Contents

AS .NET API – 2015.1 changes

Class Application

public static int **RunCommand**(string *commandName*, ParametersContainer *params*)

//Can be used to run Advance Steel or AutoCAD commands. The command string syntax is like for the palette. Example: "^C^C_AstM4CrConByVb CornerBasePlate"
//Commands run asynchronously
public static int **RunCommand**(string *commandName*)

Class DatabaseManager

// This method returns the id of the Advance Steel objects "Representation" entity. It can be passed to various AutoCAD API methods.
public static ObjectId^ **GetReprId** (FilerObject *obj*)

//Exports elements to ACIS SAT format file. ids - Advance Steel objects ids to be exported. fileName - Input file name to write data to
public static **void ExportToACIS**(ObjectId[] *ids*, string *fileName*)

Class UserInteraction

public static **void PrintMess**(string *strMessage*)

public static int **MaxDrawingLimits**(out double *max*)

public static int **PutIntoSelectionSet**(List< ObjectId> *entityIds*)

public static double **GetDistance**(string *strMessage*)

public static double **GetDouble**(string *strMessage*)

public static int **GetInteger**(string *strMessage*, out int *nInteger*)

public static int **GetKeyword**(out string *strKwd*, string *strPrompt*, string *keyword_list*, out int *keyword_no*)

public static int **GetCornerPoint**(Point3d *point1*, string *message*, int *type*, out Point3d *point2*)

public static int **GetPolygon**(string *startPrompt*, string *nextPrompt*, out List< Point2d> *polygonVertices*)

public static int **GetPolygon**(string *startPrompt*, string *nextPrompt*, out List< Point3d> *polygonVerticesWCS*)

```
public static int GetPolyline(string startPrompt, string nextPrompt, out List< Point3d> c)

//Returns -1 when the user cancelled
public static int GetYesOrNo(string strPrompt)
public static int GetCurrentSelectionSet(out List< ObjectId> ssIds);
public static double GetAngle(Point3d ptOrigin, string strMessage)
public static int GetCurrentOrNewSSet(out List< ObjectId> ids, string strPrompt)

//Returns the object id of the selected object
public static ObjectId SelectObject ()

//Returns a list of the selected objects ids
public static List<ObjectId> SelectObjects ()
```

Class OpenDatabase

```
//Added the following functions for reading and writing model defaults:
// The set methods also update the defaults.
public static int setDwgDataDefault(string strDefault, int nVal)
public static int getDwgDataDefault(string strDefault, out int nVal)

public static int setDwgDataDefault(string strDefault, double dVal)
public static int getDwgDataDefault(string strDefault, out double dVal)

public static int setDwgDataDefault(string strDefault, string strVal)
public static int getDwgDataDefault(string strDefault, out string strVal)
```

Class CompoundStraightBeam

```
//Throws exceptions of type ASAPIException when the creation fails
public ObjectId CreateComponents(string CompoundClassName, string CompoundTypeName)
public ObjectId CreateComponents(string CompoundClassName)
public ObjectId CreateComponents()
//We can change the section of a beam using this method
//We can get section info using ProfSectionType and ProfSectionName read-only properties of class Beam
//Example for calls of ChangeProfile () method:
```

- ChangeProfile("C nach DIN", "C100X2"); // for straight beam
- ChangeProfile("HEA DIN18800-1","HEA100"); // for straight beam
- ChangeProfile("Compound2WDiag","Default CISC"); // for compound beam
- ChangeProfile("TaperedBeam", "Default"); // for tapered beam

public **int** **ChangeProfile** (**string** *newSectionType*, **string** *newSectionName*)

Class FoldedPlate

// By this method, we provide access to folded plate relations – see PlateFoldRelation:
public **PlateFoldRelation []** **GetAllFoldRelations** ()

Class Grating

// The function creates a variable grating along the given polygon, places it in the database and adds the necessary fillet features. The class and name of the grating are taken from the database.
public static **Grating** **Create** (**Polyline3d** *poly*)

Class Plate

//The function creates a plate along the given polygon, places it in the database and adds the necessary fillet features.
public static **Plate** **Create** (**Polyline3d** *poly*)

Class PlateBase

//The plate must be in the database. The function adds the necessary fillet features.
public **int** **SetOuterContour** (**Polyline3d** *polyContour*)

Class KernelServices

//The method has been extended to allow passing an array of template ids and template locations for which excel exports will be created;
//ModelFileName – represents the full path to the dwg
//idsSelectedObjects – array of object ids for which the model extract will be created; if it is empty it will use the current selection or the full model for the model extract

//If no template info is given the method will create a model extract and return the summary info.
public static **BOMSummaryInfo** **RunBOMProcess** (**OpenDatabase** *pDatabase*, **string** *modelFileName*,
ObjectId [] *idsSelectedObjects*, **BOMTemplateInfo []** *templates*)

Class GeomUtils

//This method calculates the center of gravity for a set of given objects
public static **Point3d** **CalculateBalancePoint** (**ObjectId** **[]** *objects*)

Class AtomicElement

//Exports current element to ACIS SAT format file. fileName - Input file name to write data to. Returns 1 if successful, or 0 if not
public **int** **ExportToACIS** (**string** *fileName*)

//Create an ACIS solid/s geometrically identical to current element. Returns an object array with objects ids of the solid entities
public **ObjectId []** **CreateACIS** ()

```
//Convert from AS ObjectId to AutoCAD ObjectId
Autodesk.AutoCAD.DatabaseServices.ObjectId acadId = new
Autodesk.AutoCAD.DatabaseServices.ObjectId (id.AsOldId ());
```

Class UserAutoConstructionObject

//Save all joint parameters into the filer
public **Filer** **Save** ()

//Load joint parameters from the input filer
public **void Load** (**Filer** *filer*)

Class Filer

Introduced new class named Filer to allow access to the joint parameters.

//Returns all joint parameters map [key, object value].
public **Dictionary<string, object>** **GetItems** ()

```
//Returns number of filer items
public int Length ()


//Returns item value for the input name. Ex: double dValue = (double) filer.ReadItem
(“PlateLength”)
public object ReadItem (string itemName)

//Returns the filer version
public int ReadVersion ()

//Write item to the joint filer. Ex filer.WriteItem (100.0, “PlateLength”);
public void WriteItem (object value, string itemName)

//Write filer version
public void WriteVersion (int nVersion)
```

AS COM API – 2015.1 changes

Geometry API

Interface IGeomUtils

```
//Introduced new interface named IGeomUtils
//calculate the center of gravity for a set of given objects.
HRESULT CalculateBalancePoint (/* [in]*/ IAstObjectsArr* pArr, /*[out, retval]*/IPoint3d**
ptCenter)
```

Modeling API

Interface IBeam

Change the section of a beam using new method

```
//Example for calls of ChangeProfile (…) method:
```

- ChangeProfile("C nach DIN", "C100X2");   // for straight beam
  ChangeProfile("HEA DIN18800-1","HEA100");   // for straight beam
- ChangeProfile("Compound2WDiag","Default CISC"); // for compound beam
  ChangeProfile("TaperedBeam", "Default");   // for tapered beam

```
HRESULT ChangeProfile (/*[in]*/BSTR sectionType, /*[in]*/BSTR sectionName)
```

```
//Gets beam section info using new method
HRESULT GetProfile (/*[out]*/BSTR* className,/*[out]*/BSTR* sectionName)
```

UserInterface API

Interface IAstUI

```
//The outErrCode parameter will return either kUICancel if the user cancels the request, or
kUINormal otherwise.
HRESULT AcquireYesOrNo (/*[in]*/int nPrompt, /*[out, retval]*/VARIANT_BOOL* pVal)
```

Others

Interface IAstCreator

```
//Creates an instance of IGeomUtils

CreateGeomUtils (/*[out, retval]*/ IGeomUtils** pGeomUtils)
```

Interface IJoint

```
//Creates an instance of IGeomUtils

HRESULT CreateGeomUtils (/*[out, retval]*/ IGeomUtils** pGeomUtils)
```

```
//Save all joint parameters into the filer
HRESULT Save (/*[out, retval] */IFiler** pFiler)

//Load joint parameters from the input filer
HRESULT Load (/*[in] */IFiler* pFiler)
```

Interface IFiler

```
//Returns all joint parameters map.
Ex: Array items = filer.GetItems (); IFilerItem item = items.GetValue (0) as IFilerItem;
HRESULT GetItems (/*[out, retval] */SAFEARRAY** pItems)
```

Interface IFilerItem

Introduced new interface named IFilerItem to allow access to the joint parameters.

```
HRESULT get_ItemKey (/* [retval][out] */ BSTR *pKey)
HRESULT get_ItemValue (/* [retval][out] */ VARIANT *pValue)
```