeg**)** | Creates a copy of the targetLineSeg input object into this line segment. |

### LineSeg3dArray

|  |  |
| --- | --- |
| **LineSeg3dArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the segment at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( LineSeg3d** segToAdd **)** | Appends a new segment in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### Curve3d

|  |  |
| --- | --- |
| **Curve3d – methods and properties** | |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this curve. |
| **IntersectCurve ( Curve3d** withCurve **)** | **Returns** an array of points (*Point3dArray*) – the intersection points of the curve with the specified curve. |
| **GetPerpIntersections (Point3d** basePt, **Boolean** bAllowCurveExtension**)** | **Returns** an array of points – the intersection points between the curve and the perpendicular line(s) on the curve starting from the input point. The *Boolean* parameter specifies if intersection point(s) on curve’s extension must be returned. |
| **GetTanIntersections (Point3d** basePt, **Boolean** bAllowCurveExtension**)** | **Returns** an array of points – the intersection points between the curve and the tangent line(s) to the curve starting from the input point. The Boolean parameter specifies if intersection point(s) on curve’s extension must be returned. |
| **SetFrom (Curve3d** targetCurve**)** | Creates a copy of the targetCurve input object into this curve. |

### Arc3d

|  |  |
| --- | --- |
| **Arc3d – methods and properties** | |
| **Center** (property – get) | **Returns** the arc center point. |
| **Radius** (property – get) | **Returns** the arc radius. |
| **Normal** (property – get) | **Returns** the arc normal vector. |
| **Angle** (property – get) | **Returns** the arc angle (in radians). |
| **Length** (property – get) | **Returns** the arc length. |
| **StartPoint** (property – get) | **Returns** the arc start point. |
| **EndPoint** (property – get) | **Returns** the arc end point. |
| **StartVector** (property – get) | **Returns** the arc start vector. |
| **EndVector** (property – get) | **Returns** the arc end vector. |
| **ClosestPointTo (Point3d** ptIn**)** (property – get) | **Returns** the closest point on arc to the specified point **ptIn**. |
| **Type** (property – get) | **Returns** the arc type.  For information about possible values of [**eArcType**](#_Profiles_Access_data), see the appendix. |
| **AugPolyline** (property – get) | **Returns** the arc’s augmented polyline. |
| **CreateFromPoints ( Point3d** start, **Point3d** middle, **Point3d** end **)** | Creates the arc from the specified points. |
| **ChangeRadius ( double** newRadius **)** | Sets the new arc radius. |
| **IsPointOnArc (Point3d** ptIn**, double** dPrecision**)** | Checks with the specified precision if a specified point is on the arc. **Returns True** or **False**. |
| **GetParamOf (Point3d** ptIn**, double** dPrecision**)** | Calculates and **returns**, with the specified precision, the double value parameter of the arc associated to the specified point. |
| **GetPointAt (double** dParam**)** | Calculates and **returns**, the point on the arc associated to the given parameter. |
| **GetTangentVector (Point3d** ptIn**)** | **Returns** a vector tangential to the arc at the specified point. |
| **SetFrom (Arc3d** otherArc**)** | Creates a copy of the otherArc object into this arc. |

### Plane

|  |  |
| --- | --- |
| **Plane – methods and properties** | |
| **DefinitionValues ( double&** a, **double&** b, **double&** c, **double&** d **)** | Returns the definition values in the parameters. |
| **GeometricDefinition(Point3d&** ptOrig**, Vector3d&** vecX**, Vector3d&** vecY**)** (property - get) | Returns the geometric definition of the plane (origin and Ox, Oy vectors) in the parameters. |
| **Normal** (property – get) | **Returns** the plane’s normal vector. |
| **ClosestPointTo(Point3d** ptIn**)** (property – get) | **Returns** the closest point on this plane to the input point **ptIn**. |
| **DistanceTo(Point3d** ptIn**)** (property – get) | **Returns** the distance from this plane to the input point. |
| **Intersect ( Plane** withPlane **)** | **Returns** a line (Line3d) representing the intersection line between the two planes. |
| **PointOnPlane()** (property – get) | **Returns** the origin of this plane |
| **PointIsOnPlane ( Point3d** point **)** | Tests if the point is on this plane. **Returns** True or False. |
| **CreateFromPointAndNormal ( Point3d** point, **Vector3d** normal **)** | Creates the plane from a point and a vector. |
| **CreateFromPoints ( Point3d** point1, **Point3d** point2, **Point3d** point3 **)** | Creates the plane from the specified points. |
| **Translate ( double** offset **)** | Translates the plane along the normal vector with the specified value. |
| **intersectWithLine(Line3d** inLine**, Point3d&** ptIntersection**)** | Calculates the intersection point of this plane with a line. **Returns** **True** or **False** depending on whether the objects intersect. |
| **intersectWithArc(Arc3d** inArc**, long&** nNumberOfIntersectionPoints**, Point3d&** ptInt1**, Point3d&** ptInt2**)** | Calculates the number of intersection points and the intersection points themselves between this plane and an arc. **Returns** **True** or **False** depending on whether the objects intersect. |
| **intersectWithPlanes(Plane** pl1**, Plane** pl2**, Point3d&** ptIntersection) | Calculates the intersection point between this plane and **pl1** and **pl2**. **Returns** **True** or **False** depending on whether the objects intersect. |
| **LineIsOnPlane(Line3d** inLine**, double** dPrec**)** | Determines if the input line is contained on this plane taking into account the input checking precision. **Returns True** or **False**. |
| **IsParallelTo(Plane** plOther**, double** dPrec) | Determines with the specified precision if this plane is parallel to the input plane. **Returns True** or **False**. |
| **IsCoplanarWith(Plane** plOther**, double** dPrec) | Determines with the specified precision if this plane is coplanar with the input plane. **Returns True** or **False**. |
| **SetFrom (Plane** targetPlane**)** | Copies the targetPlane to the plane. |

### Matrix3d

|  |  |
| --- | --- |
| **Matrix3d – methods and properties** | |
| **Inverse** (property – get) | **Returns** the inverse matrix. |
| **Multiply ( Matrix3d** withMatrix **)** | Multiplies this matrix with the specified matrix. |
| **SetToTranslation ( Vector3d** transVector **)** | Create translation matrix |
| **SetToRotation ( Vector3d** refAxis, **double** angle **[, Point3d** refPoint **] )** | Create rotation matrix |
| **GetAt ( int** lineNumber, **int** columnNumber **)** | **Returns** the element (data type – double) in the matrix at the specified position. |
| **Invert ( )** | Invert matrix |
| **SetFrom (Matrix3d** targetMatrix**)** | Copies the targetMatrix to the matrix. |

### Matrix3dArray

|  |  |
| --- | --- |
| **Matrix3dArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the matrix at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( Matrix3d** pVal **)** | Appends a new matrix in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### AugPolyline3d

|  |  |
| --- | --- |
| **AugPolyline3d – methods and properties** | |
| **Vertices** **(**property**)** | Gets or sets the vertices of the polyline as an array of points (Point3dArray). |
| **Validity (**property**)** | Gets or sets the polyline validity.  For information about possible values of [**ePolyValidity**](#_Profiles_Access_data), see the appendix. |
| **VertexInfos (**property – get**)** | Gets the polyline’s array of vertex information. |
| **Normal (**property**)** | Gets or sets the polyline’s normal. |
| **VertexAt (long** nIndex**) (**property**)** | Gets or sets the vertex at the specified index. |
| **VertexInfoAt (long** nIndex**) (**property**)** | Gets or sets the vertex information at the specified index. |
| **ApproxVertices (**property – get**)** | Gets the approximate vertices of the polyline. |
| **Perimeter (**property – get**)** | Gets the polyline’s perimeter. |
| **ClosestPointTo(Point3d** ptIn**) (**property – get**)** | **Returns** the closest point on this polyline to the input point **ptIn**. |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this polyline. |
| **Project ( Plane** targetPlane, **Vector3d** projDir **)** | Projects the polyline on the specified plane along the specified projection direction. |
| **orthoProject ( Plane** targetPlane **)** | Orthogonal projection on the specified plane. |
| **AppendVertex ( Point3d** pointToAdd **)** | Adds the specified point to the polyline vertices. |
| **CreateFromVertices (Point3dArray** vertexArr, **VertexInfoArray** vertexInfoArr**)** | Creates a polyline described by its vertices and vertex information. After creation the **Reinitialize()** method should be called, in order to ensure a valid polyline. |
| **AppendNewVertex (Point3d** newVertex, **VertexInfo** newVertexInfo, **Boolean** bCheckValidity**)** | Appends a new vertex to the polyline. If **bCheckValidy** is set to **True**, then it also reinitializes the polyline. |
| **Expand (double** dGap**)** | Expands the polyline with the specified gap. |
| **Reinitialize ()** | Computes the polyline’s validity and normal and eliminates bad vertices (which determine collinear or parallel edges) and bad edges (zero-length lines or arcs). |
| **Reverse ()** | Reverses the order of vertices and vertex information in the polyline and reinitializes the polyline. |
| **CreatePolygon ()** | Creates and **returns** a polygon created from this polyline. |
| **DistanceTo (Point3d** ptTo**)** | **Returns**  the closest distance from the specified point to this polyline. |
| **SetNewDeviation (double** dNewDev**)** | Sets a new precision for working with the vertex information of this polyline. |
| **Clear ( )** | Removes all vertices from the polyline. |
| **SetFrom (AugPolyline3d** targetPoly**)** | Copies the targetPoly to this polyline. |
| **AppendVertices (Point3dArray** newVertices**)** | Appends an array of vertices to the polyline’s contour. |

### AugPolylinesArray

|  |  |
| --- | --- |
| **AugPolylinesArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the polyline at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( AugPolyline3d** pVal**)** | Appends a new polyline in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### AugPolygon3d

|  |  |
| --- | --- |
| **AugPolygon3d – methods and properties** | |
| **Vertices** (property) | Gets or sets the vertices of the polygon as an array of points (Point3dArray). |
| **VertexInfos (**property – get**)** | Gets the polygon’s array of vertex information. |
| **ApproxVertices (**property – get**)** | Gets the approximate vertices of the polygon. |
| **Validity (**property – get**)** | Gets the validity of the polygon.  For information about possible values of [**ePolyValidity**](#_Profiles_Access_data), see the appendix. |
| **Area (**property – get**)** | Gets the polygon’s area. |
| **SignedArea (**property – get**)** | Gets the polygon’s signed area (a vector whose direction and length represent the polygon’s normal and the polygon’s area). |
| **Perimeter (**property – get**)** | Gets the polygon’s perimeter. |
| **Normal (**property – get**)** | Getsthe polygon’s normal vector. |
| **Extents (**property – get**)** | Gets the geometrical extents of the polygon. |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this polygon. |
| **Project ( Plane** targetPlane, **Vector3d** projDir **)** | Projects the polygon on the specified plane along the given projection direction. |
| **orthoProject ( Plane** targetPlane **)** | Orthogonal projection on the specified plane. |
| **AppendVertex ( Point3d** pointToAdd **)** | Adds the specified point to the polygon vertices. |
| **Clear ( )** | Removes all vertices from the polygon. |
| **CreateFromVertices (Point3dArray** vertexArr, **VertexInfoArray** vertexInfoArr**)** | Creates a polygon described by its vertices and vertex information. |
| **Reinitialize (Boolean** bChangeVertexAllowed**)** | Computes the polygon’s validity and normal. If **bChangeVertexAllowed** is set to **True**, it also eliminates bad vertices (which determine collinear or parallel edges) and bad edges (zero-length lines or arcs). |
| **AppendNewVertex (Point3d** newVertex, **VertexInfo** newVertexInfo, **Boolean** bCheckValidity**)** | Appends a new vertex to the polygon. If **bCheckValidy** is set to **True**, then it also reinitializes the polygon. |
| **AddPolygon (AugPolygon3d** polyToAdd**)** | Adds this polygon with **polyToAdd** polygon. |
| **AddPolygons (AugPolygonsArray** polysTo | Adds this polygon with **polysToAdd** polygons. |
| **Expand (double** dGap**)** | Expands the polygon with the specified gap. |
| **IsConcave()** | Determines whether the polygon is concave. **Returns True** or **False**. |
| **SeparateByPolyline (AugPolyline3d** polyline, **double** dGap**)** | Splits the polygon into more polygons by the given polyline with the specified gap. **Returns** an array of the polygons resulting after the split. |
| **SeparateByPolylines (AugPolylinesArray** polylines, **double** dGap**)** | Splits the polygon into more polygons by the given polylines with the specified gap. **Returns** an array of the polygons resulting after the split. |
| **intersectWithOtherPolygon (AugPolygon3d** otherPoly**)** | **Returns** an array of 3D points representing the intersections between this polygon and the **otherPoly** polygon. |
| **IsPointInside (Point3d** ptToCheck, **Boolean** bCountIfOnContour, **double** dPrecision**)** | Checks if the specified point is inside this polygon with the given precision. If **bCountIfOnContour** is set to **True**, then points on the polygon’s contour are considered to be inside it. **Returns True** or **False**. |
| **IsPolygonInside (AugPolygon3d** polyToCheck, **double** dPrecision**)** | Checks if **polyToCheck** is inside this polygon, with the specified precision. **Returns True** or **False**. |
| **SetNewDeviation (double** dNewDev**)** | Sets a new precision for working with the vertex information of this polygon. |
| **SetFrom (AugPolygon3d** otherPoly**)** | Copies otherPoly to this polygon. |
| **AppendVertices (Point3dArray** newVertices**)** | Appends an array of vertices to the polygon’s contour. |
| **Subtract (AugPolygon3d** polyToSubtract, **double** dGap**)** | Subtracts the specified polygon from this polygon using the given gap. **Returns** an array containing the resulting polygons after splitting. |
| **GetGravityCenter (Double** area**)** | Returns the center of gravity point for the area. |
| **GetMomentsOfInertia ()** | Returns the moments of inertia and the center of gravity point for the polygon. |
| **Intersect (AugPolygon3d** otherPoly**)** | **Returns** a polygon representing the intersections between this polygon and the **otherPoly** polygon. |
| **Reverse ( )** | Reverses the polygon such that the normal is in the opposite direction. |

### AugPolygonsArray

|  |  |
| --- | --- |
| **AugPolygonsArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the AugPolygon3d at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( AugPolygon3d** polygonToAdd **)** | Appends a new AugPolygon3d in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### AugPolygonsArrayArray

|  |  |
| --- | --- |
| **AugPolygonsArrayArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the AugPolygon3dArray at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( AugPolygon3d** polygonToAdd **)** | Appends a new AugPolygon3dArray in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### CS3d

|  |  |
| --- | --- |
| **CS3d – methods and properties** | |
| **Origin** (property) | Gets or sets the CS origin (as a Point3d). |
| **XAxis** (property) | Gets or sets the CS X axis (as a Vector3d). |
| **YAxis** (property) | Gets or sets the CS Y axis (as a Vector3d). |
| **ZAxis** (property) | Gets or sets the CS Z axis (as a Vector3d). |
| **Matrix3d** (property) | Gets or sets the CS definition matrix (as a Matrix3d object). |
| **ThisToWCS ( )** | Returns direct transformation matrix |
| **WCSToThis ( )** | Returns reverse transformation matrix |
| **RotateCSAroundX ( double** angleVal **)** | Rotates the CS around the X-axis with the specified angle value (in radians). |
| **RotateCSAroundY ( double** angleVal **)** | Rotates the CS around the Y-axis with the specified angle value (in radians). |
| **RotateCSAroundZ ( double** angleVal **)** | Rotates the CS around the Z-axis with the specified angle value (in radians). |
| **TranslateCS ( Vector3d** translationVect **)** | Translates the CS origin with the specified vector. The vector is interpreted as being in this CS, not in WCS. For example: To translate this CS along its own X-axis the vector (1,0,0) multiplied by the desired value should be passed to this method. |
| **ChangeCS2XY ( )** | Transforms the CS: the X-axis becomes the new Y-axis, the Y-axis becomes the new Z-axis, and the Z-axis becomes the new X-axis. |
| **TransformBy ( Matrix3d** matr **)** | Applies the transformation specified by the matrix to this CS. |
| **SetToAlignCS ( CS3d** toCS **)** | **Returns** the Matrix3d that transforms objects from this coordinate system to the “toCS” coordinate system (the matrix can be used in an *object.TransformBy(matrix)* method call). |
| **SetFrom (CS3d** targetCS**)** | Copies targetCS to this CS. |

### CS3dArray

|  |  |
| --- | --- |
| **CS3dArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the coordinate system at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( CS3d** pVal **)** | Appends a new coordinate system in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### Extents

|  |  |
| --- | --- |
| **Extents – methods and properties** | |
| **MinPoint** (property – get) | **Returns** the minimum point. |
| **MaxPoint** (property – get) | **Returns** the maximum point. |
| **ExtendByPoint ( Point3d** inPoint **)** | Enlarges the extents by the given point. |
| **ExtendByPoints ( Point3dArray** inPoints **)** | Enlarges the extents by the given points. |
| **Contains ( Point3d** inPoint **)** | **Returns True** if the extents contain the given point, otherwise it returns **False**. |
| **IsInsideOrIntersecting (Extents** extOther**)** | **Returns True** if **extOther** is inside or intersecting the extents, otherwise it returns **False**. |
| **SetFrom (Extents** targetExt**)** | Copies targetExt to this Extents. |

### PlaneArray

|  |  |
| --- | --- |
| **PlaneArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the plane at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( Plane** pVal**)** | Appends a new plane in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### VertexInfo

|  |  |
| --- | --- |
| **VertexInfo – methods and properties** | |
| **Radius (**property – get**)** | **Returns** the vertex information’s radius. |
| **Center (**property – get**)** | **Returns** the vertex information’s center. |
| **Normal (**property – get**)** | **Returns** the vertex information’s normal direction vector. |
| **Type (**property – get**)** | **Returns** the vertex information’s type.  For information about possible values of [**eVertexInfoType**](#_Profiles_Access_data), see the appendix. |
| **ErrorAbsoulte (**property**)** | Gets or sets the approximation to use for working with the vertex information. |
| **TransformBy (Matrix3d** mat**)** | Transforms the vertex information by the given matrix. |
| **Project (Plane** plane**, Vector3d** direction**)** | Projects the vertex information in the input plane, using the input direction. |
| **Expand (double** dGap**)** | Expands the vertex information with the given gap. |
| **getApproxVertices (Point3d** ptStart**, Point3d** ptEnd**, long** nSamplesPerFullCircle**)** | **Returns** an array of points representing approximate vertices between the specified start and end point. The arc is sampled using the specified number of samples (per full circle). |
| **CreateFromCenterAndNormal (double** dRadius, **Point3d** ptCenter, **Vector3d** vNormal**)** | Creates the vertex information described by the specified radius, center and normal. |
| **CreateFromStartEndPointsAndNormal (Point3d** ptStart, **Point3d** ptEnd, **Vector3d** vNormal, **double** dBulge, **double** dEps**)** | Creates the vertex information described by the specified start point, end point, normal, bulge value and approximation. |
| **IsEqualTo (VertexInfo** other**)** | Determines if this vertex information and the “other” vertex information are equal. **Returns True** or **False**. |
| **SetFrom (VertexInfo** targetNfo**)** | Copies targetNfo to this vertex information. |

### VertexInfoArray

|  |  |
| --- | --- |
| **VertexInfoArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the vertex information at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( VertexInfo** pVal **)** | Appends a new VertexInformation in the array. |
| **Clear ( )** | Deletes all elements from the array. |

### 4.1.24 GeomUtils

|  |  |
| --- | --- |
| **GeonUtils – methods and properties** | |
| **CalculateBalancePoint (AstObjectsArr**pArr**)** | Calculate the center of gravity for a set of given objects. |

## Profiles access

* The following objects are used to get information from profiles. The most common type of data that you might want to read from an object (e.g. beam) can be accessed with the following object types.
* You can read, for example, the height of a beam profile (actually, the height of the profile cross-section), the cross-section width, flange thickness, etc.

### ProfType

|  |  |
| --- | --- |
| **ProfType – methods and properties** | |
| **getOuterAugPolygon ( Boolean** exactSection, **double** deviation, **Boolean** useFilletsOnConcavVertices, **Boolean** useFilletsOnConvexVertices **)** | **Returns** the profile outer contour as an AugPolygon3d object. |
| **getInnerAugPolygons** **( Boolean** exactSection, **double** deviation, **Boolean** useFilletsOnConcavVertices, **Boolean** useFilletsOnConvexVertices **)** | **Returns** the profile inner contours as an AugPolygonsArray object. |
| **createProfType ( string** className, **string** sectionName **)** | Creates a profile type with specified class type and section name.  className = value from the ***AstorProfiles*** database, ***ProfileMasterTable*** table, column TypeNameText. (class internal name)  sectionName = from the ***AstorProfiles*** database, definition table for className, value of column SectionName (section internal name) |
| **getSectionClass ( string&** className, **string&** RTLName, **string&** typeName **)** | The parameters are filled as follows:  className = value from the ***AstorProfiles*** database, ***ProfileMasterTable*** table, column SubtypeName (section shape)  RTLName = value from the ***AstorProfiles*** database, ***ProfileMasterTable*** table, column RunName (class runname)  typeName = value from the ***AstorProfiles*** database, ***ProfileMasterTable*** table, column TypeNameText. (class internal name) |
| **getRunname ( )** | **Returns** the section runname, i.e. from the definition table for profile class type, value of column StandardName (the value in the beams dialog for section – section runname). |
| **getSectionName ( )** | **Returns** from the definition table for section class, value of column SectionName (section internal name). |
| **getKey ( )** | **Returns** a string – the complete string identifying the profile name internally. The string has the following format:  class\_internal\_name#@§@#section\_internal\_name  **example:**  for the profile with class runname “US MC Channels (Imperial)” and section runname “MC6X12” you will get:  “AISC MC Channels#@§@#MC150X17.9” |
| **getGeometricalData ( eProfCommonData** data **)** | **Returns** a double data type representing the value of the parameter specified by eProfCommonData parameter.  For information about possible values of [**eProfCommonData**](#_Profiles_Access_data), see the appendix. |
| **getRefAxes ( )** | **Returns** the defined axes, as an array of integers. |
| **getRefOffset ( eProfRefAxis** refAxis **)** | **Returns** a vector.  For information about possible values of [**eProfRefAxis**](#_Profiles_Access_data), see the appendix. |
| **getVirtualPoints ( )** | **Returns** an array of points. |
| **getFittingLines ( )** | **Returns** an array of points. |
| **getFace ( Point3d&** origin, **Vector3d&** outside, **Point3d** facePoint **)** | Returns a face point and face normal vector oriented outside of the beam nearest to the specified point.  **Returns** an integer. |
| **getNextOuterFlangePoint ( Point3d&** origin, **Point3d** facePoint **)** | Returns the extents vertex nearest to the input point.  **Returns** an integer. |
| **getGapSection ( ProfGapCollection&** gaps, **Point3d** facePoint, **Point3dArray&** cont **)** | Returns an approximated section (array of points) and a ProfGapCollection object.  **Returns** an integer. |
| **getGapExactSection ( ProfGapCollection&** gaps, **Point3d** facePoint, **Point3dArray&** cont **)** | Returns the exact section (array of points) and a ProfGapCollection object.  **Returns** an integer. |
| **getGeomExtents ( )** | **Returns** the bounding box of the section, as an Extents object. |
| **getCircumference ( )** | **Returns** the profile circumference (double). |
| **getDraftLines ( )** | **Returns** an array of points. |
| **getStructuralData ( string&** profName **)** | **Returns** the profile name asa ProfSectVal object. |
| **getDSTVValues ( )** | **Returns** an IAstProfTypeDSTVValues object. |
| **getNumberOfStiffeners ( eProfStiffenerType** type **)** | **Returns** the number of possible stiffeners.  For information about possible values of [**eProfStiffenerType**](#_Profiles_Access_data), see the appendix. |
| **getStiffenerRefPoint (eProfStiffenerType** type, **int** n **)** | **Returns** the reference point for the n-th possible stiffener. |
| **getStiffenerContour ( eProfStiffenerType** type, **eProfStiffenerGen** gen, **int** n,  **Point3dArray&** retPoints **[, double** width **] )** | **Returns** a points array with the contour of the n-th possible stiffener.  For information about possible values of [**eProfStiffenerType**](#_Profiles_Access_data)and [**eProfStiffenerGen**](#_Profiles_Access_data), see the appendix. |
| **getVertexData ( eProfStiffenerType** stype, **int** n, **int** v, **eProfVertexType&** vtype, **double&** val1, **double&** val2 **)** | **Returns** the vertex feature parameter of the v-th vertex of the contour of the n-th possible stiffener.  **Returns** an integer.  For information about possible values of [**eProfVertexType**](#_Profiles_Access_data), see the appendix. |
| **getAdjacentStiffener ( int** n **)** | **Returns** the number of the adjacent stiffener for the n-th stiffener. |
| **definedAsDSTV ( )** | **Returns** an integer. |
| **getGravityUsedForWeight ( )** | **Returns** the gravity. |
| **setMirroredInX ( int** mirroredInX **)** | Sets the profile to be mirrored |
| **existsInDatabase ( )** | **Returns** an integer indicating whether the ProfType exists in the database (***AstorProfiles***). |
| **getRadius ( int** contourIndex, **double&** radius **)** | **Returns** the radius of the contour specified by the contourIndex. |
| **resetCache ( )** | Clears profile cache |
| **isUnwindable ( )** | **Returns** an integer indicating whether the profile is unwindable. |
| **getUnwind ( )** | **Returns** an integer. |
| **getDimensionPoint ( int** bExact, **Point3dArray&** dimensionPoints **)** | Returns an array with the dimension points and an integer. |
| **getSnaps ( eProfSnapType** snapType, **Point3dArray&** points **)** | **Returns** an array with the s. |
| **getMat2NCCoordSys ( )** | **Returns** a **Matrix3d** object representing the transformation matrix from the local beam’s coordinate system to the NC coordinate system. |
| **getOuterAugPolygons (Boolean** bExactSection, **Boolean** bUseFilletsOnConcavVertices, **Boolean** bUseFilletsOnConvexVertices, **double dfDeviation)** | **Returns** an **AugPolygonsArray** object representing the outer contours of a profile |
| **getCompInnerAugPolygons (Boolean** bExactSection, **Boolean** bUseFilletsOnConcavVertices, **Boolean** bUseFilletsOnConvexVertices, **double dfDeviation)** | **Returns** an **AugPolygonsArrayArray** object representing the outer contours of a compound profile. Every item in the main array contains the item inner contours of its corresponding profile in the compound profile. |
| **getKDistance ( )** | **Returns** a **double** representing the k-distance from the profile database table; for the moment this is supported only for I and U profiles; if the profile does not support k-distance or if the value is less than 0.001, it returns -1. |

### ProfSectVal

|  |  |
| --- | --- |
| **ProfSectVal – methods and properties** | |
| **A** (property) | Gets or sets the section area. |
| **Iz** (property) | Gets or sets the z moment of inertia . |
| **Iy** (property) | Gets or sets the y moment of inertia |
| **Izy** (property) | Gets or sets the centrifugal moment |
| **Sz** (property) | Gets or sets the moment. |
| **Sy** (property) | Gets or sets the data per length. |
| **U** (property) | Gets or sets the lateral surface = area per length [mm] |
| **m\_iz** (property) | Gets or sets iz |
| **m\_iy** (property) | Gets or sets iy |
| **ez** (property) | Gets or sets ez |
| **ey** (property) | Gets or sets ey |
| **G** (property) | Gets or sets the weight per length [g/mm] |

### ProfGapCollection

|  |  |
| --- | --- |
| **ProfGapCollection – methods and properties** | |
| **createGapGeneral ( double** general **)** | Creates a gap. |
| **createGapFull ( double** general, **double** face, **double** fl\_Out, **double** fl\_In, **double** web **)** | Creates a gap. |
| **createGapGeneralFace ( double** general, **double** face **)** | Creates a gap. |
| **limitTo ( double** fl\_In, **double** web **)** | **Returns** an integer. |
| **getGap ( eProfGapMode** mode **)** | **Returns** a double. |
| **getMode ( )** | **Returns** an eProfGapMode data type.  For information about possible values of [**eProfGapMode**](#_Profiles_Access_data), see the appendix. |

## Util facet

### AstModeler

This object can be obtained from a beam or a plate and represents the body of the base object.

|  |  |
| --- | --- |
| **AstModeler – methods and properties** | |
| **Volume** (property - get) | Gets the body’s volume (**double**). |
| **GeomExtens** (property - get) | Gets the body’s extents (**Extents** object). |
| **IsNull** (property - get) | Gets the null state of the body (**Boolean** values). |
| **Slice (Plane** inPlane**)** | Slices the body with the given plane. |
| **TransformBy (Matrix3d** TransformMatrix**)** | Transforms the body by the given matrix. |
| **ConvexHull ()** | Transforms the body into a convex hull. |
| **Triangulate2Dxf (string** strDxfFile**)** | Exports the body into the specified DXFfile.  Useful for debug purposes. |
| **Copy ()** | Makes a copy of the body. **Returns** an **AstModeler** object, representing the copy. |
| **Contains (Point3d** inPoint**)** | Checks whether the specified point is inside the body. **Returns** True or False. |
| **DetectOuterContoursOnPlane (Plane** inPlane**)** | Detects the outer contours of the body contained in the specified plane.  **Returns** an **AugPolygonsArray** object. |
| **DetectInnerContoursOnPlane (Plane** inPlane**)** | Detects the inner contours of the body contained in the specified plane.  **Returns** an **AugPolygonsArray** object. |
| **intersectWithLine ( Line3d** line **)** | **Returns** an array of points, representing the intersection points between the line and this object (body of a beam or a plate). |
| **intersectWithPlane ( Plane** plane **)** | **Returns** an array of line segments (**LineSeg3dArray**) representing the intersection points between the plane and this object (body of a beam or a plate). |
| **intersectWithArc ( Arc3d** arc, **double** eps **)** | **Returns** an array of points representing the intersection points between the arc and this object (body of a beam or a plate). |
| **intersectWithAxis(Point3d pt, Vector3d v, Boolean found)** | **Returns** a point representing the intersection point between the vector/point combination and this object (body of a beam or a plate). |
| **intersectWithAstModeler(AstModeler modeler)** | **Returns** true or false depending if the two **AstModelers** intersect. |

## Database access

The next object types allow you to read information from the Steel Connections databases.

### OdbcUtils

|  |  |
| --- | --- |
| **OdbcUtils – methods and properties** | |
| **GetUPSString ( long** key **)** | **Returns** a string read from the ***ErrorMessages*** table, at key (UPS / PRP / key) in the ***AstCrtlDb*** database. |
| **GetGUIString ( long** key **)** | **Returns** a string read from the ***ErrorMessages*** table, at key (GUI / PRP / key) in ***AstCrtlDb*** in the database. |
| **GetDefaultInt ( int** defaultClass, **string** defaultName **)** | **Returns** the default’s value (int) in the **AstorSettings** database. |
| **GetDefaultDouble ( int** defaultClass, **string** defaultName **)** | **Returns** the default’s value (double) in the ***AstorSettings*** database. |
| **GetDefaultString ( int** defaultClass, **string** defaultName **)** | **Returns** the default’s value (string) in the ***AstorSettings*** database. |

### OdbcTable

|  |  |
| --- | --- |
| **OdbcTable – methods and properties** | |
| **SetCurrent ( string** tableName **)** | Sets the table name in which searches will be performed. |
| **SetCurrentKey ( VARIANT** key **)** | Sets the table key. |
| **GetCurrentKey ( )** | **Returns** the table key. |
| **AddSearchCriteria ( long** columnNumber, **VARIANT** columnValue **)** | Adds a search criterion using the column number and column value.  Multiple search criteria can be defined for a table. |
| **Search ( )** | **Returns** the key of the first record found that matches all search criteria that were defined for this table.  After the search finishes, all criteria defined for the table are deleted.  Sets the current record to the one with the found key. |
| **GetAt ( long** columnNumber **)** | **Returns** the value found in the current record in the specified column. |
| **SetAt ( long** columnNumber, **VARIANT** value **)** | Set the value in the specified column for the current record. |
| **AppendNewRecord ( )** | Creates a new record and **returns** the key of the newly created record. |

## SteelAutomation

The following objects are part of the Steel Connections kernel and cannot be instantiated by any programming language through direct calls, like *new* (in C#) or *CreateInstance* (in C++).

As a consequence:

* they are either created by Steel Connections kernel for joint developers and passed in as “already created objects”.

Some samples for this are ***IRulePageArray***, ***IFiler***, ***IPropertySheetData***.

* they are created by the joint developer using calls to the IJoint interface.

For instance, if you need to create a beam or a plate you should call:

* ObjectOfTypeJoint.CreateBeam or
* ObjectOfTypeJoint.CreatePlate

The following contains all the objects available in this library.

### AstObject

|  |  |
| --- | --- |
| **AstObject – methods and properties** | |
| **Id** (property) | Gets or sets the object ID. |
| **Type** (property – get) | Gets the object class type, as an **eClassType** data type.  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |
| **RoleObject** (property) | Gets or sets the object’s role, as a Role object. |
| **Color(**property**)** | Gets or sets the object’s color.  **Note**: This only works in Advance Steel. |
| **GroupIdentifier** (property – get) | Gets the object’s group identifier. |
| **JointTransfer (property)** | Gets or sets the object’s JointTransfer. |
| **GetConnectedObjects ( )** | **Returns** an array of the connected objects of the object (as **AstObjectArr)**  All the parts in the structure, except the main part are contained in the returned array (not only the directly attached ones). |
| **IsKindOf (eClassType** type**)** | Determines if the current object represents typeof **type** class object. **Returns True** or **False**  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |

### EqualPartObject

**Note**: This only works in Advance Steel.

|  |  |  |
| --- | --- | --- |
| **EqualPartObject – methods and properties** | | |
| **DrawingNumber** (property) | Gets or sets the drawing number (as **String**) |
| **Role** (property) | Gets or sets the role name. |
| **IsMainPart** (property) | Gets or sets if the object is a main part (**True** or **False**). |
| **IsAttachedPart** (property – get) | Gets if the object is an attached part (**True** or **False**). |
| **MainPartPrefix** (property) | Gets or sets the main part prefix (**String**). |
| **SinglePartPrefix** (property) | Gets or sets the single part prefix (**String**) |
| **MainPartNumber** (property) | Gets or sets the main part number (**String**) |
| **SinglePartNumber** (property) | Gets or sets the single part number (**String**) |
| **Standalone** (property) | **True** or **False**. |
| **Material** (property) | Gets or sets the object's material (**String**) |
| **Coating** (property) | Gets or sets the object's coating (**String**) |
| **Denotation** (property) | Gets or sets the object's denotation (**String**) |
| **Assembly** (property) | Gets or sets the object's assembly (**String**) |
| **Note** (property) | Gets or sets the object's note (**String**) |
| **ItemNumber** (property) | Gets or sets the object's item number (**String**) |
| **SinglePartDetailStyle** (property) | Gets or sets the object's single part detail style (**Integer**) |
| **MainPartDetailStyle** (property) | Gets or sets the object's main part detail style (**Integer**) |
| **Name** (property) | Gets or sets the object's name (**String**) |
| **MaterialDescription** (property – get) | Gets the material description |
| **CoatingDescription** (property – get) | Gets the coating description |
| **UnwindStartFactor** (property) | Gets or sets the unwind start factor |
| **ExplicitQuantity** (property) | Gets or sets the explicit quantity |
| **SetLayer ( string** newLayer**)** | Sets the object’s layer |
| **UserAttribute (**int nId**)** | Gets the user attribute at provided index |
| **UserAttribute (**int nId, BSTR sectionName**)** | Sets the user attribute at provided index |

### AstFeatObject

|  |  |
| --- | --- |
| **AstFeatObject – methods and properties** | |
| **UseGap (**property**)** | Gets or sets the “use of additional gap” property – whether to use or not a gap when subtracting features from objects bodies. |
| **GapAddition (**property – get**)** | Gets the gap addition used for features modeling. |

### PlateFeat

|  |  |
| --- | --- |
| **PlateFeat – methods and properties** | |
| **Plate** (property – get) | Gets the owner plate of the feature. |
| **getAstModeler (eBodyContext** context**)** | **Returns** an **AstModeler** object representing the feature’s body in the given body context.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix. |

### PlateFeatVertex

|  |  |
| --- | --- |
| **PlateFeatVertex – methods and properties** | |
| **VertexIndex** (property) | Gets or sets the feature’s vertex index on the owner plate’s contour. |
| **getContextPolygon ()** | **Returns** an **AugPolygon3d** object representing the feature’s contour polygon. |

### PlateFeatEdge –

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; it is available in the Advance Steel UI.

|  |  |
| --- | --- |
| PlateFeatVertex – methods and properties | |
| **EdgeIndex** (property) | Gets or sets the feature’s edge index on the owner plate’s contour. |
| **getContextPolygon ()** | **Returns** an **AugPolygon3d** object representing the feature’s contour polygon. |

### PlateFeatContour

|  |  |
| --- | --- |
| **PlateFeatVertex – methods and properties** | |
| **Length** (property – get) | Gets the length of the feature (**double**). |
| **Offset** (property) | Gets or sets the feature’s offset (**Vector3d –** a 2D vector actually – z component always equal to 0). |
| **Width** (property – get) | Gets the feature’s width (**double**). |
| **Gap** (property) | Gets or sets the feature’s gap (**double**). |
| **Radius** (property – get) | Gets the feature’s radius (**double**). |
| **LengthIncrement** (property) | Gets or sets the feature’s length increment (**double**). |
| **RadiusIncrement** (property) | Gets or sets the feature’s radius increment (**double**). |
| **ContourType** (property) | Gets or sets the contour type (**eContourTypes**).  For information about possible values of [**eContourTypes**](#_SteelAutomation_data), see the appendix. |
| **CS** (property) | Gets or sets the feature’s coordinate system (**CS3d**). |
| **NumVertexInfo** (property – get) | Returns the number of vertices of the feature’s contour (**long**). |
| **VertexInfoAt (int** nIndex**)** (property) | Gets or sets the vertex information on the feature’s contour at the specified index (**double**). |
| **setLength (double** length, **Boolean** bIgnoreRaster**)** | Sets the length of the feature, considering the **bIgnoreRaster** setting. |
| **setWidth (double** width, **Boolean** bIgnoreRaster**)** | Sets the width of the feature, considering the **bIgnoreRaster** setting. |
| **setRadius (double** radius, **Boolean** bIgnoreRaster**)** | Sets the radius of the feature, considering the **bIgnoreRaster** setting. |
| **getContourPolygon (int** nTop**)** | Gets the feature’s contour polygon at the top or bottom of the plate. **Returns** a **Point3dArray** object. |
| **setContourPolygon (Point3dArray** contour**)** | Sets the feature’s contour. |
| **setContourVertex (int** indexGrip**, Point3d** newVertex**)** | Sets a specified vertex on the feature’s contour. |
| **transformBy (Matrix3d** matrix**)** | Transforms the feature by the given matrix. |
| **getNumberOfEcs ()** | **Returns** the number of coordinate systems of the feature. |
| **getEcs (int** nNumber**)** | **Returns** a **Matrix3d** object representing the coordinate system at the specified index. |

### Chamfer

|  |  |
| --- | --- |
| **Chamfer – methods and properties** | |
| **Length** (property – get) | Gets the chamfer’s length (**double**). |
| **Width** (property – get) | Gets the chamfer’s width (**double**). |
| **LengthIncrement** (property) | Gets or sets the chamfer’s length increment (**double**). |
| **setLength (double** length, **Boolean** bIgnoreRaster**)** | Sets the chamfer’s length, considering the **bIgnoreRaster** setting. |
| **setWidth (double** width, **Boolean** bIgnoreRaster**)** | Sets the chamfer’s width, considering the **bIgnoreRaster** setting. |
| **getBaseContourPolygon (double** zpos**)** | Gets the chamfer’s contour polygon at the specified z position. **Returns** a **Point3dArray** object. |
| **setBaseContourVertex (int** indexGrip, **Point3d** newVertex**)** | Sets the specified vertex on the chamfer’s polygon contour. |

### Fillet

|  |  |
| --- | --- |
| **Fillet – methods and properties** | |
| **Radius** (property – get) | Gets the fillet’s radius (**double**). |
| **RadiusIncrement** (property) | Gets or sets the fillet’s radius increment (**double**). |
| **FilletType** (property) | Gets or sets the fillet’s type (**eFilletTypes**).  For information about possible values of [**eFilletTypes**](#_SteelAutomation_data), see the appendix. |
| **setRadius (double** radius, **Boolean** bIgnoreRaster**)** | Sets the fillet’s radius, considering the **bIgnoreRaster** setting. |
| **getBaseContourPolygon (double** zpos**)** | Gets the chamfer’s contour polygon at the specified z position. **Returns** a **Point3dArray** object. |
| **setBaseContourVertex (int** indexGrip, **Point3d** newVertex**)** | Sets the specified vertex on the chamfer’s polygon contour. |

### PlateContourNotch

|  |  |
| --- | --- |
| **PlateContourNotch – methods and properties** | |
| **ContourType** (property) | Gets or sets the contour notch’s contour type (**eContourTypes**).  For information about possible values of [**eFilletTypes**](#_SteelAutomation_data), see the appendix. |
| **Normal** (property – get) | Gets the normal at the contour notch’s plane (**Vector3d**). |
| **ContourPolygon** (property – get) | Gets the contour notch’s contour polygon. |
| **Gap** (property) | Gets or sets the contour notch’s gap (**double**). |
| **Length** (property) | Gets or sets the contour notch’s length (**double**). |
| **Width** (property) | Gets or sets the contour notch’s width (**double**). |
| **Offset** (property) | Gets or sets the contour notch’s width (**Vector3d –** a 2D vector actually – z component always 0 equal). |
| **LengthIncrement** (property) | Gets or sets the contour notch’s length increment (**double**). |
| **Radius** (property) | Gets or sets the contour notch’s radius (**double**). |
| **RadiusIncrement** (property) | Gets or sets the contour notch’s radius increment (**double**). |
| **CS** (property) | Gets or sets the contour notch’s coordinate system (**CS3d**). |
| **NumVertexInfo** (property – get) | Gets the number of vertices on the contour notch’s contour polygon. |
| **VertexInfoAt (int** nIndex**)** (property) | Gets or sets the vertex information on the contour notch’s contour polygon at the specified index (**double**). |
| **setContourAndCS (AugPolygon3d** contour**, CS3d** cs**)** | Sets the contour and the coordinate system of the contour notch. |
| **MoveVertex (int** nIndex, **Vector3d** offset**)** | Moves the contour notch’s contour vertex at the specified index with the specified offset. |
| **MoveEdge (int** nIndex, **Vector3d** offset**)** | Moves the contour notch’s contour edge at the specified index with the specified offset. |
| **getNumberOfEcs ( )** | **Returns** the number of coordinate systems of the contour notch. |
| **getEcs (int** nNumber**)** | **Returns** a **CS3d** object representing the coordinate system at the specified index. |
| **transformBy (Matrix3d** matrix**)** | Transforms the contour notch by the specified matrix. |

### BeamNotch

|  |  |
| --- | --- |
| **BeamNotch – methods and properties** | |
| **Beam** (property – get) | Gest the owner beam of the notch (**Beam** object). |
| **RefLength** (property) | Gets or sets the length of the notch (**double**). |
| **RefDepth** (property) | Gets or sets the depth of the notch (**double**). |
| **End** (property) | Gets or sets the end of the beam where the notch lies (**eBeamEnd**).  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **Side** (property) | Gets or sets the side of the beam where the notch lies (**eBeamSide**).  For information about possible values of [**eBeamSide**](#_SteelAutomation_data), see the appendix. |
| **CS** (property) | Gets or sets the coordinate system of the notch (**CS3d**). |
| **CornerType** (property – get) | Gets the notch corner type (**eNotchCornerType**).  For information about possible values of [**eNotchCornerType**](#_SteelAutomation_data), see the appendix. |
| **CornerRadius** (property – get) | Gets the notch corner radius (**double**) |
| **LengthDefiningPlane** (property – get) | Gets the plane which defines the notch’s length. |
| **DepthDefiningPlane** (property – get) | Gets the plane which defines the notch’s depth. |
| **GetVertices ( )** | **Returns** a **Point3dArray** representing the vertices of the notch’s contour. |
| **getAstModeler (eBodyContext** context**)** | Returns an AstModeler object representing the notch’s body in the given body context.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix. |
| **getNumberOfEcs ( )** | Returns the number of coordinate systems of the notch. |
| **getEcs (int** nNumber**)** | Returns a CS3d object representing the coordinate system at the specified index. |
| **SetCorner (eNocthCornerType** type, **double** dRadius**)** | Sets the notch corner.  For information about possible values of [**eNotchCornerType**](#_SteelAutomation_data), see the appendix. |

### BeamNotchStd

|  |  |
| --- | --- |
| **BeamNotchStd – methods and properties** | |
| Same as BeamNotch |  |

### BeamNotchEx

|  |  |
| --- | --- |
| **BeamNotchEx – methods and properties** | |
| **XRotation** (property) | Gets or sets the X rotation of the notch (**eNotchXRotation**).  For information about possible values of [**eNotchXRotation**](#_SteelAutomation_data), see the appendix. |
| **XAngle** (property) | Gets or sets the notch’s X angle (**double**). |
| **ZAngle** (property) | Gets or sets the notch’s Z angle (**double**). |
| **AxisAngle** (property) | Gets or sets the notch’s axis angle (**double**). |

### BeamContourNotch

|  |  |
| --- | --- |
| **BeamContourNotch – methods and properties** | |
| **Beam** (property – get) | Gets the owner beam of the notch (**Beam** object). |
| **Gap** (property) | Gets or sets the contour notch’s gap. |
| **End** (property) | Gets the end of the beam where the contour notch lies (**eBeamEnd**).  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **GetVertices ( )** | **Returns** a **Point3dArray** representing the vertices of the notch’s contour. |
| **MoveVertex (int** nNumber**, Vector3d** offset**)** | Moves the specified vertex with the specified offset. |
| **getAstModeler (eBodyContext** context**)** | Returns an AstModeler object representing the contour notch’s body in the given body context.  For information about possible values of [eBodyContext](#_SteelAutomation_data), see the appendix. |
| **getNumberOfEcs ( )** | **Returns** the number of coordinate systems of the contour notch. |
| **getEcs (int** nNumber**)** | **Returns** a CS3d object representing the coordinate system at the specified index. |
| **TransformBy (Matrix3d** matrix**)** | Transforms the contour notch by the specified matrix. |

### BeamMultiContourNotch

|  |  |
| --- | --- |
| **BeamNotch – methods and properties** | |
| **Beam** (property – get) | Gets the owner beam of the multi-contour notch (**Beam** object). |
| **ContourType** (property) | Gets or sets the type of the multi-contour notch (**eContourTypes**).  For information about possible values of [**eContourTypes**](#_SteelAutomation_data), see the appendix. |
| **Normal** (property – get) | Gets the normal at the multi-contour notch’s plane (**Vector3d**). |
| **ContourPolygon** (property – get) | Gets the contour polygon of the multi-contour notch’s plane (**AugPolygon3d**). |
| **Gap** (property) | Gets or sets the gap of the multi-contour notch (**double**). |
| **Length** (property) | Gets or sets the length of the multi-contour notch (**double**). |
| **Width** (property) | Gets or sets the width of the multi-contour notch (**double**). |
| **Offset** (property) | Gets or sets the offset of the multi-contour notch (**Vector3d –** a 2D vector actually – z component always 0 equal). |
| **LengthIncrement** (property) | Gets or sets the length increment of the multi-contour notch(**double**). |
| **Radius** (property) | Gets or sets the radius of the multi-contour notch(**double**). |
| **RadiusIncrement** (property) | Gets or sets the radius increment of the multi-contour notch (**double**). |
| **End** (property) | Gets the end of the beam where the multi-contour notch lies (**eBeamEnd**).  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **CS** (property) | Gets or sets the coordinate system of the notch (**CS3d**). |
| **GetVertices ( )** | **Returns** a **Point3dArray** representing the vertices of the notch’s contour. |
| **NumVertexInfo** (property – get) | Gets the number of vertices on the multi-contour notch’s contour polygon. |
| **VertexInfoAt (int** nIndex**)** (property) | Gets or sets the vertex information on the multi-contour notch’s contour at the specified index (**double**). |
| **setContourAndCS (AugPolygon3d** contour**, CS3d** cs**)** | Sets the contour and the coordinate system of the multi-contour notch. |
| **MoveVertex (int** nIndex, **Vector3d** offset**)** | Moves the contour notch’s contour vertex at the specified index with the specified offset. |
| **MoveEdge (int** nIndex, **Vector3d** offset**)** | Moves the contour notch’s contour edge at the specified index with the specified offset. |
| **getAstModeler (eBodyContext** context**)** | Returns an AstModeler object representing the notch’s body in the given body context.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix. |
| **getNumberOfEcs ( )** | Returns the number of coordinate systems of the shortening. |
| **getEcs (int** nNumber**)** | Returns a CS3d object representing the coordinate system at the specified index. |
| **TransformBy (Matrix3d matrix)** | Transforms the multi-contour notch by the specified matrix. |

### AstFeaturesArray

|  |  |
| --- | --- |
| **AstFeaturesArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the AstFeatObject at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( AstFeatObject** feature **)** | Appends a new AstFeatObject in the array. |

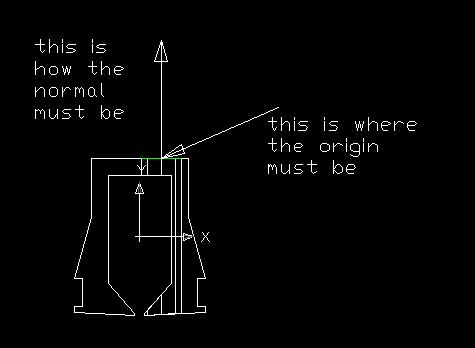
### BeamShortening

|  |  |
| --- | --- |
| **BeamShortening – methods and properties** | |
| **BeamEnd** (property) | Gets or setsthe beam end where the shortening is created (**eBeamEnd**).  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **RefLength** (property) | Same as InsLength. |
| **InsLength** (property) | Gets or sets the distance from the shortening plane to the beam end (**double**). |
| **AngleY** (property) | Gets or sets rotation angle Y (rotation angle of the cutting plane around the main axes) (**double**).  Y is the angle between the beam x-axis and the projection of the cutting plane normal vector onto the beam’s xz plane. |
| **AngleZ** (property) | Gets or sets rotation angle Z (rotation angle of the cutting plane around the main axes) (**double**).  Z is the angle between the beam x-axis and the projection of the cutting plane normal vector onto the beam’s xy plane. |
| **Beam** (property – get) | Gets the owner beam of the shortening (**Beam** object). |
| **Normal** (property – get) | Gets the normal at the shortening plane. (**Vector3d**). |
| **CS** (property – get) | Gets the coordinate system of the shortening (**CS3d**). |
| **MoveAlongSystemAxis (Vector3d** offset, **double** lengthIncrement**)** | Moves the shortening along the beam’s system line, with the specified offset and length increment. |
| **MoveAlongNormal (Vector3d** offset**, double** lengthIncrement**)** | Moves the shortening along its normal, with the specified offset and length increment. |
| **getNumberOfEcs ( )** | **Returns** the number of coordinate systems of the shortening. |
| **getEcs (int** nNumber**)** | **Returns** a CS3d object representing the coordinate system at the specified index. |

### Beam

|  |  |
| --- | --- |
| **Beam – methods and properties** | |
| **Length** (property – get) | Gets the beam length. |
| **StartPoint** (property) | Gets or sets the beam start point (**Point3d**). |
| **EndPoint** (property) | Gets or sets the beam end point (**Point3d**). |
| **StartLengthOffset** (property – get) | Gets the beam length offset from the start of the system line to the physical start of the beam. |
| **EndLengthOffset** (property – get) | Gets the beam length offset from the end of the system line to the physical end of the beam. |
| **Weight** (property – get) | Gets the weight of the beam. |
| **BalancePoint** (property – get) | Gets the balance point of the beam body (Point3d). |
| **SinglePartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a single part for numbering.  **Note**: This only makes sense in Advance Steel. |
| **MainPartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a main part for numbering.  **Note**: This only makes sense in Advance Steel. |
| **SinglePartUsedForBOM** (property) | Gets or sets whether the beam is considered as a single part for the BOM.  **Note**: This only makes sense in Advance Steel. |
| **MainPartUsedForBOM** (property) | Gets or sets whether the beam is considered as a main part for the BOM.  **Note**: This only makes sense in Advance Steel. |
| **NumberOfRods** (property - get) | Gets the number of rods of the beam |
| **SideTag ( Point3d** point**)** (property) | Gets or sets the side tag of the beam at the specified point.  For information about possible values of [**eBeamTag**](#_SteelAutomation_data), see the appendix.  **Note**: This only makes sense in Advance Steel. |
| **TaggedSide (eBeamTag** tag**)** (property - get) | Gets the end of the beam at the specified side tag  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data)and[**eBeamTag**](#_SteelAutomation_data), see the appendix.  **Note**: This only makes sense in Advance Steel. |
| **TaggedUpDown ( eBeamTag** tag**)** (property - get) | Gets the side of the beam with the given tag.  For information about possible values of [**eBeamTag**](#_SteelAutomation_data)and[**eBeamSide**](#_SteelAutomation_data), see the appendix.  **Note**: This only makes sense in Advance Steel. |
| **EndAt ( Point3d** point**)** (property - get) | Gets the end of the beam closest to the given point.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **SideAt ( Point3d** point) | Gets the side of the beam closest to the given point.  For information about possible values of [**eBeamSide**](#_SteelAutomation_data), see the appendix. |
| **RefDistFrom (eBeamSide** sideOfBeam**)** (property - get) | Gets the distance from the current reference axis to the specified side of the beam. It will return a double value.  For information about possible values of [**eBeamSide**](#_SteelAutomation_data), see the appendix. |
| **PaintArea** (property - get) | Gets the paint area of the beam’s profile. |
| **SystemlineLength** ( property - get) | Gets the length of the system line of the beam |
| **StartInnerAugmentedPolygons** (**eBodyContext** ctx, **Boolean** bUseFilletsOnConcaveVertices, **Boolean** bUseFilletsOnConvexVertices)(property - get ) | Gets the augmented inner contour polygons of the beam.  The contours lie in WCS.  The **ctx** parameter determines the body context in which the contours should be analyzed.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix.  **bUseFilletsOnConcaveVertices** and **bUseFilletsOnConvexVertices** determine whether to use fillets on vertices of the contours.  The contour is returned as an array of **AugPolygon3d** objects. |
| **StartOuterAugmentedPolygons** (**eBodyContext** ctx, **Boolean** bUseFilletsOnConcaveVertices, **Boolean** bUseFilletsOnConvexVertices)(property - get ) | Gets the augmented outer contour polygons of the beam.  The contours lie in WCS.  The **ctx** parameter determines the body context in which the contours should be analyzed.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix.  **bUseFilletsOnConcaveVertices** and **bUseFilletsOnConvexVertices** determine whether to use fillets on vertices of the contours.  The contour is returned as an array of **AugPolygon3d** objects. |
| **NumberOfInnerContours** ( property - get) | Gets the number of inner contours of the beam. |
| **InnerContourAt (int** nNumber**)** (property - get) | Gets the specified inner contour of the beam as an **AugPolygon3d** object. |
| **CrossSectionMirrored** (property) | Gets or sets the mirror state of the cross section of the beam. |
| **CS** (property - get) | Gets the beam’s coordinate system at the start of the system line as a **CS3d** object. |
| **Angle (** property**)** | Gets or sets the beam rotation around its main X-axis (in radians). |
| **PhysicalCSStart** ( property – get) | Gets the coordinate system at the physical start of the beam as a **CS3d** object. |
| **PhysicalCSEnd** ( property – get) | Gets the coordinate system at the physical end of the beam as a **CS3d** object. |
| **PhysicalCSMid** ( property – get) | Gets the coordinate system at the physical middle of the beam as a **CS3d** object. |
| **PhysicalCSAt (eBeamEnd** endOfBeam**)** (property – get) | Gets the coordinate system at the specified physical end of the beam as a **CS3d** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **PhysicalCSAtTag ( eBeamTag** tag**)** (property – get) | Gets the coordinate system at the specified physical tag of the beam as a **CS3d** object.  For information about possible values of [**eBeamTag**](#_SteelAutomation_data), see the appendix.  **Note**: This only makes sense in Advance Steel. |
| **RefAxis** ( property) | Gets or sets the reference axis of the beam.  For information about possible values of [**eProfRefAxis**](#_SteelAutomation_data), see the appendix. |
| **RefOffset (eProfRefAxis** refAxis**)** (property - get) | Gets the offset of the beam’s reference axis. The **Vector3d** object returned is a 2D vector actually, so will always have the **z** component set to **0**. |
| **BeamOffsetX** (property) | Gets or sets the beam’s X offset. |
| **BeamOffsetY** (property) | Gets or sets the beam’s Y offset. |
| **getSysCSStart ( )** | **Returns** the CS at the start of the system line of the beam.  The origin is the start point of the system line, the axes are orthogonal to the system line.  Shortenings do not affect the result. |
| **getSysCSEnd ( )** | **Returns** the CS at the end of the system line of the beam.  The origin is the end point of the system line, the axes are orthogonal to the system line.  Shortenings do not affect the result. |
| **getSysCSMid ( )** | **Returns** the CS at the middle of the system line of the beam.  The origin is the middle point of the system line, the axes are orthogonal to the system line.  Shortenings do not affect the result. |
| **getCSAtPoint ( Point3d** atPoint **)** | **Returns** the coordinate system at the given point (on the insert line). |
| **getClosestPointToSystemline ( Point3d** pickPoint **)** | **Returns** a point on the system line of the beam, closest to the pickPoint. |
| **getStartAxis ( )** | **Returns** a **LineSeg3d** object.  Gets the start point plus tangential vector directed inside the beam. |
| **getEndAxis ( )** | **Returns** a **LineSeg3d** object.  Gets the end point plus tangential vector directed inside the beam. |
| **getNumberOfEcs ( )** | **Returns** the number of coordinate systems of the beam. |
| **getEcs ( short** n**)** | **Returns** a **CS3d** object representing the specified coordinate system of the beam. |
| **numFeatures ( )** | **Returns** the total amount of features on the current beam. |
| **getFeatures ( )** | **Returns** the beam’s features collection as an **AstFeaturesArray**. |
| **ChangeReferenceSystem ( double** rotationAngle **)** | Changes the reference system of the beam. |
| **getProfType ( )** | **Returns** a **ProfType** object. |
| **getMatSysAsXY2Sys ( )** | **Returns** a **Matrix3d** object (see [**Beam Transformations**](#_Beam_Transformations)). |
| **addBeamMultiContourNotch ( Role** notchRole, **eBeamEnd** beamEnd, **AugPolygon3d** notchPoints **)** | Creates a notch at the specified beam end with the given role and polygon that define it.  **Returns and creates** a **BeamMultiContourNotch** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamMultiContourNotchClip ( Role** notchRole, **eBeamEnd** beamEnd, **AugPolygon3d** notchPoints, **Point3d** pt0, **Point3d** pt1 **)** | Creates a notch at the specified beam end with the given role and polygon that define it.  The notch has z clipping with the clipping heights defined by pt0 and pt1.  **Returns and creates** a **BeamMultiContourNotch** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamCircularContourNotch ( Role** notchRole, **eBeamEnd** beamEnd, **Vector3d** notchDirection, **Point3d** notchOrigin, **double** notchDimension **)** | Creates a circular notch at the specified beam end with the given role, direction, origin and radius.  **Returns and creates** a **BeamMultiContourNotch** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamCircularContourNotchClip ( Role** notchRole, **eBeamEnd** beamEnd, **Vector3d** notchDirection, **Point3d** notchOrigin, **double** notchDimension, **Point3d** pt0, **Point3d** pt1**)** | Creates a circular notch at the specified beam end with the given role, direction, origin and radius.  The notch has z clipping with the clipping heights defined by pt0 and pt1.  **Returns and creates** a **BeamMultiContourNotch** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamShortening ( eBeamEnd** beamEnd, **Plane** shorteningPlane **)** | Shortens the beam with the specified plane at the given beam end.  **Returns and creates** a **BeamShortening** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamContourNotch (Role** contourNotchRole, **eBeamEnd** endOfBeam, **Point3dArray** contourPts, **Vector3d** vNormal**)** | Creates a contour notch at the specified beam end with the given role. The notch is described by the contour points and the normal at the contour’s plane.  **Returns and creates** a **BeamContourNotch** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **addBeamNotchStd (Role** notchRole, **eBeamEnd** endOfBeam, **eBeamSide** sideOfBeam, **double** notchLength, **double** notchDepth**)** | Creates a standard notch at the specified beam end, on the specified beam side, with the specified role. The notch is described by length and depth.  **Returns and creates** a **BeamNotchStd** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data)and [**eBeamSide**](#_SteelAutomation_data), see the appendix. |
| **addBeamNotchEx (Role** notchRole, **eBeamEnd** endOfBeam, **eBeamSide** sideOfBeam, **double** notchLength, **double** notchDepth**)** | Creates an extended notch at the specified beam end, on the specified beam side, with the specified role. The notch is described by length and depth.  **Returns and creates** a **BeamNotchStd** object.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data)and [**eBeamSide**](#_SteelAutomation_data), see the appendix. |
| **getAstModeler ( eBodyContext** bodyContext **)** | Gets a beam body with a given body context.  **Returns and creates** an **AstModeler** object.  For information about possible values of [**eBodyContext**](#_Util_Facet_data), see the appendix. |
| **CreateHolePattern ( Role** role, **eHoleType** holeType, **ePatternArranger** arranger, **double** diameter, **eBeamEnd** beamEnd, **CS3d** holeCS **)** | Creates a hole pattern with the given attributes.  **Returns and creates** a Hole object.  For information about possible values of [**eHoleType**, **ePatternArranger** and **eBeamEnd**](#_SteelAutomation_data)see the appendix. |
| **SetUpDownTag ( Point3d** point, **eBeamTag** tag**)** | Sets a tag at the side of the beam (upper or lower side) nearest to the given point. The other side will get the opposite tag.  For information about possible values of [**eHoleType**](#_SteelAutomation_data)see the appendix. |
| **getSysCSAt (eBeamEnd** endOfBeam**)** | **Returns** the CS at the specified end of the system line of the beam. It will return a **CS3d** object.  The origin is the end point of the system line, the axes are orthogonal to the system line.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data)see the appendix. |
| **isShorteningOnBeam (Point3d** point**, Vector3d** vNormal, **double** offset**)** | Checks if a shortening is on the beam, with the given offset.  The shortening plane is described by a point and a normal.  **Returns** **True** or **False**. |
| **TransformBy (Matrix3d** matrix**)** | Transforms the beam with the given matrix. |
| **getRefAxisWCS (eProfRefAxis** refAxis**, Point3d&** ptWCS**, Vector3d&** vDir**)** | Returns a point and a vector which define a line in WCS along the required reference axis of this beam’s profile. |
| **getCutOuterCS (Boolean** bOuter**, Beam** pBeam, **CS3d&** cs1, **CS3d&** cs2, **eBeamEnd&** end1, **eBeamEnd&** end2, **Boolean** bOnMinAngleSide**)** | Gets one or two CS describing the cutting outer contour systems against the specified beam and the appropriate beam end(s) of this beam.  The origin and Z-axis of each CS define the cutting plane. The origins are ordered parametrically on the cut beam. If only one CS is found, the second is set to NULL.  If bOuter is true, the function calculates the outer CS, otherwise it calculates the upper CS.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix.  The last parameter specifies whether the CS must be on the minimum angle side or on the other side of the beam. |
| **getCutMiterCS (int** nType**, Beam** pBeam, **CS3d&** cs1, **CS3d&** cs2, **eBeamEnd&** end1, **eBeamEnd&** end2**)** | Gets one or two CS describing the miter cuts against the specified beam and the appropriate beam end(s) of this beam.  The origin and Z-axis of each CS define the cutting plane. The origins are ordered parametrically on the cut beam. If only one CS is found, the second is set to NULL.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **GetCurrentSys2Phys3dOffset (Vector3d&** offset**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **GetMatSys2Phys2d (Matrix3d&** mat**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **GetMatProf2Local (Matrix3d&** mat**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **GetMatProf2WCS (Matrix3d&** mat**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **GetMatProf2WCSEndCS (Matrix3d&** mat**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **GetMatProf2WCSMidCS (Matrix3d&** mat**)** | Please see [**Beam Transformations**](#_Beam_Transformations). |
| **ChangeProfile (sectionType, sectionName)** | Change the section of a beam. |
| **GetProfile (BSTR className, BSTR sectionName)** | Gets beam section info. |

Please see the following on how the “holeCS” for “CreateHolePattern” should be set in order to correctly drill a hole through a random profile:



The idea is that the origin of the hole should be on the face of the profile from which the hole should start, and the zAxis of the CS should point outside of the profile. In the above picture for example, if we decide to drill the hole in the other direction we could move the CS origin on the lower face which is currently affected by the hole, and we should also reverse the normal.

### StraightBeam

|  |  |
| --- | --- |
| **StraightBeam – methods and properties** | |
| **CenterPoint** (property - get) | Gets the center point of the straight beam as a **Point3d** object. |
| **GetOuterCS ( StraightBeam** toBeam, **eBeamEnd&** thisBeamEnd **)** | Gets a CS describing a cutting outer contour system against the specified beam and the appropriate beam end of this beam.  **Returns** a **CS3d** object from which the origin and Z-axis define the cutting plane.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **GetMiterCS ( StraightBeam** toBeam, **eBeamEnd&** thisBeamEnd **)** | Gets a CS describing a miter cut against the specified beam and the appropriate beam end of this beam.  **Returns** a **CS3d** object from which the origin and Z-axis define the cutting plane.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **BeamRefAxis ( eProfRefAxis** axisId, **Vector3d** retVect **)** | Gets a beam reference axis.  **Returns** a point at the start of the beam. This returned point and vector define the axis.  For information about possible values of [**eProfRefAxis**](#_Profiles_Access_data), see the appendix. |
| **SetBeamRefAxis ( eProfRefAxis** axisId **)** | Changes the system line of the beam to the specified axis.  For information about possible values of [**eProfRefAxis**](#_Profiles_Access_data), see the appendix. |
| **GetParentStraightCompound ( )** | **Returns** as a **CompoundBeamStraight** object the parent compound beam for this straight beam.  **Note**: This only makes sense in Advance Steel. |
| **GetSplitBeams ()** | **Returns** an **AstObjectsArr** array containing the pieces of a split beam as separate beams. The pieces are returned in order from start to end. |

### UnfoldedBeam

This object type cannot be created and modified directly from the Revit UI.

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; it is available in the Advance Steel UI.

|  |  |
| --- | --- |
| **UnfoldedBeam – methods and properties** | |
| **PolylineSection** (property) | Gets or sets the polyline that describes the unfolded beam section (**AugPolyline3d** object). |
| **Thickness** (property) | Gets or sets the unfolded beam thickness. |
| **Portioning** (property) | Gets or sets the unfolded beam portioning (**double** value inside the interval [ 0 , 1 ] ). |
| **NumberOfVertices** (property – get) | Gets the number of vertices of the polyline of the unfolded beam’s cross section. |
| **VertexAt (int** nIndex**)** (property) | Gets or sets the specified vertex on the polyline of the unfolded beam’s cross section. |
| **GetRadius ( int** vertexIndex **)** | **Returns** the radius at the specified vertex of the polyline that describes the unfolded beam section. |
| **SetRadius ( int** vertexIndex, **double** radius **)** | Sets the radius at the specified vertex of the polyline that describes the unfolded beam section. |
| **GetIsClosed ( )** | Gets the close/open state of the unfolded beam.  **Returns True** if the unfolded beam is closed and **False** if open. |

### BentBeam

This object type cannot be created and modified directly from the Revit UI.

**Note**: This only works in Advance Steel.

|  |  |
| --- | --- |
| **BentBeam – methods and properties** | |
| **ParamAtPoint ( Point3d** point**)** (property – get) | Gets the parameter value of the specified point. |
| **CSAtParam (double** param**)** (property – get) | Returns the coordinate system on the curved beam’s system line at the given parameter.  Returns a **CS3d** object. |
| **CSAtPoint ( Point3d** point**)** (property – get) | Returns the coordinate system on the curved beam’s system line at the given point.  Returns a **CS3d** object. |
| **ECSAtPoint (Point3d** point**)** (property – get) | Returns the coordinate system on the curved beam at the specified point.  Returns a **CS3d** object. |
| **Tangent ( Point3d** point**)** (property – get) | Gets the tangent at the given point  Returns a **Line3d** object. |
| **CurveOffset (Vector3d** offsetInCSStart**)** (property – get) | This method is obsolete. **GetCurveOffset()** should be called instead. |
| **Deviation** (property) | Gets or sets the deviation of the curved beam’s curve. |
| **OffsetCurveRadius** (property) | Gets or sets the radius of the system line with the curve’s offset |
| **ArcNormal** (property - get) | Gets the system line’s arc normal. |
| **ArcCenter** (property - get) | Gets the system line’s arc center. |
| **SystemlineArc3d (Boolean** bInWCS**)** (property - get) | Gets the system line’s arc. |
| **PhysicalArc3d (eBodyContext** ctx**)** (property - get) | Gets the system line’s physical arc in the given body context. |
| **SetSystemLine ( Point3d** startPoint**, Point3d** anyPointOnCurve**, Point3d** endPoint**)** | Sets the curved beam’s system line. The system line is described by the three input points. |
| **GetClosestPointToSystemLine ( Point3d** point**)** | Gets the closest point to the input parameter **point** on the curved beam’s system line.  **Returns** a **Point3d** object. |
| **GetPhysicallyMinMaxPoints (eBodyContext** ctx, **Point3d&** physStartPoint, **Point3d&** physEndPoint) | Gets the curved beam’s physical extents in the given body context.  For information about possible values of [**eBodyContext**](#_SteelAutomation_data), see the appendix. |
| **GetCurveOffset ()** | Gets the curve’s offset in the curved beam’s plane. |
| ChangeReferenceSystem (double angle) | Gets or sets the beam rotation around its main X-axis (angle is in degrees). |

### PolyBeam

This object type cannot be created and modified directly from the Revit UI.

**Note**: This only works in Advance Steel.

|  |  |
| --- | --- |
| **PolyBeam – methods and properties** | |
| **getPolyline (Boolean** bInWCS**)** | **Returns** the defining polyline |

### Plate

|  |  |
| --- | --- |
| **Plate – methods and properties** | |
| **Width** (property) | Gets or sets the plate’s width. |
| **Length** (property) | Gets or sets the plate’s length. |
| **Thickness** (property) | Gets or sets the plate’s thickness. |
| **Area** (property – get) | Gets the plate’s area. |
| **Weight** (property – get) | Gets the plate’s weight. |
| **Circumference** (property – get) | Gets the plate’s circumference of the base contour. |
| **BalancePoint** (property – get) | Gets the plate’s balance point of the body (Point3d object). |
| **Offset** (property) | Gets or set the plate's offset. |
| **LengthIncrement** (property) | Gets or sets the plate's length increment |
| **RadIncrement** (property) |  |
| **Portioning** (property) | Gets or sets the plate’s portioning – double value in the interval [ 0 , 1 ]. |
| **TopZDirection** (property) | This indicates the top side of the plate. It differs from plate normal only in some cases of mirrored plates. |
| **LowerZPos** (property – get) | Depends on justification and plate thickness |
| **UpperZPos** (property – get) | Depends on justification and plate thickness |
| **CenterPoint** (property – get) | Gets the plate's center point. |
| **SinglePartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a single part for numbering.  **Note**: This only makes sense in Advance Steel. |
| **MainPartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a main part for numbering.  **Note**: This only makes sense in Advance Steel. |
| **SinglePartUsedForBOM** (property) | Gets or sets whether the beam is considered as a single part for the BOM.  **Note**: This only makes sense in Advance Steel. |
| **MainPartUsedForBOM** (property) | Gets or sets whether the beam is considered as a main part for the BOM.  **Note**: This only makes sense in Advance Steel. |
| **getDefinitionPlaneCS ( )** | **Returns** a **CS3d** object representing the definition plane coordinate system. |
| **getCS ( )** | **Returns** a **CS3d** object representing the logical coordinate system; depends on the base contour. |
| **getUpperAndLowerPlane ( Plane&** upperPlane, **Plane&** lowerPlane **)** | Returns in the two parameters the upper and lower planes of the plate. |
| **moveBaseContourEdgeTo ( int** indexGrip, **Point3d** vertexNew **)** | Move plate edge |
| **setBaseContourVertex ( int** indexGrip, **Point3d** vertexNew **)** | Move plate vertex |
| **addChamfer ( double** valA, **double** valB, **int** vertexIndex **)** | **Returns and creates** a Chamfer object.  The plate corner is identified by the vertexIndex. |
| **addFillet ( double** radiusVal, **eFilletTypes** fillet\_type, **int** vertexIndex **)** | **Returns and creates** a Fillet object.  The plate corner is identified by the vertexIndex.  For information about possible values of [**eFilletTy****pes**](#_SteelAutomation_data), see the appendix. |
| **transformBy ( Matrix3d** transformationMatrix **)** | The plates position and / or orientation is modified with the specified transformation matrix (plate can be rotated, translated, etc.). |
| **getNumberOfEcs ( )** | **Returns** the number of coordinate systems of the plate. |
| **getEcs ( short** nr**)** | **Returns** a **Matrix3d** object representing the specified coordinate system of the plate. |
| **getSysCSAt ( short** vertex**)** | **Returns** a **CS3d** object representing the corresponding coordinate system on the plate vertex of the base contour. The origin of this coordinate system lies on the plate’s plane. |
| **shrinkBy ( double** gap **)** | Shrinks the plate with the specified gap. |
| **isPntInSys ( Point3d** testPoint **)** | **Returns** True if the specified point is within the plate. |
| **addPlateContourNotch ( Role** notchRole, **int** vertexIndex, **AugPolygon3d** notchPoints **)** | **Returns and creates** a PlateContourNotch object. |
| **addPlateContourNotchClip ( Role** notchRole, **int** vertexIndex, **AugPolygon3d** notchPoints, **Point3d** pt0, **Point3d** pt1 **)** | **Returns and creates** a PlateContourNotch object.  The notch has z clipping with the clipping heights defined by pt0 and pt1. |
| **addPlateCircularContourNotch ( Role** notchRole, **int** vertexIndex, **Vector3d** notchDirection, **Point3d** notchOrigin, **double** notchDimension **)** | Creates a circular contour notch at the given plate vertex with the specified role, origin, direction and radius.  **Returns and creates** a PlateContourNotch object. |
| **addPlateCircularContourNotchClip ( Role** notchRole, **int** vertexIndex, **Vector3d** notchDirection, **Point3d** notchOrigin, **double** notchDimension, **Point3d** pt0, **Point3d** pt1 **)** | Creates a circular contour notch at the given plate vertex with the specified role, origin, direction and radius.  **Returns and creates** a PlateContourNotch object.  The notch has z clipping with the clipping heights defined by pt0 and pt1. |
| **addPlateRectangleContourFeat (Role** rectContourFeatRole, **Vector3d** xVector, **Point3d** ptCenterWCS, **double** length, **double** width**)** | Creates a rectangular contour feature with the specified role. The feature is described by its x direction, center point (in WCS), length and width.  **Returns and creates** a PlateFeatContour object. |
| **addPlateCircleContourFeat (Role** circleContourFeatRole, **Vector3d** xVector, **Point3d** ptCenterWCS, **double** radius**)** | Creates a circular contour feature with the specified role. The feature is described by its x direction, center point (in WCS) and radius.  **Returns and creates** a PlateFeatContour object. |
| **addPlatePolygonContourFeat (Role** polygonContourFeatRole, **Point3dArray** ptsContourWCS**)** | Creates a polygonal contour feature with the specified role. The feature is described by its contour points (in WCS).  **Returns and creates** a PlateFeatContour object. |
| **addPlateWeldingBevel (Role** weldingBevelRole, **int** edgeIndex, **double** xDist, **double** yDist, **int** plateSide**)** | Creates a welding bevel preparation on the specified edge and side of the plate, with the specified role. The welding bevel preparation is described by its x and y lengths.  **Returns and creates** a **PlateWeldingBevel** object. |
| **addPlateWeldingFillet (Role** weldingFilletRole, **int** edgeIndex, **double** radius, **eFilletTypes** filletType, **int** plateSide**)** | Creates a welding fillet preparation on the specified edge and side of the plate, with the specified role. The welding fillet preparation is described by its radius and type.  **Returns and creates** a **PlateWeldingFillet** object. |
| **CreateHolePattern ( Role** role, **eHoleType** holeType, **ePatternArranger** arranger, **double** diameter, **CS3d** holeCS **)** | Creates a hole pattern with the given attributes.  **Returns and creates** a Hole object.  For information about possible values of [**eHoleType** and **ePatternArranger**](#_SteelAutomation_data) see the appendix. |
| **getAstModeler ( eBodyContext** bodyContext **)** | Gets a plate body with a given body context.  **Returns and creates** an AstModeler object.  For information about possible values of [**eBodyContext**](#_Util_Facet_data), see the appendix. |
| **AddBendingLine ( double** angle, **double** radius, **double** elongation, **double** plane, **int** edge **)** | Inserts a new Bending line creating an unfolded plate. |
| **CutAtCS ( eValueType** YAbsOrRelative, **double** YValue, **CS3d** csToCutAt, **Boolean** bAllowExtend **)** | **Returns** a PlateWeldingBevel object.  Cuts the plate by the given CS (actually by xOy plane of the CS) and creates a welding preparation (bevel) depending on the plane inclination relative to the plate and the YValue options.  YAbsOrRelative - use Y value as absolute or relative  YValue - Y depth the new bevel should have  bAllowExtend –extend the plate when cutting CS is outside the plate |
| **getFeatures( )** | **Returns** the plate’s features collection as an AstFeaturesArray. |

### PlateFolded

This object type cannot be created and modified directly from the Revit UI.

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; it is available in the Advance Steel UI.

|  |  |
| --- | --- |
| **PlateFolded – methods and properties** | |
| **IsUnwindable** (property – get) | Checks if this folded plate is unwindable or not. **Returns** **True** or **False**. |
| **getAstModelerUnfolded** (**eBodyContext** ctx) | Gets the unfolded body of this folded plate.  For information about possible values of [**eBodyContext**](#_Util_Facet_data), see the appendix. |
| **SetThickness** (**double** dfThickness) | Sets the folded plate’s thickness. |
| **GetMainFold** () | Gets the ID of the main fold of this folded plate (**long**). |
| **SetMainFold** (**long** nNewMainFoldId) | Sets a new main fold for the folded plate. The new main fold is specified by its ID. |
| **GetClosestFold (long** nFoldId, **Point3d** pPlateFold**)** | Get the ID of the closest folded plate’s fold to the given point (**long**) . |
| **GetFold (long** nFoldId, **PlateFold** pPlateFold**)** | Sets the ID and parent of the given **PlateFold**. The ID to be passed in is the one returned by **Joint.CreateFoldedPlate**.Without calling this the **PlateFold** object will be in an incomplete state. |
| **GetAllFolds ()** | Gets an array containing all folded plate’s folds IDs. |
| **AddFeatureTo (long** nFoldId, **AstFeatObject** pFeatToAdd**)** | Adds the given feature on the specified plate fold. |
| **GetFoldAstModeler (long** nFoldId**, eBodyContext** ctx**)** | Gets the body of the specified fold.  For information about possible values of [**eBodyContext**](#_Util_Facet_data), see the appendix. |
| **GetDriverFold (long** nFoldId**)** | Gets the ID of the specified fold’s driver fold . |
| **GetAllDrivenFolds (long** nFoldId**)** | Gets the IDs of all folds driven by the specified fold. |
| **GetNeighbourFolds (long** nFoldId**)** | Gets the IDs of the specified fold’s neighbor folds. |
| **GetDriverRelation (long** nFoldId**)** | Gets the relation which drives the specified fold. |
| **GetDrivenRelations (long** nFoldId**)** | Gets relations driven directly by the specified fold. |
| **GetRelations (long** nFoldId**)** | Gets all relations which drive/are driven directly by the specified fold. |
| **GetAllDrivenRelations (long** nFoldId**)** | Gets recursively all relations driven by the specified fold and by its descendants (driven folds). |
| **GetAllRelations ()** | Gets all folded plate’s relations. |
| **Connect (PlateFolded** otherPlate**, Point3d** ptOnThisPlate**, Point3d** ptOnOtherPlate**, long** nFoldOnThisId, **long** nFoldOnOtherId, **Role** resultingRelationRole, **PlateFoldRelation&** relation**)** | Connects this fold with the specified plate. The connection is done at the closest edge of the plates to the given points. **Returns** **True** or **False** depending if the connection was done successfully. If the connection was done successfully the **relation** parameter will contain the newly created relation. The relation will have the specified role. |
| **ExtendBy (PlateFold** pFold, **long** nMethod, **long** idFoldToConnectTo, **Point3d** ptOnFoldedPlate, **Point3d** ptOnFold, **string** resultingRelationRole, **long&** nFoldId, **PlateFoldRelation&** pRelation, | This method adds the specified fold along with all its children to the folded plate. The nMethod parameter indicates whether the position of the fold is changed in the process; if the value is 0 the selected edge on pFold (via ptOnFold) is alligned with the selected edge on the folded plate (via ptOnFoldedPlate) Otherwise, the plates are connected along their clossest edges witch are moved until they touch, if necessary. **Returns** **True** if the connection succeded or **False** otherwise. If the connection was done successfully the **relation** parameter will contain the newly created relation. The relation will have the specified role. |

### PlateFoldRelation

This object type cannot be created and modified directly from the Revit UI.

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; available in the Advance Steel UI.

|  |  |
| --- | --- |
| **PlateFoldRelation – methods and properties** | |
| **FoldedPlate** (property – get) | Gets the folded plate which owns this relation |
| **DriverFold** (property – get) | Gets the ID of this relation’s driver fold. |
| **DrivenFold** (property – get) | Gets the ID of this relation’s driven fold. |
| **Angle** (property) | Gets or sets the angle of this relation. |
| **Justification** (property) | Gets or sets the justification of this relation. |
| **Radius** (property) | Gets or sets the radius of this relation. |
| **CS** (property – get) | Gets the coordinate system of this relation. |
| **EdgeLength** (property – get) | Gets the relation’s connection length. |
| **ParentAlignMatrix** (property – get) | Returns matrix witch aligns child fold to parent fold |
| **Center** (property – get) | Gets the center point of this relation. |
| **getBendingInformation (Point3d&** ptBend**, Vector3d&** vBend**, double&** dAngle**, double&** dRadius**)** | Returns the bending information of this relation. |
| **getAstModeler (eBodyContext** ctx**)** | Gets the relation’s body.  For information about possible values of [**eBodyContext**](#_Util_Facet_data), see the appendix. |

#### **PlateFoldArray**

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; it is available in the Advance Steel UI.

|  |  |
| --- | --- |
| **PlateFoldArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the PlateFold at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( PlateFold** pVal **)** | Appends a new PlateFold in the array. |

#### **PlateFoldRelationArray**

**Note**: This is only available from the Revit Steel Connections API - not from the Revit UI; it is available in the Advance Steel UI.

|  |  |
| --- | --- |
| **PlateFoldRelationArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the PlateFoldRelation at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add (PlateFoldRelation** pVal **)** | Appends a new PlateFoldRelation in the array. |

### Bolt

|  |  |
| --- | --- |
| **Bolt – methods and properties** | |
| **ScrewLength** (property) | Gets or sets the bolt length. |
| **ScrewDiameter** (property) | Gets or sets the bolt diameter. |
| **BindingLength** (property) | Gets or sets the bolt’s binding length. |
| **BindingLengthAddition** (property) | Gets or sets the bolt binding length addition. |
| **FinishCalculationAtGap** (property) | Gets or sets the bolt’s finish calculation at gap. |
| **HoleTolerance** (property – get) | Gets the bolt hole tolerance. |
| **Norm** (property) | Gets or sets the bolt standard. |
| **Material** (property) | Gets or sets the bolt material. |
| **BoltSet** (property) | Gets or sets the bolt set. |
| **Assembly** (property – get) | Gets the bolt assembly. |
| **Annotation** (property – get) | Gets the bolt annotation. |
| **Name** (property – get) | Gets the bolt name. |
| **Coating** (property) | Gets or sets the bolt coating. |
| **MidPoints** (property - get) | Gets the middle points of a bolt pattern. |
| **CS** (property) | Gets or sets the bolt pattern’s coordinate system. |
| **Normal** (property - get) | Gets the normal of the bolt pattern. |
| **BoltNormal** (property - get) | Gets the normal of a bolt pattern’s bolts. |
| **Inverted** (property) | Gets or sets the inverted property of a bolt pattern’s bolts (bolts are inverted if the pattern and bolt normal are opposed). |
| **NumberOfScrews** (property - get) | Gets the number of bolts in a bolt pattern. |
| **SetHoleTolerance ( double** val, **Boolean** forwardToAllHoles **)** | Sets the hole tolerance. |
| **Connect ( AstObjectsArr** objectsToConnect, **eAssembleLocation** location **)** | Connects all the objects in the array with this bolt.  The holes are now created in the connected objects.  For information about possible values of [**eAssembleLocation**](#_SteelAutomation_data), see the appendix. |
| **ChangeHole ( AstObject** objectWithHole, **BoltHoleDefinition** newParametersForHole **)** | Changes the parameters of the hole created by this bolt in the specified connected element.  The parameters must be filled before calling this method in the newParametersForHole parameter.  The function must be called **ONLY AFTER THE Connect** method was called (the bolt must first create its holes, so that they can be changed). |

### Weld

|  |  |
| --- | --- |
| **Weld – methods and properties** | |
| **CenterPoint** (property – get) | **Returns** a Point3d object representing the center point. |
| **Thickness** (property) | Gets or sets the weld thickness. |
| **Connect ( AstObjectsArr** objectsToConnect, **eAssembleLocation** location **)** | Connects all the objects in the array with this weld.  For information about possible values of [**eAsse****mbleLocation**](#_SteelAutomation_data), see the appendix. |
| **ChangeSeamType ( eWeldSeam** whichSeam, **eWeldType** seamType, **eWeldSurface** surface, **eWeldSeamExecType** seamExecuting, **double** thickness, **double** distanceFromUpper **)** | Changes the seam definition for a created weld.  For information about possible values of [**eWeldSeam**](#_SteelAutomation_data), [**eWeldSurface**](#_SteelAutomation_data), [**eWeldSeamExecType**](#_SteelAutomation_data), see the appendix. |
| **ChangeWeldInfos ( double** singleSeamLength, **int** isClosed, **string** additionalInfo **)** | Changes other weld information. |
| **ChangeUpperWeldThickness ( double** thickness **)** | Changes the upper weld thickness. |
| **ChangeLowerWeldThickness ( double** thickness **)** | Changes the lower weld thickness. |
| **Pitch** (property) | Gets or sets the weld pitch. |
| **MainWeldText** (**String** MainWeldText) | Gets or sets the weld MainWeldText. |
| **DoubleWeldText** (**String** DoubleWeldText) | Gets or sets the weld DoubleWeldText. |
| **RootOpening** (property) | Gets or sets the weld RootOpening. |
| **EffectiveThroat** (property) | Gets or sets the weld Effective Throat. |
| **PreparationDepth** (property) | Gets or sets the weld **PreparationDepth.** |
| **Prefix (String prefix)** | Gets or sets the weld prefix. |

### StudShear

|  |  |
| --- | --- |
| **StudShear – methods and properties** | |
| **Standard** (property) | Gets or sets the stud standard. |
| **Material** (property) | Gets or sets the stud material. |
| **ny** (property) | Gets or sets the number of studs in the Y direction. |
| **nx** (property) | Gets or sets the number of studs in the X direction. |
| **dy** (property) | Gets or sets the distance between studs in the Y direction. |
| **dx** (property) | Gets or sets the distance between studs in the X direction. |
| **Diameter** (property) | Gets or sets the stud diameter. |
| **ConnectorLength** (property) | Gets or sets the stud length. |

### Hole

|  |  |
| --- | --- |
| **Hole – methods and properties** | |
| **GetCircularArrangerInfo ( double&** radius, **long&** numberOfHoles **)** | Returns the circular arranger radius and the number of holes of this pattern. |
| **SetCircularArrangerInfo ( double** radius, **long** numberOfHoles **)** | Sets the circular arranger radius and the number of holes of this pattern. |
| **GetRectangularArrangerInfo ( double&** dx, **double&** dy, **long&** nx, **long&** ny **)** | Returns the distances between holes of the pattern (on the x and y axes of the pattern creation CS) and the number of holes in both directions. |
| **SetRectangularArrangerInfo ( double** dx, **double** dy, **long** nx, **long** ny **)** | Sets the distances between holes of the pattern (on the x and y axes of the pattern creation CS) and the number of holes in both directions. |
| **GetBoundedArrangerInfo ( double&** length, **double&** width, **double&** dx, **double&** dy, **long&** nx, **long&** ny,  **double&** wx, **double&** wy **)** | Returns the following data:  Bounding rectangle length and width;  Distances between holes on x and y;  Number of holes on x and y;  Edge distances on x and y. |
| **SetBoundedArrangerInfo ( double** length, **double** width, **double** dx, **double** dy, **long** nx, **long** ny,  **double** wx, **double** wy **)** | Sets the following data to the hole pattern:  Bounding rectangle length and width ( the parameter values are ignored and recalculated from the other parameters if the arranger type is **kBoundedArrangerLowerLeft** );  Distances between holes on x and y ( the parameter values are ignored and recalculated from the other parameters if the arranger type is **kBoundedArrangerAll** );  Number of holes on x and y;  Edge distances on x and y. |
| **SetHoleSlotDirection ( eAxisDirection** axisDirection **)** | Sets the direction of the hole slot – on x or on y.  For information about possible values of [**eAxisDirection**](#_SteelAutomation_data), see the appendix. |
| **SetHoleSlotLength ( double** length **)** | Sets the hole’s slot length. |
| **SetExtendedParameter ( double** sunkDepth, **double** headDiameter, **double** alpha\_e, **eHoleTapping** tapping **)** | Sets the extended hole pattern information, that means the parameters for hole types.  For information about possible values of [**eHoleTapping**](#_SteelAutomation_data), see the appendix. |

### SpecialPart

This object type cannot be created and modified directly from the Revit UI.

**Note**:This only works in Advance Steel.

|  |  |
| --- | --- |
| **SpecialPart – methods and properties** | |
| **Weight** (property) | Gets or sets the special part weight. |
| **CenterPoint** (property – get) | Gets the special part’s center point |
| **CS** (property) | Gets or sets the special part’s coordinate system |
| **BlockName** (property – get) | Gets the special part’s block name. |
| **Length** (property – get) | Gets the special part’s length. |
| **Width** (property – get) | Gets the special part’s width. |
| **Depth** (property – get) | Gets the special part’s depth. |
| **Scale** (property) | Gets or sets the special part’s scale. |
| **PaintArea** (property – get) | Gets the special part’s paint area. |
| **SinglePartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a single part for numbering. |
| **MainPartUsedForNumbering** (property) | Gets or sets whether the beam is considered as a main part for numbering. |
| **SinglePartUsedForBOM** (property) | Gets or sets whether the beam is considered as a single part for the BOM. |
| **MainPartUsedForBOM** (property) | Gets or sets whether the beam is considered as a main part for the BOM. |
| **SetBlockByFile (string** filename, **double** scaleFactor**, Boolean** bOverrideIfExists**)** | Sets the special part block and scale using the input parameters **filename** (block’s filename) and **scaleFactor**. The block’s file is overridden if it already exists depending on the **bOverrideIfExists** setting. |
|  |  |

### 

### BoltHoleDefinition

|  |  |
| --- | --- |
| **BoltHoleDefinition – methods and properties** | |
| **HoleType** (property) | Gets or sets the hole type as an **eHoleType** data.  For information about possible values of [**eHoleType**](#_SteelAutomation_data), see the appendix. |
| **HoleTolerance** (property) | Gets or sets the hole tolerance. |
| **HoleDirection** (property) | Gets or sets the hole direction as an **eAxisDirection** data.  For information about possible values of [**eAxisDirection**](#_SteelAutomation_data), see the appendix. |
| **HoleSlotLength** (property) | Gets or sets the hole slot length. |
| **HoleOffset** (property) | Gets or sets the hole offset. |
| **HoleSunkDepth** (property) | Gets or sets the hole sunk depth. |
| **HoleHeadDiameter** (property) | Gets or sets the hole head diameter. |
| **HoleAlpha\_e** (property) | Gets or sets the hole alpha\_e parameter. |
| **HoleTapping** (property) | Gets or sets the hole tapping as an **eHoleTapping** data.  For information about possible values of [**eHoleTapping**](#_SteelAutomation_data), see the appendix. |

### AstUI

The following notes should be taken into account when working with **AstUserInteraction** methods:

1. Working with keywords input

The methods **AcquirePoint( ), AcquireDistance( ), AcquireDouble( ), AcquireInteger( ), AcquireUserOption( ), AcquireSingleObject( ), AcquireSingleObjectWithPoint( ), AcquireBeamWithEnd( )** allow keywords user input. In order to enable the usage of keywords the following steps must be accomplished:

* An array of accepted keywords must be constructed. The elements in the array must be of type **long** and represent **GUI-PRP** numbers.
* Before calling the **AcquireXXX()** method, the **SetKeywords()** method must be called, passing it the keywords array.
* All **AcquireXXX()** methods mentioned above have an “out” parameter **eUIErrorCodes&** outErrCode which returns an error code representing the user input. This parameter must be inspected after the call to **AcquireXXX()**. If the parameter has the “kKwd” value, it means the user has entered a valid keyword.
* To find out the keyword entered, the **GetSelectedKeyword( )** must be called. This method will return the index of the selected keyword in the keywords array. Note that after each call to an **AcquireXXX()** the keywords array is cleared.

1. Working with object filters

The methods **AcquireSingleObject( )**, **AcquireMultipleObjects( )** and **AcquireSingleObjectWithPoint( )** work with class filters for specifying the valid entities for the selection. For a correct usage of these methods, the following steps must be performed:

* A class filter must be obtained from **AstUI** by calling the **GetClassFilter** method **of AstUI** interface.
* The class filter must be empty for safe usage. Therefore, the **ClearClassFilter** method must be the very next method called after getting the class filter.
* The class filter must be filled with the desired classes which should be accepted as valid by the object selection methods. This can be done by calling the **AppendAcceptedClass** method of the **ClassFilter** interface, passing it an **eClassType** value. For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix.
* The object selection method should be called. Any object selected whose class is not appended in the class filter, will be rejected.

The following is a simple C# sample illustrating the selection of one beam:

|  |  |
| --- | --- |
| **AstUI – methods and properties** | |
| **AcquirePoint (Boolean** inWithRubberBand, **Point3d** rubberStartPoint, **Boolean** inRubberStartPtInWCS, **Boolean** inResPointInWCS, **int** nPrompt, **eUIErrorCodes&** outErrCode**)** | A point will be asked for input at the joint creation input moment. Arguments:  - **inWithRubberBand –** specifies whether a helper rubber-band should be displayed for selecting the point;  - **rubberStartPoint –** the rubber-band start point (the rubber-band will be drawn between this point and the current crosshair position);  - **inRubberStartPtInWCS –** if set to **True** then the **rubberStartPoint** is considered to be in **WCS,** otherwise it is in **UCS**;  - **inResPointInWCS –** if set to **True** then the result point will be in **WCS**, otherwise it is in **UCS**;  - **nPrompt** – string displayed by **AcquirePoint**, before it pauses;  - **outErrorCode –** error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** and **kNoResult** error codes. |
| **AcquireDistance ( int** nPrompt, **eUIErrorCodes&** outErrCode**)** | This method is deprecated.  Pauses for user input of a linear distance and **returns** the distance as a **double** value.  The **nPrompt** represents a string that **AcquireDistance** displays before it pauses.  The user can specify the distance by entering a number in the current units format. The user can set the distance also by specifying two locations on the graphics screen.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** and **kNoResult** error codes.  **Note**: This only works in Advance Steel. |
| **AcquireDouble (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | This method is deprecated.  Pauses for user input of a **double** value and **returns** the value entered.  The **nPrompt** represents a string that **AcquireDouble** displays before it pauses.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** and **kNoResult** error codes.  **Note**: This only works in Advance Steel. |
| **AcquireInteger (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | This method is deprecated.  Pauses for user input of an **integer** value and **returns** the value entered.  The **nPrompt** represents a string that **AcquireInteger** displays before it pauses.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** and **kNoResult** error codes.  **Note**: This only works in Advance Steel. |
| **AcquireUserOption (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | This method is deprecated.  Pauses for user input of a keyword and **returns** the selected keyword as a **string** value.  The **nPrompt** represents a string that **AcquireUserOption** displays before it pauses.  The maximum length of the result string is 131 characters. Any characters beyond 131 are discarded.  The user can enter the keyword from the keyboard. The list of keywords that **AcquireUserOption()** will accept is set by a prior call to **SetKeywords()**.If the user enters a string not specified in the call to **SetKeywords()**, CAD displays an error message and tries again (and redisplays prompt, if one was specified). If the user types only [Return], **AcquireUserOption()** returns an empty string.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kNoResult** error code (if the user enters a null response or arbitrary input).  The **AcquireUserOption()** function does not set **outErrorCode** to **kKwd**. The **kKwd** value is used only as a status (or error) code for other user-input (AcquireXXX()) functions.  **Note**: This only works in Advance Steel. |
| **AcquireYesOrNo (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | This method is deprecated.  Pauses for a (Yes/No) input from the user and **returns** True if the user entered Yes and False if the user entered No (particular case of **AcquireUserOption** method).  The **nPrompt** represents a string that **AcquireYesOrNo** displays before it pauses.  The outErrCode parameter will return either **kUICancel** value if the user cancels the request, or **kUINormal** value otherwise.  **Note**: This only works in Advance Steel |
| **AcquireSingleObject (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | Asks an object to be selected by the user, an entity and **returns** the selected entity as an **AstObject** object. The accepted entities types must be set using the class filtering mechanism described above prior to calling this method.  The **nPrompt** represents a string that **AcquireSingleObject** displays before it pauses.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** error code. |
| **AcquireMultipleObjects (int** nPrompt, **eUIErrorCodes&** outErrCode**)** | Asks a set of objects to be selected by the user and **returns** the selected entities as an **AstObjectArr** object. The accepted entities types must be set using the class filtering mechanism described above prior to calling this method.  The **nPrompt** represents a string that **AcquireSingleObject** displays before it pauses.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** or **kNoResult**, according to the user selection |
| **AcquireSingleObjectWithPoint (int** nPrompt, **eUIErrorCodes&** outErrCode, **Point3d& pickPoint)** | Asks an object to be selected by the user and **returns** the selected entity as an **AstObject** object. The accepted entities types must be set using the class filtering mechanism described above prior to calling this method.  The **nPrompt** represents a string that **AcquireSingleObjectWithPoint** displays before it pauses.  The **pickPoint** parameter will contain the point picked when the entity is selected.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** error code. |
| **AcquireBeamWithEnd (int** nPrompt, **eUIErrorCodes&** outErrCode, **eBeamEnd&** BeamEnd**)** | Asks a beam to be selected by the user at the desired end and **returns** the selected beam as an **AstObject** object.  The **nPrompt** represents a string that **AcquireBeamWithEnd** displays before it pauses.  The **BeamEnd** out parameter will contain the closest end of beam to the point picked when the beam was selected.  The **outErrorCode** represents an error code about the user selection. It will be set to **kNormal** if the method succeeds, **kError** if it fails or **kCancel** if the user cancels the request. A prior call to **SetKeywords** method can also enable keywords user input and the **kKwd** error code.  For information about possible values of [**eBeamEnd**](#_SteelAutomation_data), see the appendix. |
| **PrintMessage(int** nPrompt**)** | Prints a message in CAD command window. The **nPrompt** argument specifies the message to display. |
| **SetKeywords (SAFEARRAY(long)** inKeywordsArr**)** | Initializes the keywords list to be used by the next call of one of the following user input functions:  **AcquirePoint( ), AcquireDistance( ), AcquireDouble( ), AcquireInteger( ), AcquireUserOption( ), AcquireSingleObject( ), AcquireSingleObjectWithPoint( ), AcquireBeamWithEnd( )**.  The input parameters represent an array of prompts, each one of them representing one valid keyword.  For more details, please see the above comments in the **Working with keywords input** section.  **Note**: This only makes sense in Advance Steel. |
| **GetSelectedKeyword ( )** | **Returns** the index of the selected keyword in the keywords array. For more details, please see the above comments, in the **Working with keywords input** section.  **Note**: This only makes sense in Advance Steel. |
| **GetClassFilter ( );** | **Returns** a **ClassFilter** object for filtering objects. Class filters are used for object selection methods such as **AcquireSingleObject( )**, **AcquireMultipleObjects( )** and **AcquireSingleObjectWithPoint( )**. For more details, please see the above comments, in the **Working with object filters** section. |
| **ClearClassFilter ( )** | Empties the class filter, rejecting all accepted classes. |

### RulePage

|  |  |
| --- | --- |
| **RulePage – methods and properties** | |
| **hWnd** (property) | Gets or sets the page window handle as long. |
| **title** (property) | Gets or sets the page title – prompt number. |

### RulePageArray

|  |  |
| --- | --- |
| **RulePageArray – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the RulePage at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( RulePage** page **)** | Appends a new RulePage in the array. |
| **Clear ( )** | Empty the array. |

### PropertySheetData

|  |  |
| --- | --- |
| **PropertySheetData – methods and properties** | |
| **FirstPageBitmapIndex** (property) | Gets or sets the bitmap index of the bitmap to display on the “Properties” page of the joint. |
| **SheetPrompt** (property) | Gets or sets the global sheet prompt of the joint. |
| **ResizeOption** (property) | Gets or sets the page dimensions as an **eGUIDimension** data.  For information about possible values of [**eGUIDimension**](#_SteelAutomation_data), see the appendix. |

### HRLJoint

|  |  |
| --- | --- |
| **HRLJoint – methods and properties** | |
| **CreatedObjects** (property – get) | **Returns** as an **AstObjectsArr** the objects created by the HRL joint.  All these objects must be added to the created objects of the new API joint, along with the HRL joint. |
| **AddInputParamInt ( int** param **)** | Adds an **int** parameter for the HRL joint start. |
| **AddInputParamDouble ( double** param **)** | Adds a **double** parameter for the HRL joint start. |
| **AddInputParamString ( string** param **)** | Adds a **string** parameter for the HRL joint start. |
| **AddInputParamPoint ( Point3d** param **)** | Adds a **Point3d** parameter for the HRL joint start. |
| **AddInputParamVector ( Vector3d** param **)** | Adds a **Vector3d** parameter for the HRL joint start. |
| **AddInputObject ( EqualPartObject** inputObj, **int** nGroupId **)** | Adds an object parameter for the HRL joint start. |
| **RunPlayPart ( )** | Run the HRL joint. |

### Joint

|  |  |
| --- | --- |
| **Joint – methods and properties** | |
| **InputObjects** (property) | Gets or sets the joint input objects as an **AstObjectArr** object. |
| **CreatedObjects** (property) | Gets or sets the joint created objects as an **AstObjectArr** object. |
| **Updateable** (property) | Gets or sets if this joint is automatically updated. |
| **Units** (property – get) | **Returns** an **IAstUnitsManager** object. |
| **CreationStatus** (property) | Gets or sets the joint creation status. The objects created by the joint until the error occurred will be deleted and the old objects will be available. |
| **UpdateDrivenConstruction ( )** | Updates the joint. |
| **CreateSubrule ( string** subRuleName **)** | **Creates and returns** a Rule object. |
| **SaveData ( Rule** rule **)** | Saves data to joint. |
| **LoadData ( Rule** rule **)** | Loads data from joint. |
| **CreateStraightBeam ( string** className, **string** sectionName, **Role** beamRole, **Point3d** startPoint, **Point3d** endPoint, **CS3d** inputCS **)** | Creates a straight beam and **returns** a **StraightBeam** object.  The beam’s className and sectionName are the internal names for the section.  className = value from **AstorProfiles** database, ProfileMasterTable table , column TypeNameText. (class internal name)  sectionName = from the definition table for className, value of column SectionName (section internal name) |
| **CreateBentBeam ( string** className, **string** sectionName, **Role** beamRole, **Point3d** startPoint, **Point3d** endPoint, **Point3d** anyArcPoint, **double** angle **)** | Creates a bent beam and **returns** a BentBeam object.  The beam’s className and sectionName are the internal names for the section.  className = value from **AstorProfiles** database, ProfileMasterTable table , column TypeNameText. (class internal name)  sectionName = from the definition table for className, value of column SectionName (section internal name)  **Note**: This only works in Advance Steel. |
| **CreateUnfoldedBeam ( AugPolyline3d** polylineSect, **Role** beamRole, **Point3d** startPoint, **Point3d** endPoint, **CS3d** inputCS **)** | Creates an unfolded beam and **returns** an **UnfoldedBeam** object.  The beam’s cross-section is defined by the polyline, and the system line of the beam passes through the first point of the polyline.  **Note**: This only works in Advance Steel. |
| **CreateUnfoldedBeamWCS (AugPolyline3d** polyLine, **CS3d** polyLineCS, **Role** pRole, **Point3d** pt0, **Point3d** pt1, **CS3d** InputCS**)** | Creates an unfolded beam and **returns** an **UnfoldedBeam** object.  The beam’s cross-section is defined by the polyline together with its coordinate system, and the system line of the beam is defined by **pt0** and **pt1** points.  **Note**: This only works in Advance Steel. |
| **CreatePlatePoly ( Role** plateRole, **AugPolygon3d** baseContour, **double** thickness **)** | Creates a plate with the shape of the **AugPolygon3d** and **Returns** a Plate object. |
| **CreatePlateRectangular(Role** pRole**, double** dLength**, double** dWidth**, double** dThickness**, CS3d** cs**)** | Creates and **returns** a rectangular Plate object with the specified length, width and thickness in the specified coordinate system. The center of the plate will be in the origin of the given coordinate system. |
| **CreateFoldedPlate ( Role** plateRole, **PlateFold** pPlateFold, **long** nFoldId **) –** | Creates and returns a **FoldedPlate** object and its ID from the specified **PlateFold** object.  **Note**: This only works in Advance Steel. |
| **CreateWeld ( Role** weldRole**, eWeldType** weldType, **double** thickness, **Point3d** location, **CS3d** inputCS **)** | Creates and **returns** a **Weld** object. |
| **CreateBoltInfinitRectSlope ( Role** boltRole, **string** material, **string** norm, **Vector3d** slopeVect, **double** wx\_dim, **double** wy\_dim, **double** dx\_dim, **double** dy\_dim, **double** nx\_dim, **double** ny\_dim, **double** diameter, **CS3d** inputCS **)** | **Returns** an **AstObjectsArr** object.  Creates one or more bolt patterns depending on the slope vector.  The slope vector could be thought of as the axis of a beam which is to be connected by an end plate, which in turn uses bolts.  Ideally, the slope vector equals z(CS). If the slope vector will equal z(bolt), no deflection is necessary.  Typically here, there is a slope, i.e. an x or y component in terms of the bolt system.  Infinite rectangular patterns are created. |
| **CreateBoltInfinitMid ( Role** boltRole, **string** material, **string** norm, **double** dx\_dim, **double** dy\_dim, **double** nx\_dim, **double** ny\_dim, **double** diameter, **CS3d** inputCS **)** | Creates a bolt pattern of type infinite, centered.  The center of the bolt pattern is at the origin of the specified CS.  dx\_dim: distance between bolts in the X direction  dy\_dim: distance between bolts in the Y direction  nx\_dim: number of bolts in the X direction  ny\_dim: number of bolts in the Y direction  **Returns** a Bolt object. |
| **CreateGeomUtils ()** | Creates an instance of **IGeomUtils**. |
| **Save ()** | Save all joint parameters into the filer. |
| **Load** | Load joint parameters from the input filer. |
| **CreateBoltInfinitRect ( Role** boltRole, **string** material, **string** norm, **double** wx\_dim, **double** wy\_dim, **double** dx\_dim, **double** dy\_dim, **double** nx\_dim, **double** ny\_dim, **double** diameter, **CS3d** inputCS **)** | Creates a bolt pattern of type infinite, rectangular. The center of the bolt pattern is at the origin of the specified CS.  wx\_dim: edge distance in the X direction  wy\_dim: edge distance in the Y direction  dx\_dim: distance between bolts in the X direction  dy\_dim: distance between bolts in the Y direction  nx\_dim: number of bolts in the X direction  ny\_dim: number of bolts in the Y direction  **Returns** a **Bolt** object. |
| **CreateBoltFinitRect ( Role** boltRole, **string** material, **string** norm, **double** wx\_dim, **double** wy\_dim, **double** dx\_dim, **double** dy\_dim, **double** nx\_dim, **double** ny\_dim, **double** diameter, **CS3d** inputCS **)** | Creates a bolt pattern of type finite, rectangular.  The center of the bolt pattern is at the origin of the specified CS.  wx\_dim: edge distance in the X direction  wy\_dim: edge distance in the Y direction  dx\_dim: distance between bolts in the X direction  dy\_dim: distance between bolts in the Y direction  nx\_dim: number of bolts in the X direction  ny\_dim: number of bolts in the Y direction  **Returns** a **Bolt** object. |
| **CreateCircularBoltPattern(Role** boltRole, **string** material, **string** norm, **double** radius, **int** numberOfBolts, **double** diameter, **CS3d** inputCS**)** | Creates a circular bolt pattern. |
| **CreateObjectsArray ( )** | **Creates and returns** an **AstObjectsArr** object. Steel Connections objects can be appended in the array. |
| **CreateRole ( string** roleName **)** | **Creates and returns** a **Role** object with the specified name. |
| **CreateJointTransfer (string** name**)** | **Creates and returns** a **JointTransfer** object with the specified name.  Use a **JointTransfer** when you want to be able to edit the properties of an object created by the joint. |
| **CreateBlock ( AstObjectsArr** fromObjects, **string** path, **string** blockName, **Boolean** deleteSourceObjects, **Point3d** blockOrigin **)** | Creates a block from the elements in the array. These objects can be deleted after the block is created, depending on the value of deleteSourceObjects. You must NOT add these elements to the array of created objects of the joint if they will be deleted when creating the block.  The created block can now be inserted as a special part using the **CreateSpecialPart** method.  **Returns** the real name (as a string) with what the block was created in the model. This name must be passed as a parameter to the **CreateSpecialPart** method.  The origin of the block will be set in the blockOrigin point. |
| **CreateSpecialPart ( Role** spRole, **double** scale, **string** realBlockName, **CS3d** inputCS **)** | **Creates and returns** a **SpecialPart** object.  For the insertion point, the inputCS origin must be correctly specified. |
| **CreateStudShearConnector ( Role** role, **CS3d** inputCS, **ePatternArranger** arrangerType, **AstObject** connectedObj **)** | **Creates and returns** a **StudShear** object.  The center of the stud shear pattern is at the origin of the specified CS. |
| **CreateBoltHoleParameter ( )** | **Creates and returns** a **BoltHoleDefinition** object. This object can be used to change the holes created by a bolt. |
| **CreateHRLSubJoint ( string** HRLsubruleName **)** | Creates and returns a **HRLJoint** object. You can now fill the info needed in this object and call its **RunPlayPart** method. |
| **GetUCS ( )** | **Returns** the current UCS. |
| **CreateStructuralBox ( CS3d** inputCS, **double** boxDepth, **double** boxWidth, **double** boxHeight **)** | **Creates and returns** a StructuralBox object.  The parameters for creation must be:  boxDepth – dimension on inputCS Z axis  boxWidth – dimension on inputCS X axis  boxHeight – dimension on inputCS Y axis  **Note**: This only works in Advance Steel. |
| **CreateFiler ( )** | **Creates and returns** a Filer object. You must use this object if you want to call as a subjoint another joint written with the new API.  In order to call a new API joint as a subjoint, check the examples in the appendix. |
| **CreateAstColor (eAstColorMethod** colorMethod**)** | **Creates and returns** an **AstColor** object. The given parameter for the creation must be the color method for the new color.  For information about possible values of [**eAstColorMethod**](#_SteelAutomation_data), see the appendix.  **Note**: This only makes sense in Advance Steel. |
| **CreatePolyBeam ( string** sectClass**, string** Name**, Role** pRole**, AugPolyline3d** line**, Vector3d** vecRefOrientation**, Vector3d** zVec**)** | **Creates and returns** a **PolyBeam** object.  The beam’s className and sectionName are the internal names for the section.  className = value from **AstorProfiles** database, **ProfileMasterTable** table, column TypeNameText. (class internal name)  sectionName = from the definition table for className, value of column SectionName (section internal name)  line – defines the new polybeam’s polyline  zVec – defines the Z direction of the polyline. |
| **CreatePolyPlateFolded (Role** pRole**, AugPolygon3d** poly**, double** thickness**)** | **Created and returns** a polygonal Folded Plate object. |
| **CreateRectangularPlateFolded (Role** pRole**, double** dLength**, double** dWidth**, double** dThickness**, CS3d** cs**)** | **Creates and returns** a rectangular Folded Plate object with the specified length, width and thickness and in the specified coordinate system. The center of the folded plate will be in the origin of the coordinate system. |
| **DisplayMessage ( long** GUIPromptNumber**, bool** DisplayBox **) - does not make sense in revit;(AndreeaM** | Display a message in a message box or in the CAD command window. |
| **CreateStructuralSingle3GFrameSymm (CS3d** pCS**, double** dHeight**, double** dWidth**, double** dRHeight**)** | **Creates and returns** a structural single 3G symmetrical frame with the specified height, width and roof height, in the specified coordinate system. |
| **CreateStructuralSingleSlopeFrame (CS3d** pCS**, double** dHeight**, double** dWidth**)** | **Creates and returns** a structural single slope frame with the specified height and width, in the specified coordinate system.  **Note**: This only makes sense in Advance Steel. |
| **CreateStructuralMultipleSimpleFrame (CS3d** pCS**, double** dHeight**, double** dWidth**, int** nFields**)** | **Creates and returns** a structural multiple simple frames with the specified number of fields, height and width, in the specified coordinate system.  **Note**: This only makes sense in Advance Steel. |
| **CreateStructuralMultipleTowerFrame (CS3d** pCS**, double** dBottomHeight**, double** dTopHeight**, double** dWidth**, int** nFields**)** | **Creates and returns** a structural multiple tower frames with the specified number of fields, top height, bottom height and width, in the specified coordinate system.  **Note**: This only makes sense in Advance Steel. |

### Rule

The following methods and properties must be implemented in a new API joint. See the [Architecture](#_Architecture) chapter.

|  |  |
| --- | --- |
| **Rule – methods and properties** | |
| **Joint** (property) | Gets or sets the joint object of this rule. |
| **Query ( AstUI** ui **)** | Asks the user for input with the help of **AstUI** and adds the necessary entities to **InputObjects** of the Joint. |
| **InField ( Filer** filer **)** | Reads data from the Joint through the Filer. |
| **OutField ( Filer** filer **)** | Saves data to the Joint through the Filer. |
| **GetTableName ( )** | Returns the name of the tabular parameters table from the database. |
| **CreateObjects ( )** | This is where joints perform objects creation steps. |
| **GetUserPages ( RulePageArray** pagesRet, **PropertySheetData** pPropSheetData **)** | Gets user pages of rule. |
| **FreeUserPages ( )** | Releases user pages. |
| **GetExportData ( RuleExportFiler** pExportFiler **)** | Data to be exported (obsolete). |
| **ConvertFromHRL ( ConvertFiler** filer**, String** OldHRLRuleName **)** | This method must be written if the new joint was designed to replace and old HRL joint and the developer intends to convert the old joint to the new one.  The received filer is an incremental filer. Using this filer, all parameters of the old rule are accessible. Also, the conversion filer provides the rule version.  The method must **return** **true** if the conversion was successful.  **Note**: This only makes sense in Advance Steel. |
| **GetFeatureName** **( string&** FeatureName **)** | **Returns** **True** or **False** depending on license feature usage for this rule.  The *FeatureName* parameter must be filled with the license feature name, identical to the feature name in the license file (if a feature is used for the rule).  **Note**: This is currently unused - obsolete. |
| **InvalidFeature ( int** reserved **)** | Automatically called in the case of invalid feature for the rule (if a feature is used).  Write your own code for such an event or simply let this function do nothing.  **Note**: This is currently unused – obsolete. |

### Filer

|  |  |
| --- | --- |
| **Filer – methods and properties** | |
| **readVersion ( )** | **Returns** the current rule version. |
| **writeVersion ( long** version **)** | Sets the current rule version. |
| **readItem ( string** name **)** | **Returns** the data from the Filer with the specified name. |
| **writeItem ( VARIANT** value, **string** name **)** | Writes the data to the Filer with the specified name |
| **GetItems ()** | Returns all joint parameters map. |

### FilerItem

|  |  |
| --- | --- |
| **Filer – methods and properties** | |
| **ItemKey ( )** | Returns the joint parameter key. |
| **ItemValue ( )** | Returns the joint parameter value. |

### DbCursor

|  |  |
| --- | --- |
| **DbCursor – methods and properties** | |
| **Key** (property) | Gets or sets the external database key. |
| **AppendDataToTable ( )** | Appends data into a new record. |
| **IsSavingToTable ( Rule** rule, **bool&** bWillSaveToTable **)** | Sets bWillSaveToTable parameter to **true** if the filer will be used to store a record in the tabular parameters table after the **OutField** method will be finished.  **Returns** the value of the key at which the import will be made (only values of type LONG will be returned). |

### Role

This object will be passed after it is created to almost all of the methods that create Steel Connection objects. The role has mainly two “visible” applications: to identify objects with identical geometry and properties **and to** set which of the properties created objects can be set by the user outside of the joint.

To define a complete role means to assign a name to the role , the class type of the objects it will be applied to, and to define which of the properties of the object will be editable outside the joint.

After setting the object class type, you have to specify which will be the editable attributes. In C++ syntax, this is what you have to do:



In C# the syntax is:

If a role attribute should **not** be modified outside the joint (this is the default state of attributes – read-only), you could use:

* The line below means the beam material **cannot** be set outside the joint(C++)

- The line below means the beam material **cannot** be set outside the joint(C#)



|  |  |
| --- | --- |
| **Role – methods and properties** | |
| **Name** (property) | Gets or sets the Role name. |
| **ClassType** (property) | Gets or sets the Role class type as an eClassType.  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |
| **Attribute ( eAttributeCodes** attr **)** (property) | Gets or sets the status of the Role’s specified attribute.  For information about possible values of [**eAttributeCodes**](#_SteelAutomation_data), see the appendix. |

### AstObjectsArr

|  |  |
| --- | --- |
| **AstObjectsArr – methods and properties** | |
| **Item ( VARIANT** index **)** (property – get) | **Returns** the AstObject at the specified index in the array. |
| **Count ( )** (property – get) | **Returns** the number of elements in the array. |
| **Add ( AstObject** object **)** | Appends a new AstObject in the array. |

### ClassFilter

|  |  |
| --- | --- |
| **ClassFilter – methods and properties** | |
| **Count ( )** (property – get) | **Returns** the number of accepted classes in the class filter. |
| **ElementAt (int** Index**)** (property – get) | **Returns** the class type at the specified index in the class filter as an **eClassType** value.  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |
| **AppendAcceptedClass ( eClassType** Val**)** | Appends the specified class type to the class filter, as an accepted class.  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |
| **Clear ( )** | Empties the class filter, rejecting all classes in it. |

### AstColor

**Note**: This only makes sense in Advance Steel.

|  |  |
| --- | --- |
| **AstColor – methods and properties** | |
| **ColorMethod ( )** (property) | Gets or sets the color method for the color as an **eAstColorMethod** value.  For information about possible values of [**eAstColorMethod**](#_SteelAutomation_data), see the appendix. |
| **ColorIndex ( )** (property) | Gets or sets the color index of an ACI color.  0 and 256 values have special meaning: 0 specifies **ByBlock** color and 256 specifies **ByLayer** color. |
| **Red ( )** (property) | Gets or sets the Red value of a RGB color. |
| **Green ( )** (property) | Gets or sets the Green value of a RGB color. |
| **Blue ( )** (property) | Gets or sets the Blue value of a RGB color. |
| **getRGBColor (int&** red, **int&** green, **int&** blue**)** | Gets the values of red, green and blue of a RGB color. |
| **setRGBColor (int** red, **int** green, **int** blue**)** | Sets the values of red, green and blue for a RGB color. |
| **isByACI ( )** | **Returns True** for an ACI color and **False** otherwise. |
| **isByRGB ( )** | **Returns True** for a RGB color and **False** otherwise. |

### JointTransfer

|  |  |
| --- | --- |
| **JointTransfer – methods and properties** | |
| **Id** (property) | Gets or sets the name (ID). |
| **ClassType** (property) | Gets or sets the Role class type as an eClassType.  For information about possible values of [**eClassType**](#_SteelAutomation_data), see the appendix. |
| **Attribute ( eAttributeCodes** attr **)** (property) | Gets or sets the status of the Role’s specified attribute.  For information about possible values of [**eAttributeCodes**](#_SteelAutomation_data), see the appendix. |

### AstCreator

Very similar with the Joint interface in regards to its purpose.

All object creation methods are available, and are similar with the [Joint](#_Joint) methods.

**Note**: This only makes sense in Advance Steel.

|  |  |
| --- | --- |
| **AstCreator – methods and properties** | |
| **CreateGeomUtils ()** | Creates an instance of IGeomUtils. |

# Chapter 6 Appendix – Common data and types

## Profiles Access data

|  |  |  |
| --- | --- | --- |
| **eProfCommonData** | | |
| kProfHeight | 1 | Section height |
| kWidth | 2 | Section width |
| kWeb | 3 | Web thickness |
| kFlange | 4 | Flange thickness |
| kRadiusWeb | 5 | Inner radius web/flange |
| kRadiusFlange | 6 | Outer radius flange |
| kFlangeWx1 | 7 | Marker line along the section X-axis, distance from the physical center to the right |
| kFlangeWx2 | 8 | Second marker line along the section X-axis, distance from the physical center to the right |
| kFlangeWy1 | 9 | Marker line along the section Y-axis, distance from the physical center to the upper side |
| kFlangeWy2 | 10 | Second marker line along the section Y-axis, distance from the physical center to the upper side |
| kFlangeWy3 | 11 | Third marker line along the section Y-axis, distance from the physical center to the upper side |
| kEx | 12 | Eccentricity against the section X-axis, distance from the physical center to the left |
| kEy | 13 | Eccentricity against the section Y-axis, distance from the physical center to the lower side |
| kExWeb | 14 |  |

|  |  |  |
| --- | --- | --- |
| **eProfRefAxis** | | |
| kUpperLeft | 0 |  |
| kUpperSys | 1 |  |
| kUpperRight | 2 |  |
| kMidLeft | 3 |  |
| kSysSys | 4 |  |
| kMidRight | 5 |  |
| kLowerLeft | 6 |  |
| kLowerSys | 7 |  |
| kLowerRight | 8 |  |
| kUpperWeb | 101 |  |
| kSysWeb | 104 |  |
| kLowerWeb | 107 |  |

|  |  |  |
| --- | --- | --- |
| **eProfStiffenerType** | | |
| kFull | 1 | Full stiffener |
| kHalf | 2 | Half stiffener |

|  |  |  |
| --- | --- | --- |
| **eProfStiffenerGen** | | |
| kGeneric | 0 |  |
| kGenDefault | 1 |  |
| kExplicit | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eProfVertexType** | | |
| kPVTNone | 0 |  |
| kRadius | 1 | concave |
| kFillet | 2 | convex |
| kChamfer | 3 |  |

|  |  |  |
| --- | --- | --- |
| **eProfSnapType** | | |
| kPSTFlangeMiddle | 0 |  |

|  |  |  |
| --- | --- | --- |
| **eProfGapMode** | | |
| kGapNone | 0 |  |
| kGapGeneral | 1 |  |
| kGapFlangeOut | 2 |  |
| kGapWeb | 3 |  |
| kGapFlangeIn | 4 |  |
| kGapFace | 10 |  |
| kGapEqual | 11 |  |
| kGapEqualButFace | 12 |  |
| kGapEqualButFaceAside | 13 |  |
| kGapUnequal | 21 |  |
| kGapUnequalAndFace | 22 |  |

## SteelAutomation data

|  |  |  |
| --- | --- | --- |
| **eWeldType** | | |
| kFlangeButtWelding | 1 | Flange butt-welding joint |
| kIWeld | 2 | I weld |
| kVWeld | 3 | V weld |
| kHVWeld | 4 | HV weld |
| kYWeld | 5 | Y weld |
| kHYWeld | 6 | HY weld |
| kUWeld | 7 | U weld |
| kHUWeld | 8 | HU weld |
| kTWeld | 9 | Fillet weld |
| kPerforationWeld | 10 | Perforation weld |
| kSpotWeld | 11 | Spot weld |
| kLineWeld | 12 | Line weld |
| kSquareEdgeWeld | 13 | Square edge weld |
| kHalfSquareEdgeWeld | 14 | Half-square edge weld |
| kFrontFlushWeld | 15 | Front flush weld |
| kProfileWeld | 16 | Profile weld |
| kSlantWeld | 17 | Slant weld |
| kSeamWeld | 18 | Seam weld |
| kResufaceWeld | 19 | Resurface weld |
| kCounterWeld | 20 | Counter weld |
| kFlangeButtWeldingJointWithCounterWeld | 1001 | Flange butt-welding joint with counter weld |
| kDIWeld | 1002 | Combi welding D-I |
| kVWeldWithCounterWeld | 1003 | Combi welding V-Weld with counter weld |
| kXWeld | 1004 | Combi welding X |
| kKWeld | 1005 | Combi welding K |
| kDYWeld | 1006 | Combi welding D-Y |
| kKWebWeld | 1007 | Combi welding K-web weld |
| kDUWeld | 1008 | Combi welding D-U |
| kDHUWeld | 1009 | Combi welding D-HU |
| kVUWeld | 1010 | Combi welding V-U |
| kDoubleFilletWeld | 1011 | Combi welding double fillet weld |

|  |  |  |
| --- | --- | --- |
| **eWeldSeam** | | |
| kUpperSeam | 0 | Upper weld seam |
| kLowerSeam | 1 | Lower weld seam |

|  |  |  |
| --- | --- | --- |
| **eWeldSurface** | | |
| kUndefined | 0 |  |
| kFlat | 1 |  |
| kConcav | 2 |  |
| kConvex | 3 |  |

|  |  |  |
| --- | --- | --- |
| **eWeldSeamExecType** | | |
| kUndefinedType | 0 |  |
| kRootElaborated | 1 |  |
| kSeamSmoothed | 2 |  |
| kSeamTransitionsWrought | 3 |  |
| kAddendumUsed | 4 |  |
| kPadUsed | 5 |  |

|  |  |  |
| --- | --- | --- |
| **ePatternArranger** | | |
| kCircularArranger | 1 | Circular, the CS origin is the midpoint of the arranger. |
| kRectangularArranger | 2 | Rectangular, the CS origin is the midpoint for the arranger |
| kBoundedArrangerAll | 3 | Bounded, the CS origin is the midpoint of the arranger |
| kBoundedArrangerLowerLeft | 4 | Bounded, the CS origin is the lower left point of the arranger |

|  |  |  |
| --- | --- | --- |
| **eGUIDimension** | | |
| kStandard | 0 | Standard dialog pages dimension |
| kBig | 1 | 25% increased size |

|  |  |  |
| --- | --- | --- |
| **eAssembleLocation** | | |
| kOnSite | 0 |  |
| kInShop | 1 |  |

|  |  |  |
| --- | --- | --- |
| **eBeamEnd** | | |
| kBeamStart | 1 |  |
| kBeamEnd | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eBeamSide** | | |
| kUpperSide | 1 |  |
| kLowerSide | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eHoleTapping** | | |
| kHoleTappingLeft | 0 |  |
| kHoleTappingRight | 1 |  |

|  |  |  |
| --- | --- | --- |
| **ePosMarkType** | | |
| kPosMarkUndefined | 0 |  |
| kSinglePart | 1 |  |
| kMainPart | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eClassType** | | |
| kUnknownClass | 0 | Generic |
| kBeamClass | 200 | Beam |
| kBeamStraightClass | 203 | Straight Beam |
| kBeamShorteningClass | 206 | Beam Shortening |
| kBeamNotchStdClass | 210 | Standard Beam Notch |
| kBeamContourNotchClass | 212 | Beam Contour Notch |
| kBeamMultiContourNotch | 213 | Beam Multi-Contour Notch |
| kBeamBentClass | 214 | Curved Beam |
| kBeamUnfoldedClass | 215 | Unfolded Beam |
| kBeamCompoundClass | 216 | Compound Beam |
| kBeamStraightCompoundClass | 217 | Compound Straight Beam |
| kBeamPolyClass | 218 | Poly Beam |
| kBeamNotchExClass | 219 | Extended Beam Notch |
| kPlateClass | 300 | Plate |
| kPlateFeatVertChamfer | 303 | Plate Chamfer |
| kPlateFeatVertFillet | 304 | Plate Fillet |
| kPlateFeatContour | 305 | Plate Contour Feature |
| kPlateContourNotch | 306 | Plate Contour Notch |
| kPlateWeldingPreparation | 307 | Plate Welding Preparation |
| kPlateWeldingBevel | 308 | Plate Welding Bevel |
| kPlateWeldingFillet | 309 | Plate Welding Fillet |
| kPlateFoldRelationClass | 313 | Plate Fold Relation |
| kPlateFoldedClass | 314 | Folded Plate |
| kBoltPattern | 400 | Bolt Pattern |
| kScrewBoltPattern | 401 | Screw Bolt Pattern |
| kCircleScrewBoltPattern | 402 | Circle Screw Bolt Pattern |
| kCountableScrewBoltPattern | 403 | Countable Screw Bolt Pattern |
| kFinitrectScrewBoltPattern | 404 | Finite Rect Screw Bolt Pattern |
| kInfinitrectScrewBoltPattern | 405 | Infinite Rect Screw Bolt Pattern |
| kInfinitmidScrewBoltPattern | 406 | Infinite Mid Screw Bolt Pattern |
| kWeldPattern | 800 | Weld Pattern |
| kAstWeldLevel1 | 801 | Use this class for weld’s model roles. |
| kHoleFeat | 1000 | Hole |
| kHoleBeam | 1001 | Beam hole |
| kHolePlate | 1002 | Plate hole |
| kSpecialPartWithBlock | 1100 | Special Part |
| kUserDefinedCS | 1204 | User Defined CS |
| kUserDefinedPoint | 1205 | User Defined Point |

|  |  |  |
| --- | --- | --- |
| **eFilletTypes** | | |
| kFillet\_Convex | 0 |  |
| kFillet\_Concav | 1 |  |
| kFillet\_Undefined | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eBeamTag** | | |
| kThisSide | 0 |  |
| kOtherSide | 1 |  |

|  |  |  |
| --- | --- | --- |
| **eBeamReprType** | | |
| kOffMode | 0 |  |
| kBeamMode | 1 |  |
| kFeatMode | 2 |  |
| kRodMode | 3 |  |
| kSymbolMode | 4 |  |

|  |  |  |
| --- | --- | --- |
| **eRuleExportDataType** | | |
| kGUIPrompt | 0 | GUI/PRP message |
| kUPSPrompt | 1 | UPS/PRP message |
| kBitmap | 2 | bitmaps |
| kTable | 3 | Tables (excluding tabular parameters tables) |
| kSubRule | 4 | Subrules |

|  |  |  |
| --- | --- | --- |
| **eHoleType** | | **Possible parameter** |
| kHole | 1 | Diameter |
| kSlottedHole | 2 | Diameter, slot length, slot direction |
| kCounterSunk | 3 | Diameter , sunk depth, alpha\_e |
| kBlindHole | 4 | Diameter, head diameter, depth |
| kThreadHole | 5 | Diameter, sunk depth, alpha\_e, tapping |
| kSunkBolt | 6 | Diameter, sunk depth, head diameter, alpha\_e |
| kPunchMark | 7 | Diameter |

|  |  |  |
| --- | --- | --- |
| **eAxisDirection** | | |
| kXDirection | 1 |  |
| kYDirection | 2 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **eAttributeCodes** | | | | |
| kBeamDenotation | |  | |  |
| kBeamMaterial | |  | |  |
| kBeamCoating | |  | |  |
| kBeamSinglePartNumber | |  | |  |
| kBeamMainPartNumber | |  | |  |
| kBeamSinglePartPrefix | |  | |  |
| kBeamMainPartPrefix | |  | |  |
| kBeamReferenceAxis | |  | |  |
| kBeamReferenceAxis\_Y | |  | |  |
| kBeamReferenceAxis\_Z | |  | |  |
| kBeamAngle | |  | |  |
| kBeamSection | |  | |  |
| kBeamOffset\_Y | |  | |  |
| kBeamOffset\_Z | |  | |  |
| kBeamRole | |  | |  |
| kBeamSystemlineRadius | |  | |  |
| kBeamAssembly | |  | |  |
| kBeamNote | |  | |  |
| kBeamItemNumber | |  | |  |
| kBeamSinglePartDetailStyle | |  | |  |
| kBeamMainPartDetailStyle | |  | |  |
| kBeamDeviation | |  | |  |
| kBeamMirroredCrossSection | |  | |  |
| kBeamSPUsedForCollisionCheck | |  | |  |
| kBeamSPUsedForNumbering | |  | |  |
| kBeamSPDisplayRestriction | |  | |  |
| kBeamSPExplicitQuantity | |  | |  |
| kBeamMPUsedForCollisionCheck | |  | |  |
| kBeamMPUsedForNumbering | |  | |  |
| kBeamMPDisplayRestriction | |  | |  |
| kBeamMPExplicitQuantity | |  | |  |
| kBeamUnfoldedThickness | |  | |  |
| kBeamUnfoldedPortioning | |  | |  |
| kBeamUnfoldedCrossSection | |  | |  |
|  | |  | |  |
| kPlateThickness | |  | |  |
| kPlatePortioning | |  | |  |
| kPlateLength | |  | |  |
| kPlateWidth | |  | |  |
| kPlateRadius | |  | |  |
| kPlateLengthIncrement | |  | |  |
| kPlateRadiusIncrement | |  | |  |
| kPlateDenotation | |  | |  |
| kPlateMaterial | |  | |  |
| kPlateCoating | |  | |  |
| kPlateSinglePartNumber | |  | |  |
| kPlateMainPartNumber | |  | |  |
| kPlateSinglePartPrefix | |  | |  |
| kPlateMainPartPrefix | |  | |  |
| kPlateTopInPositive\_Z\_Direction | |  | |  |
| kPlateRole | |  | |  |
| kPlateAssembly | |  | |  |
| kPlateNote | |  | |  |
| kPlateItemNumber | |  | |  |
| kPlateSinglePartDetailStyle | |  | |  |
| kPlateMainPartDetailStyle | |  | |  |
| kPlateType | |  | |  |
| kPlateOffset | |  | |  |
| kPlateSPUsedForCollisionCheck | |  | |  |
| kPlateSPUsedForNumbering | |  | |  |
| kPlateSPDisplayRestriction | |  | |  |
| kPlateSPExplicitQuantity | |  | |  |
| kPlateMPUsedForCollisionCheck | |  | |  |
| kPlateMPUsedForNumbering | |  | |  |
| kPlateMPDisplayRestriction | |  | |  |
| kPlateMPExplicitQuantity | |  | |  |
|  | |  | |  |
| kContourNotchLength | |  | |  |
| kContourNotchWidth | |  | |  |
| kContourNotchRadius | |  | |  |
| kContourNotchLengthIncrement | |  | |  |
| kContourNotchRadiusIncrement | |  | |  |
| kContourNotchGap | |  | |  |
| kContourNotchOffset | |  | |  |
| kContourNotch\_Z\_Clipping | |  | |  |
|  | |  | |  |
| kFeatContourLength | |  | |  |
| kFeatContourWidth | |  | |  |
| kFeatContourRadius | |  | |  |
| kFeatContourLengthIncrement | |  | |  |
| kFeatContourRadiusIncrement | |  | |  |
| kFeatContourGap | |  | |  |
| kFeatContourOffset | |  | |  |
|  | |  | |  |
| kChamferLength | |  | |  |
| kChamferWidth | |  | |  |
| kChamferLengthIncrement | |  | |  |
|  | |  | |  |
| kFilletRadius | |  | |  |
| kFilletRadiusIncrement | |  | |  |
| kFilletFilletType | |  | |  |
|  | |  | |  |
| kCircularBoltPatternNumberOfScrews | |  | |  |
| kCircularBoltPatternRadius | |  | |  |
|  | |  | |  |
| kNonRadialScrewBoltPatternNumberOfScrews\_X\_Direction | |  | |  |
| kNonRadialScrewBoltPatternNumberOfScrews\_Y\_Direction | |  | |  |
| kNonRadialScrewBoltPatternInnerScrewDistance\_X\_Direction | |  | |  |
| kNonRadialScrewBoltPatternInnerScrewDistance\_Y\_Direction | |  | |  |
| kNonRadialScrewBoltPatternOuterScrewDistance\_X\_Direction | |  | |  |
| kNonRadialScrewBoltPatternOuterScrewDistance\_Y\_Direction | |  | |  |
| kNonRadialScrewBoltPatternTotalLength\_X\_Direction | |  | |  |
| kNonRadialScrewBoltPatternTotalWidth\_Y\_Direction | |  | |  |
|  | |  | |  |
| kBoltPatternCommonDiameter | |  | |  |
| kBoltPatternCommonHoles | |  | |  |
| kBoltPatternCommonLength | |  | |  |
| kBoltPatternCommonGripLength | |  | |  |
| kBoltPatternCommonGripLengthAddition | |  | |  |
| kBoltPatternCommonHoleTolerance | |  | |  |
| kBoltPatternCommonStandard | |  | |  |
| kBoltPatternCommonGarnitur | |  | |  |
| kBoltPatternCommonAssembleLocation | |  | |  |
| kBoltPatternCommonMaterial | |  | |  |
| kBoltPatternCommonCoating | |  | |  |
| kBoltPatternCommonRole | |  | |  |
| kBoltPatternCommonDenotation | |  | |  |
| kBoltPatternCommonNote | |  | |  |
| kBoltPatternCommonAssembly | |  | |  |
| kBoltPatternCommonItemNumber | |  | |  |
| kBoltPatternCommonIgnoreMaxGap | |  | |  |
| kBoltPatternCommonInvertAble | |  | |  |
| kBoltPatternCommonSPUsedForCollisionCheck | |  | |  |
| kBoltPatternCommonSPUsedForBillOfMaterial | |  | |  |
| kBoltPatternCommonSPExplicitQuantity | |  | |  |
| kBoltPatternCommonSPUsedForNumbering | |  | |  |
| kBoltPatternHoleTypeForEachConnObject | |  | |  |
| kBoltPatternHoleToleranceForEachConnObject | |  | |  |
| kBoltPatternHoleSlotOffsetForEachConnObject | |  | |  |
| kBoltPatternHoleSlotLengthForEachConnObject | |  | |  |
| kBoltPatternHoleSlotDirectionForEachConnObject | |  | |  |
| kBoltPatternHoleSunkDepthForEachConnObject | |  | |  |
| kBoltPatternHoleAlphaForEachConnObject | |  | |  |
| kBoltPatternHoleTappingForEachConnObject | |  | |  |
| kBoltPatternHoleBackTaperThreadForEachConnObject | |  | |  |
| kBoltPatternHoleRightHandedThreadForEachConnObject | |  | |  |
| kBoltPatternHoleHeadDiameterForEachConnObject | |  | |  |
| kBoltPatternHoleAngleForEachConnObject | |  | |  |
| kBoltPatternHoleUsedForNumberingForEachConnObject | |  | |  |
|  | |  | |  |
| kBeamShorteningReferenceLength | |  | |  |
| kBeamShorteningAngle\_Y | |  | |  |
| kBeamShorteningAngle\_Z | |  | |  |
|  | |  | |  |
| kBeamContourNotchLength | |  | |  |
| kBeamContourNotchWidth | |  | |  |
| kBeamContourNotchRadius | |  | |  |
| kBeamContourNotchLengthIncrement | |  | |  |
| kBeamContourNotchRadiusIncrement | |  | |  |
| kBeamContourNotchGap | |  | |  |
| kBeamContourNotchOffset | |  | |  |
| kBeamContourNotch\_Z\_Clipping | |  | |  |
|  | |  | |  |
| kWeldPatternAdditionalText | |  | |  |
| kWeldPatternThickness | |  | |  |
| kWeldPatternLowerDistance | |  | |  |
| kWeldPatternSeamType | |  | |  |
| kWeldPatternSeamExecuting | |  | |  |
| kWeldPatternSeamShape | |  | |  |
| kWeldPatternAssembleLocation | |  | |  |
| kWeldPatternSeamLength | |  | |  |
| kWeldPatternIsClosed | |  | |  |
|  | |  | |  |
| kSpecialPartMaterial | |  | |  |
| kSpecialPartCoating | |  | |  |
| kSpecialPartSinglePartNumber | |  | |  |
| kSpecialPartMainPartNumber | |  | |  |
| kSpecialPartSinglePartPrefix | |  | |  |
| kSpecialPartMainPartPrefix | |  | |  |
| kSpecialPartDenotation | |  | |  |
| kSpecialPartAssembly | |  | |  |
| kSpecialPartNote | |  | |  |
| kSpecialPartItemNumber | |  | |  |
| kSpecialPartRole | |  | |  |
| kSpecialPartWeight | |  | |  |
| kSpecialPartBlockName | |  | |  |
| kSpecialPartScale | |  | |  |
| kSpecialPartSPUsedForCollisionCheck | |  | |  |
| kSpecialPartSPUsedForNumbering | |  | |  |
| kSpecialPartSPDisplayRestriction | |  | |  |
| kSpecialPartSPExplicitQuantity | |  | |  |
| kSpecialPartMPUsedForCollisionCheck | |  | |  |
| kSpecialPartMPUsedForNumbering | |  | |  |
| kSpecialPartMPDisplayRestriction | |  | |  |
| kSpecialPartMPExplicitQuantity | |  | |  |
|  | |  | |  |
| kHolePatternDiameter | |  | |  |
| kHolePatternType | |  | |  |
| kHolePatternSlotDirection | |  | |  |
| kHolePatternSlotLength | |  | |  |
| kHolePatternSunkDepth | |  | |  |
| kHolePatternHeadDiameter | |  | |  |
| kHolePatternAlpha\_E | |  | |  |
| kHolePatternTapping | |  | |  |
| kHolePatternUsedForNumbering | |  | |  |
|  | |  | |  |
| kHolePatternArrangementRadius | |  | |  |
| kHolePatternArrangementNumberOfElements | |  | |  |
| kHolePatternArrangement\_D\_X | |  | |  |
| kHolePatternArrangement\_D\_Y | |  | |  |
| kHolePatternArrangement\_N\_X | |  | |  |
| kHolePatternArrangement\_N\_Y | |  | |  |
| kHolePatternArrangement\_W\_X | |  | |  |
| kHolePatternArrangement\_W\_Y | |  | |  |
| kHolePatternArrangementLength | |  | |  |
| kHolePatternArrangementWidth | |  | |  |
|  | |  | |  |
| kConnectorPatternStandard | |  | |  |
| kConnectorPatternMaterial | |  | |  |
| kConnectorPatternCoating | |  | |  |
|  | |  | |  |
| kConnectorPatternArrangementRadius | |  | |  |
| kConnectorPatternArrangementNumberOfElements | |  | |  |
| kConnectorPatternArrangement\_D\_X | |  | |  |
| kConnectorPatternArrangement\_D\_Y | |  | |  |
| kConnectorPatternArrangement\_N\_X | |  | |  |
| kConnectorPatternArrangement\_N\_Y | |  | |  |
| kConnectorPatternArrangement\_W\_X | |  | |  |
| kConnectorPatternArrangement\_W\_Y | |  | |  |
| kConnectorPatternArrangementLength | |  | |  |
| kConnectorPatternArrangementWidth | |  | |  |
|  | |  | |  |
| kConnectorElementLength | |  | |  |
| kConnectorElementDiameter | |  | |  |
| **eValueType** | | | | |
| kRelative | 0 | | Relative value of Y | |
| kAbsolute | 1 | | Absolute value of Y | |

|  |  |  |
| --- | --- | --- |
| **eUIErrorCodes** | | |
| kNormal | 0 | Request succeeded |
| kError | 1 | Error |
| kCancel | 2 | User canceled request – Ctrl-C |
| kRejected | 3 | rejected request – invalid |
| kKwd | 4 | Keyword returned |
| kInputTruncated | 5 | Input did not fit in the buffer |
| kNoResult | 6 | No Result |

|  |  |  |
| --- | --- | --- |
| **eContourTypes** | | |
| kRectangle | 0 | Rectangular contour |
| kCircle | 1 | Circle |
| kPoly | 2 | Polygonal contour |
| kContourUndefined | 3 | Undefined contour |

|  |  |  |
| --- | --- | --- |
| **eNotchCornerType** | | |
| kStraight | 0 | Straight notch corner |
| kRound | 1 | Round notch corner |
| kBoringOut | 2 | Boring out notch corner |

|  |  |  |
| --- | --- | --- |
| **eAstColorMethod** | | |
| kAstColorByACI | 0 | Color defined by CAD Color |
| kAstColorByRGB | 1 | Color defined by Red, Green and Blue value |

|  |  |  |
| --- | --- | --- |
| **eArcType** | | |
| kFullCircle | 0 | Complete circle |
| kSimpleArc | 1 | Simple arc |
| kBadArc | 2 | Bad arc |

|  |  |  |
| --- | --- | --- |
| **eVertexInfoType** | | |
| kNoProcessing | -1 | Simple vertex – no arc information |
| kArcInfo | 0 | Arc information |

|  |  |  |
| --- | --- | --- |
| **ePolyValidity** | | |
| kVUnknown | 0 | Unknown validity |
| kVOk | 1 | Valid |
| kVSelfIntersecting | 2 | Self-intersecting |
| kVLessVertices | 3 | Too less vertices |

|  |  |  |
| --- | --- | --- |
| **eNotchXRotation** | | |
| kXRotAroundNotch | 0 | Rotation around an axis which is parallel to the beam’s axis and intersects the notch’s axis. |
| kXRotAroundBeam | 1 | Rotation around beam’s axis |

|  |  |  |
| --- | --- | --- |
| **eStructHeight** | | |
| kSTop | 0 | Top |
| kSFromTop | 1 | From Top |
| kSFromBottom | 2 | From Bottom |
| kSBetween | 3 | Between |

## Util Facet data

|  |  |  |
| --- | --- | --- |
| **eType** | | |
| kAModeler | 0 |  |
| kAcis | 1 |  |
| kBodyUnknown | 2 |  |

|  |  |  |
| --- | --- | --- |
| **eBodyContext - for more information about these values please see next pages** | | |
| kBodyNormal | 0 | Used in a getAstModeler call, will return a body with features |
| kBodyDetailed | 1 |  |
| kBodyHull | 2 |  |
| kBodyUnNotched | 3 | Use if you intend to make intersections of the body with lines, arcs, etc.  Returns a body without features. |
| kBodyNotchedHullBox | 4 |  |
| kBodySymbol | 5 |  |
| kBodyCollision | 6 |  |
| kBodyDetailing | 7 |  |
| kbodyExactDetailing | 8 |  |
| kBodyOverviewDetailing | 9 |  |
| kBodyOverviewDetailingExact | 10 |  |
| kBodyGTENormal | 11 |  |
| kBodyGTEDetailed | 12 |  |
| kBodyNC | 13 |  |
| kBodyTubeEndDev | 14 |  |

The following table illustrates the COM-API body contexts parameters:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Context** | **Hull** | **With Features** | **Detailed Feature** | **Welding Preps.** | **Collision** | **Exact** | **Symbol** | **Exact Profile Section** | **Deviation** | **Facets Number** | **Default Name** | **Note** |
| kBodyNormal | No | Yes | No | No | No | No | No | No | 1.0 | 16 | - |  |
| kBodyDetailing | No | Yes | No | No | No | No | No | No | 1.0 | 16 | - | same as kBodyNormal |
| kBodyGTEDetailed | No | Yes | No | No | No | No | No | No | 1.0 | 16 | - | obsolete |
| kBodyDetailed | No | Yes | Yes | No | No | Yes | No | No | 1.0 | 16 | - |  |
| kBodyHull | Yes | Yes | No | No | No | No | No | No | 1.0 | 16 | - |  |
| kBodyNotchedHullBox | Yes | Yes | No | No | No | No | No | No | 1.0 | 16 | - | same as kBodyHull |
| kBodyUnNotched | No | No | No | No | No | No | No | No | 1.0 | 16 | - |  |
| kBodySymbol | No | No | No | No | No | No | Yes | No | 1.0 | 16 | - |  |
| kBodyCollision | No | Yes | Yes | Yes | Yes | Yes | No | Yes | 0.5 | 16 | - |  |
| kbodyExactDetailing | No | Yes | No | Yes | No | Yes | No | Yes | 0.5 | 16 | - |  |
| kBodyOverviewDetailing | No | Yes | No | No | No | No | No | Yes | 1.0 | 4 | MinFacetsForCircleApproximations |  |
| kBodyOverviewDetailingExact | No | Yes | No | Yes | No | Yes | No | Yes | 3.0 | 4 | MinFacetsForCircleApproximations |  |
| kBodyGTENormal | No | Yes | No | Yes | No | No | No | No | 1.0 | 16 | - |  |
| kBodyNC | No | Yes | No | Yes | No | Yes | No | No | 0.5 | 16 | - |  |
| kBodyTubeEndDev | No | Yes | Yes | No | No | Yes | No | Yes | 0.5 | 16 | - |  |
| kBodyUnNotchedHullBox | Yes | No | No | No | No | No | No | No | 1.0 | 16 | - |  |

The following table illustrates the meaning of the parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Meaning** | **Note** |
| Hull | Get beam body using a square which represents the extents of the beam’s section | Applies only to beams. |
| With Features | Applies notches, shortenings (not holes) to a body. |  |
| Detailed Feature | Creates holes in the body (just holes – no other features) |  |
| Welding preparations | Subtracts welding preparations from plate body | Applies only to plates. |
| Collision | Gets a more precise body for bolts |  |
| Exact | Gets beam body using the exact section | Applies only to beams. |
| Symbol | Gets beam body which has ~1/3 of the beam total length – for simplified display | Applies only to beams |
| Exact profile section | Gets the beam using filleted section if the section has fillets | Applies only to beams |
| Deviation | The maximum accepted distance between a point from a segment approximating an arc and the arc itself |  |
| Facets number | Minimum number of segments used for approximating a full circle |  |
| Default name | Name of a default used for overwriting the “Facets number” value |  |

## Examples

Below you will find some examples and HOW TOs that might give you some hints about things you would like to do with your rule.

### Calling a subrule

This is a code sample written in C++ about how to call a subrule(or subjoint).



# Chapter 7 AstControls library

## AstUnitControl

For this control, the following methods and properties are available and only these methods and properties should be used in setting/getting information to/from the control:

.**DoubleValue** – control value (use ONLY for double type controls).

.**IntegerValue** – control value (use ONLY for integer type controls).

.**Text** – control value (use ONLY for string type controls).

.**EditType**

.**Enabled**

.**LabelDbKey** – UPS prompt number.

.**LabelLength**

.**Visible**

For numeric controls, you must use the DoubleValue / IntegerValue property, not the Text property, to get or set the current value of the control.

## AstComboTableControl

For this control, the following methods and properties are available and only these methods and properties should be used in setting/getting information to/from the control:

.**comboCaption** (displayed runname)

.**comboIndex** (order of runname in list, starting from 0 – not the key of the table record).

.**longKey** (key of record for current runname – use ONLY when table has an integer key).

.**stringKey** (key of record for current runname – use ONLY when table has a string key).

.**Enabled**

.**LabelKey** (UPS prompt number).

.**LabelLength**

.**Sort**

.**tableName** (linked table name).

.**Visible**

## AstCheckBoxControl

For this control, the following methods and properties are available and only these methods and properties should be used in setting/getting information to/from the control:

.**captionKey** (UPS prompt number).

.**Value**

## AstProfileControl

For this control, the following methods and properties are available and only these methods and properties should be used in setting/getting information to/from the control:

.**AppendAcceptedClass (class name)** – value from AstorProfiles.ProfileMasterTable, column TypeNameText.

.**AppendAcceptedClassGroup (class group)** – value from AstorProfiles.ProfileMasterTable, column SubtypeName.

.**Appearance**

.**CaptionClass** – UPS prompt number.

.**CaptionShowHideAllSections** – UPS prompt number.

.**CaptionTyp** – UPS prompt number.

.**CheckBoxOnLeft**

.**CurrentClass**

.**CurrentSection**

.**CurrentCrossSection**

.**CurrentProfileName**

.**DropHeight**

.**DropWidth**

.**Enabled**

.**LabelLength**

.**RemoveAcceptedClass**

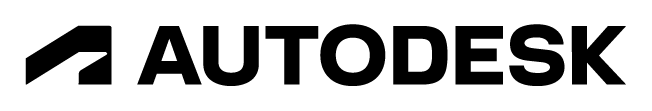
.**ShowHideAllSections**

.**SummaryDroppedDown**

.**SummaryRepresentation**

.**UseFilterClass**

.**Visib**

****