Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 1 2 3]

▼ N **AlibreScript**	
▼ N **API**	
C **AssembledPart**	A part that is in an assembly
C **AssembledSubAssembly**	A subassembly that is in an assembly
C **Assembly**	An assembly
C **Axis**	An axis
C **Bspline**	Defines a **Bspline** that can be added to 2D sketches
C **Bspline3D**	Defines a **Bspline** that can be added to 3D sketches
C **Circle**	Describes a 2D circle, which can be added to 2D sketches
C **CircularArc**	Describes a 2D circular arc, which can be added to 2D sketches
C **CircularArc3D**	Describes a 3D circular arc, which can be added to 3D sketches
C **Configuration**	Describes a configuration
C **CSharp**	Provides access to the full Alibre Design **API** by running C# code See the Advanced **API** manual for details
C **Edge**	Describes an edge (can be filleted, chamfered, swept)
C **Ellipse**	Describes an ellipse used in 2D sketches
C **EllipticalArc**	Describes an elliptical arc used in 2D sketches
C **Face**	Describes a face (can be filleted, chamfered, used for sketches, used for loft cross sections)
C **Feature**	Describes a feature of an object, e.g. boss, cut
C **GearSketch**	A 2D sketch containing an involute gear profile. Can be treated as a regular sketch
C **GlobalParameters**	A set of global parameters
C **Line**	Describes a 2D line, which can be added to 2D sketches
C **Line3D**	Describes a 3D line, which can be added to 3D sketches
C **Material**	**Material** densities in kg/cm3
C **Parameter**	Describes a parameter
C **Part**	Object that represents a part
C **Plane**	A design plane. Can be used for creating sketches
C **Point**	A design point
C **Polyline**	A line constructed from a set of line segments
C **Polyline3D**	A 3D line constructed from a set of line segments
C **PolylinePoint**	A single point in a polyline
C **PolylinePoint3D**	A single point in a polyline for 3D sketches
C **Sketch**	A 2D sketch
C **Sketch3D**	3D sketch
	A 2D sketch point

C **SketchPoint**	
C **SketchPoint3D**	A 3D sketch point
C **ThreeD**	3D mathematical operations
C **TwoD**	2D mathematical operations
C **Vertex**	Describes a vertex
C **Windows**	Graphical user interface creation and interaction

AlibreScript Namespace Reference

AlibreScript.API Namespace Reference

Classes

class **AssembledPart**

A part that is in an assembly More...

class **AssembledSubAssembly**

A subassembly that is in an assembly More...

class **Assembly**

An assembly More...

class **Axis**

An axis More...

class **Bspline**

Defines a **Bspline** that can be added to 2D sketches More...

class **Bspline3D**

Defines a **Bspline** that can be added to 3D sketches More...

class **Circle**

Describes a 2D circle, which can be added to 2D sketches More...

class **CircularArc**

Describes a 2D circular arc, which can be added to 2D sketches More...

class **CircularArc3D**

Describes a 3D circular arc, which can be added to 3D sketches More...

class **Configuration**

Describes a configuration More...

class **CSharp**

Provides access to the full Alibre Design **API** by running C# code See the Advanced **API** manual for details More...

class **Edge**

Describes an edge (can be filleted, chamfered, swept) More...

class **Ellipse**

Describes an ellipse used in 2D sketches More...

class **EllipticalArc**

Describes an elliptical arc used in 2D sketches More...

class **Face**

Describes a face (can be filleted, chamfered, used for sketches, used for loft cross sections) More...

class **Feature**

Describes a feature of an object, e.g. boss, cut More...

class **GearSketch**

A 2D sketch containing an involute gear profile. Can be treated as a regular sketch More...

class **GlobalParameters**

A set of global parameters More...

class **Line**

Describes a 2D line, which can be added to 2D sketches More...

class **Line3D**

Describes a 3D line, which can be added to 3D sketches More...

class **Material**

Material densities in kg/cm3 More...

class **Parameter**

Describes a parameter More...

class **Part**

Object that represents a part More...

class **Plane**

A design plane. Can be used for creating sketches More...

class **Point**

A design point More...

class **Polyline**

A line constructed from a set of line segments More...

class **Polyline3D**

A 3D line constructed from a set of line segments More...

class **PolylinePoint**

A single point in a polyline More...

class **PolylinePoint3D**

A single point in a polyline for 3D sketches More...

class **Sketch**

A 2D sketch More...

class **Sketch3D**

3D sketch More...

class **SketchPoint**

A 2D sketch point More...

class **SketchPoint3D**

A 3D sketch point More...

class **ThreeD**

3D mathematical operations More...

class **TwoD**

2D mathematical operations More...

class **Vertex**

Describes a vertex More...

class **Windows**

Graphical user interface creation and interaction More...

Enumerations

enum **GuideCurveTypes** { **Global** , **Local** , **Tangent** }

Type of guide curve More...

enum **LockTypes** {

None , **HideNewAnnotations** , **HideNewDesignGeometry** , **HideNewInclusions** ,

HideNewSketches , **LockActiveSectionView** , **LockColorProperties** , **LockComponentConfig** ,

LockParameterValues , **LockPropertyValues** , **SuppressNewComponents** , **SuppressNewConstraints** ,

SuppressNewFeatures , **All**

}

Type of configuration lock More...

enum **ParameterTypes** { **Distance** , **Angle** , **Count** , **Scale** }

Type of parameter More...

enum **ParameterUnits** {

Unitless , **Millimeters** , **Centimeters** , **Meters** ,

Inches , **Feet** , **FeetInches** , **Degrees** ,

DegreesMinutes , **DegreesMinutesSeconds** , **Radians** , **Kilograms** ,

Grams , **Pounds**

}

Units of parameters More...

enum **ThumbnailOptions** {

None , **BiggerSizeOk** , **InMemoryOnly** , **IconOnly** ,

ThumbnailOnly , **InCacheOnly**

}

enum **UnitTypes** { **Millimeters** , **Centimeters** , **Inches** }

Supported units More...

enum **WindowsInputTypes** {

String , **Integer** , **Real** , **Boolean** ,

Face , **Faces** , **Plane** , **Planes** ,
Edge , **Edges** , **Vertex** , **Vertices** ,
Point , **Points** , **Axis** , **Axes** ,
Sketch , **Sketches** , **Sketch3D** , **File** ,
Label , **SaveFile** , **StringList** , **Image** ,
Url , **Folder** , **Part** , **Assembly**
}

Type of Windows input More...

Enumeration Type Documentation

◆ GuideCurveTypes

enum **GuideCurveTypes**

Type of guide curve

Enumerator	
Global	Global guide curve
Local	Local guide curve
Tangent	Tangent guide curve

◆ LockTypes

enum **LockTypes**

Type of configuration lock

Enumerator	
None	No lock
HideNewAnnotations	Hide new annotations
HideNewDesignGeometry	Hide new design geometry
HideNewInclusions	Hide new inclusions
HideNewSketches	Hide new sketches
LockActiveSectionView	Lock active section view
LockColorProperties	Lock color properties
LockComponentConfig	Lock component configuration
LockParameterValues	Lock parameter values
LockPropertyValues	Lock property values
SuppressNewComponents	Suppress new components
SuppressNewConstraints	Suppress new constraints
SuppressNewFeatures	Suppress new features
All	All

◆ ParameterTypes

enum **ParameterTypes**

Type of parameter

Enumerator	
Distance	Distance parameter
Angle	Angle parameter
Count	Count parameter
Scale	Scale parameter

◆ ParameterUnits

enum **ParameterUnits**

Units of parameters

Enumerator	
Unitless	Unitless
Millimeters	Millimeters
Centimeters	Centimeters
Meters	Meters
Inches	Inches
Feet	Feet
FeetInches	Feet and inches
Degrees	Degrees
DegreesMinutes	Degrees and minutes
DegreesMinutesSeconds	Degrees, minutes and seconds
Radians	Radians
Kilograms	Kilograms
Grams	Grams
Pounds	Pounds

◆ ThumbnailOptions

enum **ThumbnailOptions**

Enumerator	
None	No options
BiggerSizeOk	Biggers size is OK
InMemoryOnly	Store in memory only
IconOnly	Icon only
ThumbnailOnly	Thumbnail only
InCacheOnly	Store in cache only

◆ UnitTypes

enum **UnitTypes**

Supported units

Enumerator	
Millimeters	Millimeters
Centimeters	Centimeters
Inches	Inches

◆ WindowsInputTypes

enum **WindowsInputTypes**

Type of **Windows** input

Enumerator	
String	Text
Integer	Integer value
Real	Decimal (floating point) value
Boolean	true or false
Face	A face
Faces	A list of faces Optional settings: • Input box height in pixels (integer)
Plane	A plane
Planes	A list of planes Optional settings: • Input box height in pixels (integer)
Edge	An edge
Edges	A list of edges Optional settings: • Input box height in pixels (integer)
Vertex	A vertex
Vertices	A list of vertices Optional settings: • Input box height in pixels (integer)
Point	A point
Points	A list of points Optional settings: • Input box height in pixels (integer)
Axis	An axis
Axes	A list of axes Optional settings: • Input box height in pixels (integer)

Sketch	A 2D sketch
Sketches	A list of 2D sketches Optional settings: • Input box height in pixels (integer)
Sketch3D	A 3D sketch
File	Path and file Optional settings: • Dialog title (string) • File type filters (string), example: 'Text File|*.txt|All Files|*.*' • Default file extension (string), example: '.txt'
Label	Text label
SaveFile	Save a file Optional settings: • Dialog title (string) • File type filters (string), example: 'Text File|*.txt|All Files|*.*' • Default file extension (string), example: '.txt'
StringList	List of text Optional settings: • Default string to show (string)
Image	Image The default value can be a filename or a list of bytes representing an image [0x11, 0x22, ...] (use ImagetoPython.py) Optional settings: • Width in pixels (integer)
Url	Website address
Folder	Folder Optional settings: • Description (string)
Part	**Part**
Assembly	**Assembly**

AssembledPart Class Reference

A part that is in an assembly More...

Inherits **Part**, IInstance, and IAssembled.

Public Member Functions

new **Point**	**AddPoint** (string **Name**, **Edge** TargetEdge, double Ratio) Adds a point on an edge More...
new **Point**	**AddPoint** (string **Name**, IAxis AxisOrEdge, IPlane PlaneOrFace) Adds a point at the the intersection of a axis or edge and a plane or face More...
new **Point**	**AddPoint** (string **Name**, IAxis AxisOrEdge1, IAxis AxisOrEdge2) Adds a point at the intersection or two axes or edges More...
new **Point**	**AddPoint** (string **Name**, IPlane PlaneOrFace1, IPlane PlaneOrFace2, IPlane PlaneOrFace3) Adds a point at the intersection of three planes or faces More...
new **Point**	**AddPoint** (string **Name**, IPoint PointOrVertex, double XOffset, double YOffset, double ZOffset) Adds a point at an offset to a point or a vertex More...
new **Point**	**AddPoint** (string **Name**, IPoint PointOrVertex1, IPoint PointOrVertex2, double Ratio) Adds a point between two points/vertices More...
new **Point**	**AddPoint** (string **Name**, IPoint SourcePointOrVertex, IPlane TargetPlaneOrFace, double XOffset, double YOffset) Adds a point by projecting a point or vertex onto a plane or face More...
new **Point**	**AddPointFromCircularEdge** (string **Name**, **Edge** TargetEdge) Adds a point at the center of a circular edge More...
new **Point**	**AddPointFromToroidalFace** (string **Name**, **Face** TargetFace) Adds a point at the center of a toroidal face More...
List []	**AssemblyPointtoPartPoint** (List [] AssemblyPoint) Converts a point in the assembly coordinate system into a point in the part coordinate system More...
Assembly	**GetAssembly** () Gets the assembly for the part More...
List []	**GetAssemblyBoundingBox** () Gets the bounding box for the part as eight points in the assembly coordinate system More...
List []	**GetAssemblyVertices** () Gets a python list of the current vertices in the part in the assembly coordinate system More...
new **Configuration**	**GetConfiguration** (string **Name**) Gets a configuration with a specific name More...
new **Edge**	**GetEdge** (string **Name**) Gets an edge using it's name "Edge<n>" More...
new List []	**GetEdges** () Gets a python list of the current edges in the part More...
new **Face**	**GetFace** (string **Name**)

Gets a face using it's name "Face<n>" More...

new List [] **GetFaces** ()

Gets a python list of the current faces in the part More...

IADOccurrence **GetMappedOccurrence** (IADAssemblySession **Assembly**)

Gets the occurrence of the part mapped into the occurrence structure of a specific assembly This occurrence can be used to create constraints in the specific assembly using the part More...

List [] **PartPointtoAssemblyPoint** (List [] PartPoint)

Converts a point in the part coordinate system into a point in the assembly coordinate system More...

▶ **Public Member Functions inherited from Part**

Properties

new List [] **Configurations** `[get]`

List of configurations defined on the part

new string **Name** `[get]`

Name of the assembled part

▶ **Properties inherited from Part**

Additional Inherited Members

▶ **Public Types inherited from Part**

Extrusion directions - extrude along... More...

Extrusion end conditions - extrude until... More...

Supported file types More...

Detailed Description

A part that is in an assembly

Member Function Documentation

◆ AddPoint() `[1/7]`

```
new Point AddPoint ( string  Name,
                     Edge    TargetEdge,
                     double  Ratio
                   )
```

Adds a point on an edge

Parameters

Name	Name of point
TargetEdge	The edge to create the point on
Ratio	Ratio along the edge from 0.0 -> 1.0

Returns

The created point

◆ AddPoint() [2/7]

```
new Point AddPoint ( string  Name,
                     IAxis   AxisOrEdge,
                     IPlane  PlaneOrFace
                   )
```

Adds a point at the the intersection of a axis or edge and a plane or face

Parameters

Name	Name of point
AxisOrEdge	**Axis** or edge
PlaneOrFace	**Plane** or face

Returns

The created point

◆ AddPoint() [3/7]

```
new Point AddPoint ( string  Name,
                     IAxis   AxisOrEdge1,
                     IAxis   AxisOrEdge2
                   )
```

Adds a point at the intersection or two axes or edges

Parameters

Name	Name of point
AxisOrEdge1	First axis or edge
AxisOrEdge2	Second axis or edge

Returns

The created point

◆ AddPoint() [4/7]

```
new Point AddPoint ( string  Name,
                     IPlane  PlaneOrFace1,
                     IPlane  PlaneOrFace2,
                     IPlane  PlaneOrFace3
                   )
```

Adds a point at the intersection of three planes or faces

Parameters

Name	Name of point
PlaneOrFace1	First plane or face
PlaneOrFace2	Second plane or face
PlaneOrFace3	Third plane or face

Returns

The created point

◆ AddPoint() [5/7]

```
new Point AddPoint ( string  Name,
                     IPoint  PointOrVertex,
                     double  XOffset,
                     double  YOffset,
                     double  ZOffset
                   )
```

Adds a point at an offset to a point or a vertex

Parameters

Name	Name of point
PointOrVertex	**Point** or vertex
XOffset	X offse
YOffset	Y offset
ZOffset	Z offset

Returns

The created point

◆ AddPoint() [6/7]

```
new Point AddPoint ( string  Name,
                     IPoint  PointOrVertex1,
                     IPoint  PointOrVertex2,
                     double  Ratio
                   )
```

Adds a point between two points/vertices

Parameters

Name	Name of point
PointOrVertex1	First point or vertex
PointOrVertex2	Second point or vertex
Ratio	Ratio of distance between points/vertices

Returns

The created point

◆ AddPoint() [7/7]

```
new Point AddPoint ( string  Name,
                     IPoint  SourcePointOrVertex,
                     IPlane  TargetPlaneOrFace,
                     double  XOffset,
                     double  YOffset
                   )
```

Adds a point by projecting a point or vertex onto a plane or face

Parameters

Name	Name of point
SourcePointOrVertex	**Point** or vertex to project
TargetPlaneOrFace	**Plane** or face to project onto
XOffset	X offset to apply to point once projected
YOffset	Y offset to apply to point once projected

Returns

The created point

◆ AddPointFromCircularEdge()

```
new Point AddPointFromCircularEdge ( string  Name,
                                     Edge    TargetEdge
                                   )
```

Adds a point at the center of a circular edge

Parameters

Name	Name of point
TargetEdge	The edge to use for creating the point

Returns

The created point

◆ AddPointFromToroidalFace()

```
new Point AddPointFromToroidalFace ( string  Name,
                                     Face    TargetFace
                                   )
```

Adds a point at the center of a toroidal face

Parameters

Name	Name of point
TargetFace	Toroidal face to use in creating the point

Returns

The created point

◆ AssemblyPointtoPartPoint()

```
List [] AssemblyPointtoPartPoint ( List []  AssemblyPoint )
```

Converts a point in the assembly coordinate system into a point in the part coordinate system

Parameters

AssemblyPoint	**Point** [X, Y, Z] in the assembly coordinate system

Returns

Point [X, Y, Z] in the part coordinate system

◆ GetAssembly()

```
Assembly GetAssembly ( )
```

Gets the assembly for the part

Returns

Assembly or None if no assembly

◆ GetAssemblyBoundingBox()

List [] GetAssemblyBoundingBox ( )

Gets the bounding box for the part as eight points in the assembly coordinate system

Returns

Python list of eight points as [P1, P2, ... P8]. Each point is [X, Y, Z]

◆ GetAssemblyVertices()

List [] GetAssemblyVertices ( )

Gets a python list of the current vertices in the part in the assembly coordinate system

Returns

Python list of vertices in assembly coordinates [[X1, Y1, Z1], ... [Xn, Yn, Zn]]

◆ GetConfiguration()

new **Configuration** GetConfiguration (string Name)

Gets a configuration with a specific name

Parameters

Name Name of confguration

Returns

Configuration object

◆ GetEdge()

new **Edge** GetEdge (string Name)

Gets an edge using it's name "Edge<n>"

Parameters

Name Name of edge

Returns

Edge if found

◆ GetEdges()

new List [] GetEdges ( )

Gets a python list of the current edges in the part

Returns

Python list of edges

◆ GetFace()

new **Face** GetFace (string Name)

Gets a face using it's name "Face<n>"

Parameters

Name Name of face

Returns

Face if found

◆ GetFaces()

new List [] GetFaces ( )

Gets a python list of the current faces in the part

Returns

Python list of faces

◆ GetMappedOccurrence()

IADOccurrence GetMappedOccurrence (IADAssemblySession Assembly)

Gets the occurrence of the part mapped into the occurrence structure of a specific assembly This occurrence can be used to create constraints in the specific assembly using the part

Parameters

Assembly **Assembly** for occurrence structure

Returns

Mapped occurrence or null if not found

◆ PartPointtoAssemblyPoint()

List [] PartPointtoAssemblyPoint (List [] PartPoint)

Converts a point in the part coordinate system into a point in the assembly coordinate system

Parameters

PartPoint **Point** [X, Y, Z] in the part coordinate system

Returns

Point [X, Y, Z] in the assembly coordinate system

AssembledSubAssembly Class Reference

A subassembly that is in an assembly More...

Inherits **Assembly**, IInstance, and IAssembled.

Public Member Functions

new **Configuration**	**GetConfiguration** (string **Name**) Gets a configuration with a specific name More...
IADOccurrence	**GetMappedOccurrence** (IADAssemblySession **Assembly**) Gets the occurrence of the sub-assembly mapped into the occurrence structure of a specific assembly This occurrence can be used to create constraints in the specific sub-assembly using the part More...
Assembly	**GetSelectionAssembly** () The assembly that the edge was selected on Only valid when a selection has been made More...

▶ **Public Member Functions inherited from Assembly**

Properties

new List []	**Configurations** `[get]` A list of configurations defined on the assembly
new string	**Name** `[get]` Name of the subassembly

▶ **Properties inherited from Assembly**

Additional Inherited Members

▶ **Public Types inherited from Assembly**

Assembly constraint bounds types More...

Detailed Description

A subassembly that is in an assembly

Member Function Documentation

◆ GetConfiguration()

new **Configuration** GetConfiguration (string Name)

Gets a configuration with a specific name

Parameters

Name Name of confguration

Returns

Configuration object

◆ GetMappedOccurrence()

IADOccurrence GetMappedOccurrence (IADAssemblySession Assembly)

Gets the occurrence of the sub-assembly mapped into the occurrence structure of a specific assembly This occurrence can be used to create constraints in the specific sub-assembly using the part

Parameters

Assembly **Assembly** for occurrence structure

Returns

Mapped occurrence or null if not found

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

Assembly Class Reference

An assembly More...

Inherited by **AssembledSubAssembly**.

Public Types

enum **ConstraintBoundsType** { **Equals** , **LessOrEquals** , **GreaterOrEquals** , **Between** }

Assembly constraint bounds types More...

Public Member Functions

Assembly (string Folder, string **Name**)

Opens an existing assembly More...

Assembly (string Folder, string **Name**, bool HideEditor)

Opens an existing assembly, optionally hiding the editor More...

Assembly (string **Name**)

Creates a new assembly More...

Assembly (string **Name**, bool CreateNew)

Creates a new assembly or accesses an already opened assembly More...

Assembly (string **Name**, bool CreateNew, bool HideEditor)

Creates a new assembly or accesses an already opened assembly, optionally hiding the editor More...

void **AddAlignConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB)

Adds a simple alignment constraint between two planes/faces/axes/edges/points More...

void **AddAlignConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**)

Adds a simple alignment constraint between two planes/faces/axes/edges/points More...

void **AddAlignConstraint2** (double Distance1, double Distance2, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**, **ConstraintBoundsType** BoundsType)

Adds an alignment constraint between two planes/faces/axes/edges/points Uses bounds type More...

void **AddAngleConstraint** (double Angle, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB)

Adds an angle constraint between two planes/faces/axes/edges/points More...

void **AddAngleConstraint** (double Angle, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**)

Adds a simple angle constraint between two planes/faces/axes/edges/points More...

void

	AddAngleConstraint2 (double Angle1, double Angle2, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**, **ConstraintBoundsType** BoundsType) Adds an angle constraint between two planes/faces/axes/edges/points Uses bounds type More...
Axis	**AddAxis** (string **Name**, List [] Point1, List [] Point2) Creates an axis based on two points More...
Axis	**AddAxis** (string **Name**, ISketchSurface Plane1, ISketchSurface Plane2) Creates an axis based on the intersection of two planes/faces More...
Configuration	**AddConfiguration** (string **Name**) Adds a configuration to the assembly More...
Configuration	**AddConfiguration** (string **Name**, string BaseConfigurationName) Adds a configuration to the assembly using another configuration as a base More...
void	**AddFastenerConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds a fastner constraint More...
void	**AddFastenerConstraint2** (double Distance1, double Distance2, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**, **ConstraintBoundsType** BoundsType) Adds a fastner constraint More...
void	**AddGearConstraint** (double RatioA, double RatioB, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds a gear constraint using ratio RatioA:RatioB More...
void	**AddMateConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB) Adds a simple mate constraint between two planes/faces/axes/edges/points More...
void	**AddMateConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds a simple mate constraint between two planes/faces/axes/edges/points More...
void	**AddMateConstraint2** (double Distance1, double Distance2, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**, **ConstraintBoundsType** BoundsType) Adds a mate constraint between two planes/faces/axes/edges/points Uses bounds type More...
AssembledPart	**AddNewPart** (string **Name**, double X, double Y, double Z) Adds a new part to the assembly More...
AssembledSubAssembly	**AddNewSubAssembly** (string **Name**, double X, double Y, double Z) Adds a new sub-assembly to the assembly More...
void	**AddOrientConstraint** (double Value, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB) Adds an orient constraint between two planes/faces/axes/edges/points More...
void	

	AddOrientConstraint (double Value, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds an orient constraint between two planes/faces/axes/edges/points More...
Parameter	**AddParameter** (string **Name**, **ParameterTypes** Type, double Value) Adds a parameter to the assembly More...
Parameter	**AddParameter** (string **Name**, **ParameterTypes** Type, string Equation) Adds a parameter to the assembly NOTE: DOESN'T SEEM TO WORK IN GD V16 - THROWS EXCEPTION ABOUT TRANSACTION ALREADY BEING OPEN More...
AssembledPart	**AddPart** (**Part Part**) Adds a part to the assembly at the origin More...
AssembledPart	**AddPart** (**Part Part**, double OffsetX, double OffsetY, double OffsetZ) Adds a part to the assembly More...
AssembledPart	**AddPart** (**Part Part**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a part to the assembly More...
AssembledPart	**AddPart** (string **FileName**) Adds a part to the assembly at the origin More...
AssembledPart	**AddPart** (string **FileName**, double OffsetX, double OffsetY, double OffsetZ) Adds a part to the assembly More...
AssembledPart	**AddPart** (string **FileName**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a part to the assembly More...
AssembledPart	**AddPart** (string Folder, string **Name**) Adds a part to the assembly at the origin More...
AssembledPart	**AddPart** (string Folder, string **Name**, double OffsetX, double OffsetY, double OffsetZ) Adds a part to the assembly More...
AssembledPart	**AddPart** (string Folder, string **Name**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a part to the assembly More...
Plane	**AddPlane** (string **Name**, List [] NormalVector, List [] PointonPlane) Adds a plane using a normal vector and a point on the plane More...
Plane	**AddPlane** (string **Name**, List [] Point1, List [] Point2, List [] Point3) Creates a plane using three points More...
Plane	**AddPlane** (string **Name**, ISketchSurface SourcePlane, **Axis** RotationAxis, double Angle) Creates a new plane at an angle to an existing plane More...
Plane	**AddPlane** (string **Name**, ISketchSurface SourcePlane, double Offset) Creates a plane based on the offset from an existing plane More...
Point	**AddPoint** (string **Name**, double X, double Y, double Z) Adds a point to the assembly More...

Point	**AddPoint** (string **Name**, **Edge** TargetEdge, double Ratio) Add a point on an edge More...
Point	**AddPoint** (string **Name**, IAxis AxisOrEdge, IPlane PlaneOrFace) Add a point at the the intersection of a axis or edge and a plane or face More...
Point	**AddPoint** (string **Name**, IAxis AxisOrEdge1, IAxis AxisOrEdge2) Add a point at the intersection or two axes or edges More...
Point	**AddPoint** (string **Name**, IPlane PlaneOrFace1, IPlane PlaneOrFace2, IPlane PlaneOrFace3) Add a point at the intersection of three planes or faces More...
Point	**AddPoint** (string **Name**, IPoint PointOrVertex, double XOffset, double YOffset, double ZOffset) Add a point at an offset to a point or a vertex More...
Point	**AddPoint** (string **Name**, IPoint PointOrVertex1, IPoint PointOrVertex2, double Ratio) Add a point between two points/vertices More...
Point	**AddPoint** (string **Name**, IPoint SourcePointOrVertex, IPlane TargetPlaneOrFace, double XOffset, double YOffset) Add a point by projecting a point or vertex onto a plane or face More...
Point	**AddPointFromCircularEdge** (string **Name**, **Edge** TargetEdge) Adds a point at the center of a circular edge More...
Point	**AddPointFromToroidalFace** (string **Name**, **Face** TargetFace) Adds a point at the center of a toroidal face More...
void	**AddPoints** (string Prefix, List [] Points) Adds a set of points to the part More...
void	**AddRackAndPinionConstraint** (double PitchDiameter, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds a rack and pinion constraint More...
void	**AddScrewConstraint** (double ThreadPitch, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool IsReversed, string **Name**) Adds a screw constraint More...
AssembledSubAssembly	**AddSubAssembly** (**Assembly Assembly**) Adds a sub-assembly to the assembly at the origin More...
AssembledSubAssembly	**AddSubAssembly** (**Assembly Assembly**, double OffsetX, double OffsetY, double OffsetZ) Adds a sub-assembly to the assembly More...
AssembledSubAssembly	**AddSubAssembly** (**Assembly Assembly**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a sub-assembly to the assembly More...
AssembledSubAssembly	**AddSubAssembly** (string **FileName**) Adds a sub-assembly to the assembly at the origin More...
AssembledSubAssembly	**AddSubAssembly** (string **FileName**, double OffsetX, double OffsetY, double OffsetZ) Adds a sub-assembly to the assembly More...
AssembledSubAssembly	

	AddSubAssembly (string **FileName**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a sub-assembly to the assembly More...
AssembledSubAssembly	**AddSubAssembly** (string Folder, string **Name**) Adds a sub-assembly to the assembly at the origin More...
AssembledSubAssembly	**AddSubAssembly** (string Folder, string **Name**, double OffsetX, double OffsetY, double OffsetZ) Adds a sub-assembly to the assembly More...
AssembledSubAssembly	**AddSubAssembly** (string Folder, string **Name**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Adds a sub-assembly to the assembly More...
void	**AddTangentConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool Outside) Adds a tangent constraint between two planes/faces/axes/edges/points More...
void	**AddTangentConstraint** (double Distance, IAssembled PartorAssemblyA, IConstrainable ItemA, IAssembled PartorAssemblyB, IConstrainable ItemB, bool Outside, bool IsReversed, string **Name**) Adds a tangent constraint between two planes/faces/axes/edges/points More...
void	**AnchorPart** (**AssembledPart Part**) Anchors a part More...
void	**AnchorPart** (string **Name**) Anchors a part More...
void	**AnchorSubAssembly** (string **Name**) Anchors a sub-assembly More...
void	**Close** () Closes the assembly If it is unsaved then changes will be lost
string	**CreateUniqueName** (string BaseName) Creates a unique name that can be used to safely add a part or subassembly to the assembly if the names used in the assembly are not known in advance More...
UnitTypes	**DisplayUnits** () Gets the display units for the assembly More...
AssembledPart	**DuplicatePart** (**AssembledPart Part**, double OffsetX, double OffsetY, double OffsetZ) Duplicates a part in the assembly More...
AssembledPart	**DuplicatePart** (**AssembledPart Part**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Duplicates a part in the assembly More...
AssembledPart	**DuplicatePart** (string **Name**, double OffsetX, double OffsetY, double OffsetZ) Duplicates a part in the assembly More...
AssembledPart	**DuplicatePart** (string **Name**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Duplicates a part in the assembly More...

AssembledSubAssembly	**DuplicateSubAssembly** (**AssembledSubAssembly** SubAssembly, double OffsetX, double OffsetY, double OffsetZ) Duplicates a sub-assembly in the assembly More...
AssembledSubAssembly	**DuplicateSubAssembly** (**AssembledSubAssembly** SubAssembly, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Duplicates a sub-assembly in the assembly More...
AssembledSubAssembly	**DuplicateSubAssembly** (string **Name**, double OffsetX, double OffsetY, double OffsetZ) Duplicates a sub-assembly in the assembly More...
AssembledSubAssembly	**DuplicateSubAssembly** (string **Name**, double OffsetX, double OffsetY, double OffsetZ, double AngleX, double AngleY, double AngleZ, bool TranslationFirst) Duplicates a sub-assembly in the assembly More...
void	**ExportBIP** (string **FileName**) Exports a keyshot file More...
void	**ExportIGES** (string **FileName**) Exports the assembly as a IGES file More...
void	**ExportSAT** (string **FileName**, int Version, bool SaveColors) Exports the assembly as a SAT file More...
void	**ExportSTEP203** (string **FileName**) Exports the assembly as a STEP 203 file More...
void	**ExportSTEP214** (string **FileName**) Exports the assembly as a STEP 214 file More...
void	**ExportSTL** (string **FileName**) Exports the assembly as an STL file More...
Configuration	**GetActiveConfiguration** () Gets the currently active configuration More...
Axis	**GetAxis** (string **Name**) Gets an axis from an axis name More...
Configuration	**GetConfiguration** (string **Name**) Gets a configuration with a specific name More...
string	**GetCustomProperty** (string **Name**) Gets the value of a custonm property More...
Parameter	**GetParameter** (string **Name**) Gets a parameter with a specific name More...
AssembledPart	**GetPart** (string **Name**) Gets a part in the assembly More...
List []	**GetPartOrientation** (**AssembledPart Part**) Gets the orientation of a part in an assembly More...
List []	**GetPartOrientation** (string PartName) Gets the orientation of a part in an assembly More...

Plane	**GetPlane** (string **Name**) Gets a plane using the name of the plane More...
Point	**GetPoint** (string **Name**) Gets a point on the assembly using the point name. The point must have been created in a script More...
AssembledSubAssembly	**GetSubAssembly** (string **Name**) Gets a sub-assembly in the assembly More...
IronPython.Runtime.PythonDictionary	**GetUserData** (string **Name**) Gets user data More...
void	**HidePart** (**AssembledPart Part**) Hides a part More...
void	**HidePart** (string **Name**) Hides a part More...
void	**HideSubAssembly** (string **Name**) Hides a sub-assembly More...
void	**MovePart** (**AssembledPart Part**, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a part More...
void	**MovePart** (string **Name**, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a part More...
void	**MoveParts** (List [] Names, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a set of parts More...
void	**MoveSubAssemblies** (List [] Names, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a set of sub-assemblies More...
void	**MoveSubAssembly** (**AssembledSubAssembly** SubAssembly, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a sub-assembly More...
void	**MoveSubAssembly** (string **Name**, double OffsetX, double OffsetY, double OffsetZ, bool ApplyConstraints) Moves a sub-assembly More...
void	**PauseUpdating** () Pauses updating the assembly user interface
void	**Regenerate** () Regenerates the assembly
void	**ResumeUpdating** () Resumes updating the assembly user interface
void	**RotatePart** (**AssembledPart Part**, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a part More...

void **RotatePart** (string **Name**, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a part More...

void **RotateParts** (List [] Names, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a set of parts More...

void **RotateSubAssemblies** (List [] Names, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a set of sub-assemblies More...

void **RotateSubAssembly** (**AssembledSubAssembly** SubAssembly, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a sub-assembly More...

void **RotateSubAssembly** (IADOccurrence AssemOcc, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a sub-assembly More...

void **RotateSubAssembly** (string **Name**, double AngleX, double AngleY, double AngleZ, bool ApplyConstraints)

Rotates a sub-assembly More...

void **Save** ()

Saves the assembly using the current path and file name

void **Save** (string Folder)

Saves the assembly to a specific folder More...

void **SaveAll** (string Folder)

Save the assembly and all parts/sub-assemblies to a folder More...

void **SaveAs** (string Folder, string NewName)

Saves the assembly to a specific folder with a new name More...

void **SaveSnapshot** (string **FileName**, int Width, int Height, bool UseAspectRatio, bool UseWidthandHeight)

Saves the current view as a bitmap image More...

void **SaveThumbnail** (string **FileName**, int Width, int Height)

Saves a thumbnail image of the assembly More...

void **SetCustomProperty** (string **Name**, string Value)

Sets the value of a custom property The custom property must already be defined on the assembly or defined on the user's PC More...

void **SetUserData** (string **Name**, IronPython.Runtime.PythonDictionary Dict)

Sets user data More...

void **ShowPart** (**AssembledPart Part**)

Shows a part More...

void **ShowPart** (string **Name**)

Shows a part More...

void	**ShowSubAssembly** (string **Name**) Shows a sub-assembly More...
void	**SuppressPart** (**AssembledPart Part**) Suppresses a part More...
void	**SuppressPart** (string **Name**) Suppresses a part More...
void	**SuppressSubAssembly** (string **Name**) Suppresses a sub-assembly More...
void	**UnanchorPart** (**AssembledPart Part**) Un-anchors a part More...
void	**UnanchorPart** (string **Name**) Un-anchors a part More...
void	**UnanchorSubAssembly** (string **Name**) Un-anchors a sub-assembly More...
void	**UnsuppressPart** (**AssembledPart Part**) Un-suppresses a part More...
void	**UnsuppressPart** (string **Name**) Un-suppresses a part More...
void	**UnsuppressSubAssembly** (string **Name**) Un-suppresses a sub-assembly More...

Properties

string	**Comment** `[get, set]` Comment property
List []	**Configurations** `[get]` A list of configurations defined on the assembly
string	**CostCenter** `[get, set]` Cost center property
string	**CreatedBy** `[get, set]` Created By property
string	**CreatedDate** `[get, set]` Created Date property
string	**CreatingApplication** `[get, set]` Creating Application property
double	**Density** `[get, set]` Density of the part
string	**Description** `[get, set]` Description of the part

string **DocumentNumber** `[get, set]`

Document Number property

string **EngineeringApprovalDate** `[get, set]`

Engineering Approval Date property

string **EngineeringApprovedBy** `[get, set]`

Engineering Approved By property

string **EstimatedCost** `[get, set]`

Estimated Cost property

string **ExtendedMaterialInformation** `[get, set]`

Material (extended information) property

string **FileName** `[get]`

Path and filename of the assembly

string **Keywords** `[get, set]`

Keywords property

string **LastAuthor** `[get, set]`

Last Author property

string **LastUpdateDate** `[get, set]`

Last Update Date property

string **ManufacturingApprovedBy** `[get, set]`

Manufacturing Approved By property

string **ManufacturingApprovedDate** `[get, set]`

Product property

string **Material** `[get, set]`

Material of the part

string **ModifiedInformation** `[get, set]`

Modified Information property

string **Name** `[get]`

Name of the assembly

string **Number** `[get, set]`

User-defined number for the part

Point **Origin** `[get]`

Gets the origin (language independent)

List [] **Parameters** `[get]`

A list of parameters defined on the assembly

List [] **Parts** `[get]`

A list of parts defined on the assembly

string **Product** `[get, set]`

Product property

string **ReceivedFrom** `[get, set]`

Received From property

string **Revision** `[get, set]`

Revision property

List [] **Selections** `[get]`

Gets the currently selected items as [ItemA, ItemB, ...] Supports subassemblies, parts, faces, edges, vertices, planes, axes and points

string **StockSize** `[get, set]`

Stock Size property

List [] **SubAssemblies** `[get]`

A list of subassemblies defined on the assembly

string **Supplier** `[get, set]`

Supplier property

string **Title** `[get, set]`

Title property

string **Vendor** `[get, set]`

Vendor property

string **WebLink** `[get, set]`

Web Link property

Axis **XAxis** `[get]`

Gets the X-axis (language independent)

Plane **XYPlane** `[get]`

Gets the XY-plane (language independent)

Axis **YAxis** `[get]`

Gets the Y-axis (language independent)

Plane **YZPlane** `[get]`

Gets the YZ-plane (language independent)

Axis **ZAxis** `[get]`

Gets the Z-axis (language independent)

Plane **ZXPlane** `[get]`

Gets the ZX-plane (language independent)

Detailed Description

An assembly

Member Enumeration Documentation

◆ ConstraintBoundsType

enum **ConstraintBoundsType**

Assembly constraint bounds types

Enumerator	
Equals	Value must match
LessOrEquals	Value must be less than or equals
GreaterOrEquals	Value must be greater than or equals
Between	Value must be between two values

Constructor & Destructor Documentation

◆ Assembly() [1/5]

Assembly (string Folder,
string Name
)

Opens an existing assembly

Parameters

Folder Folder containing assembly

Name Name of assembly to open

◆ Assembly() [2/5]

```
Assembly ( string  Folder,
           string  Name,
           bool    HideEditor
         )
```

Opens an existing assembly, optionally hiding the editor

Parameters

Folder Folder containing assembly

Name Name of assembly to open

HideEditor True to hide the editor

◆ Assembly() [3/5]

```
Assembly ( string  Name )
```

Creates a new assembly

Parameters

Name Name of new assembly

◆ Assembly() [4/5]

```
Assembly ( string  Name,
           bool    CreateNew
         )
```

Creates a new assembly or accesses an already opened assembly

Parameters

Name Name of assembly to create or access

CreateNew True to create a new assembly, false to access an opened assembly

◆ Assembly() [5/5]

```
Assembly ( string  Name,
           bool    CreateNew,
           bool    HideEditor
         )
```

Creates a new assembly or accesses an already opened assembly, optionally hiding the editor

Parameters

Name	Name of assembly to create or access
CreateNew	True to create a new assembly, false to access an opened assembly
HideEditor	True to hide the editor (only valid if assembly is not already open)

Member Function Documentation

◆ AddAlignConstraint() [1/2]

```
void AddAlignConstraint ( double          Distance,
                          IAssembled      PartorAssemblyA,
                          IConstrainable  ItemA,
                          IAssembled      PartorAssemblyB,
                          IConstrainable  ItemB
                        )
```

Adds a simple alignment constraint between two planes/faces/axes/edges/points

Parameters

Distance	Alignment distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain

◆ AddAlignConstraint() [2/2]

```
void AddAlignConstraint ( double          Distance,
                          IAssembled      PartorAssemblyA,
                          IConstrainable  ItemA,
                          IAssembled      PartorAssemblyB,
                          IConstrainable  ItemB,
                          bool            IsReversed,
                          string          Name
                        )
```

Adds a simple alignment constraint between two planes/faces/axes/edges/points

Parameters

Distance	Alignment distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddAlignConstraint2()

```
void AddAlignConstraint2 ( double              Distance1,
                           double              Distance2,
                           IAssembled          PartorAssemblyA,
                           IConstrainable      ItemA,
                           IAssembled          PartorAssemblyB,
                           IConstrainable      ItemB,
                           bool                IsReversed,
                           string              Name,
                           ConstraintBoundsType BoundsType
                         )
```

Adds an alignment constraint between two planes/faces/axes/edges/points Uses bounds type

Parameters

Distance1	Align distance
Distance2	Second distance for 'between' bounds type or zero if not used
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint
BoundsType	The bounds type to use

◆ AddAngleConstraint() [1/2]

```
void AddAngleConstraint ( double          Angle,
                          IAssembled      PartorAssemblyA,
                          IConstrainable  ItemA,
                          IAssembled      PartorAssemblyB,
                          IConstrainable  ItemB
                        )
```

Adds an angle constraint between two planes/faces/axes/edges/points

Parameters

Angle	Angle in degrees
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain

◆ AddAngleConstraint() [2/2]

```
void AddAngleConstraint ( double          Angle,
                          IAssembled      PartorAssemblyA,
                          IConstrainable  ItemA,
                          IAssembled      PartorAssemblyB,
                          IConstrainable  ItemB,
                          bool            IsReversed,
                          string          Name
                        )
```

Adds a simple angle constraint between two planes/faces/axes/edges/points

Parameters

Angle	Angle in degrees
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddAngleConstraint2()

```
void AddAngleConstraint2 ( double              Angle1,
                           double              Angle2,
                           IAssembled          PartorAssemblyA,
                           IConstrainable      ItemA,
                           IAssembled          PartorAssemblyB,
                           IConstrainable      ItemB,
                           bool                IsReversed,
                           string              Name,
                           ConstraintBoundsType BoundsType
                         )
```

Adds an angle constraint between two planes/faces/axes/edges/points Uses bounds type

Parameters

Angle1	Angle for constraint
Angle2	Second angle for 'between' bounds type or zero if not used
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint
BoundsType	The bounds type to use

◆ AddAxis() [1/2]

```
Axis AddAxis ( string  Name,
               List []  Point1,
               List []  Point2
             )
```

Creates an axis based on two points

Parameters

Name Name of axis

Point1 First point

Point2 Second point

Returns

New axis

◆ AddAxis() [2/2]

```
Axis AddAxis ( string            Name,
               ISketchSurface  Plane1,
               ISketchSurface  Plane2
             )
```

Creates an axis based on the intersection of two planes/faces

Parameters

Name Name of axis

Plane1 First plane/face

Plane2 Second plane/face

Returns

New **Axis**

◆ AddConfiguration() [1/2]

Configuration AddConfiguration (string Name)

Adds a configuration to the assembly

Parameters

Name Name of configuration

Returns

New configuration

◆ AddConfiguration() [2/2]

Configuration AddConfiguration (string Name,
string BaseConfigurationName
)

Adds a configuration to the assembly using another configuration as a base

Parameters

Name Name of configuration

BaseConfigurationName Name of base configuration to use

Returns

New configuration

◆ AddFastenerConstraint()

```
void AddFastenerConstraint ( double          Distance,
                             IAssembled      PartorAssemblyA,
                             IConstrainable  ItemA,
                             IAssembled      PartorAssemblyB,
                             IConstrainable  ItemB,
                             bool            IsReversed,
                             string          Name
                           )
```

Adds a fastner constraint

Parameters

Distance	Fastener to surface mate distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddFastenerConstraint2()

```
void AddFastenerConstraint2 ( double                Distance1,
                              double                Distance2,
                              IAssembled            PartorAssemblyA,
                              IConstrainable        ItemA,
                              IAssembled            PartorAssemblyB,
                              IConstrainable        ItemB,
                              bool                  IsReversed,
                              string                Name,
                              ConstraintBoundsType  BoundsType
                            )
```

Adds a fastner constraint

Parameters

Distance1	Fastener to surface mate distance
Distance2	Second distance for 'between' bounds type or zero if not used
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint
BoundsType	The bounds type to use

◆ AddGearConstraint()

```
void AddGearConstraint ( double          RatioA,
                         double          RatioB,
                         IAssembled      PartorAssemblyA,
                         IConstrainable  ItemA,
                         IAssembled      PartorAssemblyB,
                         IConstrainable  ItemB,
                         bool            IsReversed,
                         string          Name
                       )
```

Adds a gear constraint using ratio RatioA:RatioB

Parameters

RatioA	First value in gear ratio
RatioB	Second value in gear ratio
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddMateConstraint() [1/2]

```
void AddMateConstraint ( double          Distance,
                         IAssembled      PartorAssemblyA,
                         IConstrainable  ItemA,
                         IAssembled      PartorAssemblyB,
                         IConstrainable  ItemB
                       )
```

Adds a simple mate constraint between two planes/faces/axes/edges/points

Parameters

Distance	Mate distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain

◆ AddMateConstraint() [2/2]

```
void AddMateConstraint ( double          Distance,
                         IAssembled      PartorAssemblyA,
                         IConstrainable  ItemA,
                         IAssembled      PartorAssemblyB,
                         IConstrainable  ItemB,
                         bool            IsReversed,
                         string          Name
                       )
```

Adds a simple mate constraint between two planes/faces/axes/edges/points

Parameters

Distance	Mate distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddMateConstraint2()

```
void AddMateConstraint2 ( double              Distance1,
                          double              Distance2,
                          IAssembled          PartorAssemblyA,
                          IConstrainable      ItemA,
                          IAssembled          PartorAssemblyB,
                          IConstrainable      ItemB,
                          bool                IsReversed,
                          string              Name,
                          ConstraintBoundsType BoundsType
                        )
```

Adds a mate constraint between two planes/faces/axes/edges/points Uses bounds type

Parameters

Distance1	Mate distance
Distance2	Second distance for 'between' bounds type or zero if not used
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint
BoundsType	The bounds type to use

◆ AddNewPart()

```
AssembledPart AddNewPart ( string  Name,
                           double  X,
                           double  Y,
                           double  Z
                         )
```

Adds a new part to the assembly

Parameters

Name Name of the new part

X X location of part

Y Y location of part

Z Z location of part

Returns

New part

◆ AddNewSubAssembly()

```
AssembledSubAssembly AddNewSubAssembly ( string  Name,
                                         double  X,
                                         double  Y,
                                         double  Z
                                       )
```

Adds a new sub-assembly to the assembly

Parameters

Name Name of the new assembly

X X location of assembly

Y Y location of assembly

Z Z location of assembly

Returns

New part

◆ AddOrientConstraint() [1/2]

```
void AddOrientConstraint ( double          Value,
                           IAssembled      PartorAssemblyA,
                           IConstrainable  ItemA,
                           IAssembled      PartorAssemblyB,
                           IConstrainable  ItemB
                         )
```

Adds an orient constraint between two planes/faces/axes/edges/points

Parameters

Value	Value
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain

◆ AddOrientConstraint() [2/2]

```
void AddOrientConstraint ( double          Value,
                           IAssembled      PartorAssemblyA,
                           IConstrainable  ItemA,
                           IAssembled      PartorAssemblyB,
                           IConstrainable  ItemB,
                           bool            IsReversed,
                           string          Name
                         )
```

Adds an orient constraint between two planes/faces/axes/edges/points

Parameters

Value	Value
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddParameter() [1/2]

```
Parameter AddParameter ( string          Name,
                         ParameterTypes  Type,
                         double          Value
                       )
```

Adds a parameter to the assembly

Parameters

Name Name of parameter

Type Type of parameter

Value Value for parameter

Returns

New parameter

◆ AddParameter() [2/2]

```
Parameter AddParameter ( string          Name,
                         ParameterTypes  Type,
                         string          Equation
                       )
```

Adds a parameter to the assembly NOTE: DOESN'T SEEM TO WORK IN GD V16 - THROWS EXCEPTION ABOUT TRANSACTION ALREADY BEING OPEN

Parameters

Name Name of parameter

Type Type of parameter

Equation Equation for parameter

Returns

New parameter

◆ AddPart() [1/9]

AssembledPart AddPart (**Part** Part)

Adds a part to the assembly at the origin

Parameters

Part **Part** to add

Returns

The added part

◆ AddPart() [2/9]

AssembledPart AddPart (**Part** Part,
double OffsetX,
double OffsetY,
double OffsetZ
)

Adds a part to the assembly

Parameters

Part **Part** to add

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The added part

◆ AddPart() [3/9]

```
AssembledPart AddPart ( Part    Part,
                        double  OffsetX,
                        double  OffsetY,
                        double  OffsetZ,
                        double  AngleX,
                        double  AngleY,
                        double  AngleZ,
                        bool    TranslationFirst
                      )
```

Adds a part to the assembly

Parameters

Part	**Part** to add
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added part

◆ AddPart() [4/9]

```
AssembledPart AddPart ( string  FileName )
```

Adds a part to the assembly at the origin

Parameters

FileName	Path and name of part to open

Returns

The added part

◆ AddPart() [5/9]

```
AssembledPart AddPart ( string  FileName,
                        double  OffsetX,
                        double  OffsetY,
                        double  OffsetZ
                      )
```

Adds a part to the assembly

Parameters

FileName Path and name of part to open

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The added part

◆ AddPart() [6/9]

```
AssembledPart AddPart ( string  FileName,
                        double  OffsetX,
                        double  OffsetY,
                        double  OffsetZ,
                        double  AngleX,
                        double  AngleY,
                        double  AngleZ,
                        bool    TranslationFirst
                      )
```

Adds a part to the assembly

Parameters

FileName	Path and name of part to open
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added part

◆ AddPart() [7/9]

```
AssembledPart AddPart ( string  Folder,
                        string  Name
                      )
```

Adds a part to the assembly at the origin

Parameters

Folder	Folder containing part
Name	Name of part to open

Returns

The added part

◆ AddPart() [8/9]

```
AssembledPart AddPart ( string  Folder,
                        string  Name,
                        double  OffsetX,
                        double  OffsetY,
                        double  OffsetZ
                      )
```

Adds a part to the assembly

Parameters

Folder	Folder containing part
Name	Name of part to open
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset

Returns

The added part

◆ AddPart() [9/9]

```
AssembledPart AddPart ( string  Folder,
                        string  Name,
                        double  OffsetX,
                        double  OffsetY,
                        double  OffsetZ,
                        double  AngleX,
                        double  AngleY,
                        double  AngleZ,
                        bool    TranslationFirst
                      )
```

Adds a part to the assembly

Parameters

Folder	Folder containing part
Name	Name of part to open
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added part

◆ AddPlane() [1/4]

```
Plane AddPlane ( string  Name,
                 List []  NormalVector,
                 List []  PointonPlane
               )
```

Adds a plane using a normal vector and a point on the plane

Parameters

Name	Name of plane to add
NormalVector	Normal vector as a list [nx, ny, nz]. Does not need to be a unit vector
PointonPlane	A point on the plane as a list [px, py, pz]

Returns

Created plane

◆ AddPlane() [2/4]

```
Plane AddPlane ( string  Name,
                 List []  Point1,
                 List []  Point2,
                 List []  Point3
               )
```

Creates a plane using three points

Parameters

Name	Name of plane
Point1	**Point** on plane
Point2	**Point** on plane
Point3	**Point** on plane

Returns

Created plane

◆ AddPlane() [3/4]

```
Plane AddPlane ( string          Name,
                 ISketchSurface  SourcePlane,
                 Axis            RotationAxis,
                 double          Angle
               )
```

Creates a new plane at an angle to an existing plane

Parameters

Name	Name of new plane
SourcePlane	Plane/face to use as basis for new plane
RotationAxis	**Axis** of rotation for new plane
Angle	Angle of new plane in degrees

Returns

New plane

◆ AddPlane() [4/4]

```
Plane AddPlane ( string          Name,
                 ISketchSurface  SourcePlane,
                 double          Offset
               )
```

Creates a plane based on the offset from an existing plane

Parameters

Name	Name of plane
SourcePlane	Plane/face to use as basis
Offset	Offset from basis plane in currently chosen units

Returns

Created plane

◆ AddPoint() [1/8]

```
Point AddPoint ( string  Name,
                 double  X,
                 double  Y,
                 double  Z
               )
```

Adds a point to the assembly

Parameters

Name	Name of new point
X	X coordinate
Y	Y coordinate
Z	Z coordinate

Returns

The new point

◆ AddPoint() [2/8]

```
Point AddPoint ( string  Name,
                 Edge    TargetEdge,
                 double  Ratio
               )
```

Add a point on an edge

Parameters

Name	Name of point
TargetEdge	The edge to create the point on
Ratio	Ratio along the edge from 0.0 -> 1.0

Returns

The created point

◆ AddPoint() [3/8]

```
Point AddPoint ( string  Name,
                 IAxis   AxisOrEdge,
                 IPlane  PlaneOrFace
               )
```

Add a point at the the intersection of a axis or edge and a plane or face

Parameters

Name	Name of point
AxisOrEdge	**Axis** or edge
PlaneOrFace	**Plane** or face

Returns

The created point

◆ AddPoint() [4/8]

```
Point AddPoint ( string  Name,
                 IAxis   AxisOrEdge1,
                 IAxis   AxisOrEdge2
               )
```

Add a point at the intersection or two axes or edges

Parameters

Name	Name of point
AxisOrEdge1	First axis or edge
AxisOrEdge2	Second axis or edge

Returns

The created point

◆ AddPoint() [5/8]

```
Point AddPoint ( string  Name,
                 IPlane  PlaneOrFace1,
                 IPlane  PlaneOrFace2,
                 IPlane  PlaneOrFace3
               )
```

Add a point at the intersection of three planes or faces

Parameters

Name	Name of point
PlaneOrFace1	First plane or face
PlaneOrFace2	Second plane or face
PlaneOrFace3	Third plane or face

Returns

The created point

◆ AddPoint() [6/8]

```
Point AddPoint ( string  Name,
                 IPoint  PointOrVertex,
                 double  XOffset,
                 double  YOffset,
                 double  ZOffset
               )
```

Add a point at an offset to a point or a vertex

Parameters

Name	Name of point
PointOrVertex	**Point** or vertex
XOffset	X offse
YOffset	Y offset
ZOffset	Z offset

Returns

The created point

◆ AddPoint() [7/8]

```
Point AddPoint ( string  Name,
                 IPoint  PointOrVertex1,
                 IPoint  PointOrVertex2,
                 double  Ratio
               )
```

Add a point between two points/vertices

Parameters

Name	Name of point
PointOrVertex1	First point or vertex
PointOrVertex2	Second point or vertex
Ratio	Ratio of distance between points/vertices

Returns

The created point

◆ AddPoint() [8/8]

```
Point AddPoint ( string  Name,
                 IPoint  SourcePointOrVertex,
                 IPlane  TargetPlaneOrFace,
                 double  XOffset,
                 double  YOffset
               )
```

Add a point by projecting a point or vertex onto a plane or face

Parameters

Name	Name of point
SourcePointOrVertex	**Point** or vertex to project
TargetPlaneOrFace	**Plane** or face to project onto
XOffset	X offset to apply to point once projected
YOffset	Y offset to apply to point once projected

Returns

The created point

◆ AddPointFromCircularEdge()

```
Point AddPointFromCircularEdge ( string  Name,
                                 Edge    TargetEdge
                               )
```

Adds a point at the center of a circular edge

Parameters

Name Name of point

TargetEdge The edge to use for creating the point

Returns

The created point

◆ AddPointFromToroidalFace()

```
Point AddPointFromToroidalFace ( string  Name,
                                 Face    TargetFace
                               )
```

Adds a point at the center of a toroidal face

Parameters

Name Name of point

TargetFace Toroidal face to use in creating the point

Returns

The created point

◆ AddPoints()

```
void AddPoints ( string  Prefix,
                 List []  Points
               )
```

Adds a set of points to the part

Parameters

Prefix Prefix for the point names

Points List of points [x1,y1,z1, ..., xn,yn,zn]

◆ AddRackAndPinionConstraint()

```
void AddRackAndPinionConstraint ( double          PitchDiameter,
                                  IAssembled      PartorAssemblyA,
                                  IConstrainable  ItemA,
                                  IAssembled      PartorAssemblyB,
                                  IConstrainable  ItemB,
                                  bool            IsReversed,
                                  string          Name
                                )
```

Adds a rack and pinion constraint

Parameters

PitchDiameter	Pitch diameter
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddScrewConstraint()

```
void AddScrewConstraint ( double          ThreadPitch,
                          IAssembled      PartorAssemblyA,
                          IConstrainable  ItemA,
                          IAssembled      PartorAssemblyB,
                          IConstrainable  ItemB,
                          bool            IsReversed,
                          string          Name
                        )
```

Adds a screw constraint

Parameters

ThreadPitch	Pitch of thread
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AddSubAssembly() [1/9]

AssembledSubAssembly AddSubAssembly (**Assembly** Assembly)

Adds a sub-assembly to the assembly at the origin

Parameters

Assembly Assembly	to add

Returns

The added assembly

◆ AddSubAssembly() [2/9]

```
AssembledSubAssembly AddSubAssembly ( Assembly  Assembly,
                                      double    OffsetX,
                                      double    OffsetY,
                                      double    OffsetZ
                                    )
```

Adds a sub-assembly to the assembly

Parameters

Assembly **Assembly** to add

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The added assembly

◆ AddSubAssembly() [3/9]

```
AssembledSubAssembly AddSubAssembly ( Assembly  Assembly,
                                      double    OffsetX,
                                      double    OffsetY,
                                      double    OffsetZ,
                                      double    AngleX,
                                      double    AngleY,
                                      double    AngleZ,
                                      bool      TranslationFirst
                                    )
```

Adds a sub-assembly to the assembly

Parameters

Assembly	Sub-assembly to add
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added sub-assembly

◆ AddSubAssembly() [4/9]

```
AssembledSubAssembly AddSubAssembly ( string  FileName )
```

Adds a sub-assembly to the assembly at the origin

Parameters

FileName	Path and name of sub-assembly to open

Returns

The added sub-assembly

◆ AddSubAssembly() [5/9]

```
AssembledSubAssembly AddSubAssembly ( string  FileName,
                                      double  OffsetX,
                                      double  OffsetY,
                                      double  OffsetZ
                                    )
```

Adds a sub-assembly to the assembly

Parameters

FileName Path and name of sub-assembly to open

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The added sub-assembly

◆ AddSubAssembly() [6/9]

```
AssembledSubAssembly AddSubAssembly ( string  FileName,
                                      double  OffsetX,
                                      double  OffsetY,
                                      double  OffsetZ,
                                      double  AngleX,
                                      double  AngleY,
                                      double  AngleZ,
                                      bool    TranslationFirst
                                    )
```

Adds a sub-assembly to the assembly

Parameters

FileName	Path and name of sub-asembly to open
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added assembly

◆ AddSubAssembly() [7/9]

```
AssembledSubAssembly AddSubAssembly ( string  Folder,
                                      string  Name
                                    )
```

Adds a sub-assembly to the assembly at the origin

Parameters

Folder	Folder containing sub-assembly
Name	Name of sub-assembly to open

Returns

The added sub-assembly

◆ AddSubAssembly() [8/9]

```
AssembledSubAssembly AddSubAssembly ( string Folder,
                                      string Name,
                                      double OffsetX,
                                      double OffsetY,
                                      double OffsetZ
                                    )
```

Adds a sub-assembly to the assembly

Parameters

Folder Folder containing sub-assembly

Name Name of sub-assembly to open

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The added sub-assembly

◆ AddSubAssembly() [9/9]

```
AssembledSubAssembly AddSubAssembly ( string   Folder,
                                      string   Name,
                                      double   OffsetX,
                                      double   OffsetY,
                                      double   OffsetZ,
                                      double   AngleX,
                                      double   AngleY,
                                      double   AngleZ,
                                      bool     TranslationFirst
                                    )
```

Adds a sub-assembly to the assembly

Parameters

Folder	Folder containing sub-assembly
Name	Name of sub-assembly to open
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The added sub-assembly

◆ AddTangentConstraint() [1/2]

```
void AddTangentConstraint ( double          Distance,
                            IAssembled      PartorAssemblyA,
                            IConstrainable  ItemA,
                            IAssembled      PartorAssemblyB,
                            IConstrainable  ItemB,
                            bool            Outside
                          )
```

Adds a tangent constraint between two planes/faces/axes/edges/points

Parameters

Distance	Alignment distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
Outside	true for an outside tangent constraint, false for an inside tangent constraint

◆ AddTangentConstraint() [2/2]

```
void AddTangentConstraint ( double          Distance,
                            IAssembled      PartorAssemblyA,
                            IConstrainable  ItemA,
                            IAssembled      PartorAssemblyB,
                            IConstrainable  ItemB,
                            bool            Outside,
                            bool            IsReversed,
                            string          Name
                          )
```

Adds a tangent constraint between two planes/faces/axes/edges/points

Parameters

Distance	Alignment distance
PartorAssemblyA	First part/assembly to constrain
ItemA	Plane/face/axis/edge/point on first part/assembly to constrain
PartorAssemblyB	Second part/assembly to constrain
ItemB	Plane/face/axis/edge/point on second part/assembly to constrain
Outside	true for an outside tangent constraint, false for an inside tangent constraint
IsReversed	true to reverse constraint
Name	Name of constraint

◆ AnchorPart() [1/2]

```
void AnchorPart ( AssembledPart  Part )
```

Anchors a part

Parameters

Part **Part** to anchor

◆ AnchorPart() [2/2]

void AnchorPart (string Name)

Anchors a part

Parameters

Name Name of part to anchor

◆ AnchorSubAssembly()

void AnchorSubAssembly (string Name)

Anchors a sub-assembly

Parameters

Name Name of sub-assembly to anchor

◆ CreateUniqueName()

string CreateUniqueName (string BaseName)

Creates a unique name that can be used to safely add a part or subassembly to the assembly if the names used in the assembly are not known in advance

Parameters

BaseName Base name to use

Returns

Unique name

◆ DisplayUnits()

UnitTypes DisplayUnits ( )

Gets the display units for the assembly

Returns

The display units

◆ DuplicatePart() [1/4]

```
AssembledPart DuplicatePart ( AssembledPart Part,
                              double        OffsetX,
                              double        OffsetY,
                              double        OffsetZ
                            )
```

Duplicates a part in the assembly

Parameters

Part **Part** to duplicate

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The duplicate part

◆ DuplicatePart() [2/4]

```
AssembledPart DuplicatePart ( AssembledPart Part,
                              double        OffsetX,
                              double        OffsetY,
                              double        OffsetZ,
                              double        AngleX,
                              double        AngleY,
                              double        AngleZ,
                              bool          TranslationFirst
                            )
```

Duplicates a part in the assembly

Parameters

Part	**Part** to duplicate
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The duplicate part

◆ DuplicatePart() [3/4]

```
AssembledPart DuplicatePart ( string  Name,
                              double  OffsetX,
                              double  OffsetY,
                              double  OffsetZ
                            )
```

Duplicates a part in the assembly

Parameters

Name Name of part to duplicate

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The duplicate part

◆ DuplicatePart() [4/4]

```
AssembledPart DuplicatePart ( string  Name,
                              double  OffsetX,
                              double  OffsetY,
                              double  OffsetZ,
                              double  AngleX,
                              double  AngleY,
                              double  AngleZ,
                              bool    TranslationFirst
                            )
```

Duplicates a part in the assembly

Parameters

Name	Name of part to duplicate
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The duplicate part

◆ DuplicateSubAssembly() [1/4]

```
AssembledSubAssembly DuplicateSubAssembly ( AssembledSubAssembly SubAssembly,
                                            double              OffsetX,
                                            double              OffsetY,
                                            double              OffsetZ
                                          )
```

Duplicates a sub-assembly in the assembly

Parameters

SubAssembly	Sub-assembly to duplicate
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset

Returns

The duplicate sub-assembly

◆ DuplicateSubAssembly() [2/4]

```
AssembledSubAssembly DuplicateSubAssembly ( AssembledSubAssembly SubAssembly,
                                            double               OffsetX,
                                            double               OffsetY,
                                            double               OffsetZ,
                                            double               AngleX,
                                            double               AngleY,
                                            double               AngleZ,
                                            bool                 TranslationFirst
                                          )
```

Duplicates a sub-assembly in the assembly

Parameters

SubAssembly	Sub-assembly to duplicate
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The duplicate sub-assembly

◆ DuplicateSubAssembly() [3/4]

```
AssembledSubAssembly DuplicateSubAssembly ( string  Name,
                                            double  OffsetX,
                                            double  OffsetY,
                                            double  OffsetZ
                                          )
```

Duplicates a sub-assembly in the assembly

Parameters

Name Name of sub-assembly to duplicate

OffsetX X offset

OffsetY Y offset

OffsetZ Z offset

Returns

The duplicate sub-assembly

◆ DuplicateSubAssembly() [4/4]

```
AssembledSubAssembly DuplicateSubAssembly ( string  Name,
                                            double  OffsetX,
                                            double  OffsetY,
                                            double  OffsetZ,
                                            double  AngleX,
                                            double  AngleY,
                                            double  AngleZ,
                                            bool    TranslationFirst
                                          )
```

Duplicates a sub-assembly in the assembly

Parameters

Name	Name of sub-assembly to duplicate
OffsetX	X offset
OffsetY	Y offset
OffsetZ	Z offset
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
TranslationFirst	if true translation occurs before rotation, if false rotation occurs before translation

Returns

The duplicate sub-assembly

◆ ExportBIP()

```
void ExportBIP ( string  FileName )
```

Exports a keyshot file

Parameters

FileName	Path and name of keyshot file

◆ ExportIGES()

```
void ExportIGES ( string  FileName )
```

Exports the assembly as a IGES file

Parameters

FileName Path and name of IGES file

◆ ExportSAT()

```
void ExportSAT ( string  FileName,
                 int     Version,
                 bool    SaveColors
               )
```

Exports the assembly as a SAT file

Parameters

FileName Path and name of SAT file

Version Exported SAT file version

SaveColors true to preseve colors

◆ ExportSTEP203()

```
void ExportSTEP203 ( string  FileName )
```

Exports the assembly as a STEP 203 file

Parameters

FileName Path and name of STEP 203 file

◆ ExportSTEP214()

```
void ExportSTEP214 ( string  FileName )
```

Exports the assembly as a STEP 214 file

Parameters

FileName Path and name of STEP 214 file

◆ ExportSTL()

void ExportSTL (string FileName)

Exports the assembly as an STL file

Parameters

FileName Path and name of STL file

◆ GetActiveConfiguration()

Configuration GetActiveConfiguration ( )

Gets the currently active configuration

Returns

Configuration object

◆ GetAxis()

Axis GetAxis (string Name)

Gets an axis from an axis name

Parameters

Name Name of axis to find

Returns

Found axis

◆ GetConfiguration()

Configuration GetConfiguration (string Name)

Gets a configuration with a specific name

Parameters

Name Name of confguration

Returns

Configuration object

◆ GetCustomProperty()

string GetCustomProperty (string Name)

Gets the value of a custonm property

Parameters

Name Name of the custom property

Returns

The value of the property as a string

◆ GetParameter()

Parameter GetParameter (string Name)

Gets a parameter with a specific name

Parameters

Name Name of parameter

Returns

Parameter object

◆ GetPart()

AssembledPart GetPart (string Name)

Gets a part in the assembly

Parameters

Name Name of part instance to get

Returns

The part

◆ GetPartOrientation() [1/2]

List [] GetPartOrientation (**AssembledPart** Part)

Gets the orientation of a part in an assembly

Parameters

Part **Part** in an assembly

Returns

Part orientation as [OffsetX, OffsetY, OffsetZ, AngleX, AngleY, AngleZ], translation before rotation

◆ GetPartOrientation() [2/2]

List [] GetPartOrientation (string PartName)

Gets the orientation of a part in an assembly

Parameters

PartName Name of part to get orientation

Returns

Part orientation as [OffsetX, OffsetY, OffsetZ, AngleX, AngleY, AngleZ], translation before rotation

◆ GetPlane()

Plane GetPlane (string Name)

Gets a plane using the name of the plane

Parameters

Name Name of plane to find

Returns

The plane

◆ GetPoint()

Point GetPoint (string Name)

Gets a point on the assembly using the point name. The point must have been created in a script

Parameters

Name Name of point to get

Returns

The point

◆ GetSubAssembly()

AssembledSubAssembly GetSubAssembly (string Name)

Gets a sub-assembly in the assembly

Parameters

Name Name of sub-assembly instance to get

Returns

The sub-assembly

◆ GetUserData()

IronPython.Runtime.PythonDictionary GetUserData (string Name)

Gets user data

Parameters

Name Name of data to get

Returns

Data as a python dictionary or None if not found

◆ HidePart() [1/2]

void HidePart (**AssembledPart** Part)

Hides a part

Parameters

Part Part to hide

◆ HidePart() [2/2]

void HidePart (string Name)

Hides a part

Parameters

Name Name of part to hide

◆ HideSubAssembly()

void HideSubAssembly (string Name)

Hides a sub-assembly

Parameters

Name Name of sub-assembly to hide

◆ MovePart() [1/2]

void MovePart (**AssembledPart** Part,
double OffsetX,
double OffsetY,
double OffsetZ,
bool ApplyConstraints
)

Moves a part

Parameters

Part **Part** to move

OffsetX X offset to apply

OffsetY Y offset to apply

OffsetZ Z offset to apply

ApplyConstraints true to apply constraints

◆ MovePart() [2/2]

```
void MovePart ( string  Name,
                double  OffsetX,
                double  OffsetY,
                double  OffsetZ,
                bool    ApplyConstraints
              )
```

Moves a part

Parameters

Name	Name of part to move
OffsetX	X offset to apply
OffsetY	Y offset to apply
OffsetZ	Z offset to apply
ApplyConstraints	true to apply constraints

◆ MoveParts()

```
void MoveParts ( List []  Names,
                 double  OffsetX,
                 double  OffsetY,
                 double  OffsetZ,
                 bool    ApplyConstraints
               )
```

Moves a set of parts

Parameters

Names	Names of parts to move
OffsetX	X offset to apply
OffsetY	Y offset to apply
OffsetZ	Z offset to apply
ApplyConstraints	true to apply constraints

◆ MoveSubAssemblies()

```
void MoveSubAssemblies ( List []  Names,
                         double   OffsetX,
                         double   OffsetY,
                         double   OffsetZ,
                         bool     ApplyConstraints
                       )
```

Moves a set of sub-assemblies

Parameters

Names	Names of sub-assemblies to move
OffsetX	X offset to apply
OffsetY	Y offset to apply
OffsetZ	Z offset to apply
ApplyConstraints	true to apply constraints

◆ MoveSubAssembly() [1/2]

```
void MoveSubAssembly ( AssembledSubAssembly  SubAssembly,
                       double                OffsetX,
                       double                OffsetY,
                       double                OffsetZ,
                       bool                  ApplyConstraints
                     )
```

Moves a sub-assembly

Parameters

SubAssembly	Sub-assembly to move
OffsetX	X offset to apply
OffsetY	Y offset to apply
OffsetZ	Z offset to apply
ApplyConstraints	true to apply constraints

◆ MoveSubAssembly() [2/2]

```
void MoveSubAssembly ( string  Name,
                       double  OffsetX,
                       double  OffsetY,
                       double  OffsetZ,
                       bool    ApplyConstraints
                     )
```

Moves a sub-assembly

Parameters

Name	Name of sub-assembly to move
OffsetX	X offset to apply
OffsetY	Y offset to apply
OffsetZ	Z offset to apply
ApplyConstraints	true to apply constraints

◆ RotatePart() [1/2]

```
void RotatePart ( AssembledPart  Part,
                  double         AngleX,
                  double         AngleY,
                  double         AngleZ,
                  bool           ApplyConstraints
                )
```

Rotates a part

Parameters

Part	**Part** to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotatePart() [2/2]

```
void RotatePart ( string  Name,
                  double  AngleX,
                  double  AngleY,
                  double  AngleZ,
                  bool    ApplyConstraints
                )
```

Rotates a part

Parameters

Name	Name of part to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotateParts()

```
void RotateParts ( List []  Names,
                   double  AngleX,
                   double  AngleY,
                   double  AngleZ,
                   bool    ApplyConstraints
                 )
```

Rotates a set of parts

Parameters

Names	Names of parts to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotateSubAssemblies()

```
void RotateSubAssemblies ( List []  Names,
                           double   AngleX,
                           double   AngleY,
                           double   AngleZ,
                           bool     ApplyConstraints
                         )
```

Rotates a set of sub-assemblies

Parameters

Names	Names of sub-assemblies to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotateSubAssembly() [1/3]

```
void RotateSubAssembly ( AssembledSubAssembly  SubAssembly,
                         double                AngleX,
                         double                AngleY,
                         double                AngleZ,
                         bool                  ApplyConstraints
                       )
```

Rotates a sub-assembly

Parameters

SubAssembly	Sub-assembly to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotateSubAssembly() [2/3]

```
void RotateSubAssembly ( IADOccurrence  AssemOcc,
                         double         AngleX,
                         double         AngleY,
                         double         AngleZ,
                         bool           ApplyConstraints
                       )
```

Rotates a sub-assembly

Parameters

AssemOcc	Occurence of sub-assembly to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ RotateSubAssembly() [3/3]

```
void RotateSubAssembly ( string  Name,
                         double  AngleX,
                         double  AngleY,
                         double  AngleZ,
                         bool    ApplyConstraints
                       )
```

Rotates a sub-assembly

Parameters

Name	Name of sub-assembly to rotate
AngleX	X rotation angle in degrees
AngleY	Y rotation angle in degrees
AngleZ	Z rotation angle in degrees
ApplyConstraints	true to apply constraints

◆ Save()

```
void Save ( string  Folder )
```

Saves the assembly to a specific folder

Parameters

Folder Folder to save to

◆ SaveAll()

```
void SaveAll ( string  Folder )
```

Save the assembly and all parts/sub-assemblies to a folder

Parameters

Folder Folder to save to

◆ SaveAs()

```
void SaveAs ( string  Folder,
              string  NewName
            )
```

Saves the assembly to a specific folder with a new name

Parameters

Folder Folder to save to

NewName New name for assembly

◆ SaveSnapshot()

```
void SaveSnapshot ( string  FileName,
                    int     Width,
                    int     Height,
                    bool    UseAspectRatio,
                    bool    UseWidthandHeight
                  )
```

Saves the current view as a bitmap image

Parameters

FileName	Path and mame of file to save to
Width	Width in pixels
Height	Height in pixels
UseAspectRatio	if true uses greater of width/height along with current aspect ratio
UseWidthandHeight	if true uses current width/height of view

◆ SaveThumbnail()

```
void SaveThumbnail ( string  FileName,
                     int     Width,
                     int     Height
                   )
```

Saves a thumbnail image of the assembly

Parameters

FileName	Path and name of file to save to
Width	Width of thumbnail in pixels
Height	Height of thumbnail in pixels

◆ SetCustomProperty()

```
void SetCustomProperty ( string  Name,
                         string  Value
                       )
```

Sets the value of a custom property The custom property must already be defined on the assembly or defined on the user's PC

Parameters

Name Name of the custom property

Value New value for the custom property

◆ SetUserData()

```
void SetUserData ( string                              Name,
                   IronPython.Runtime.PythonDictionary Dict
                 )
```

Sets user data

Parameters

Name Data name of the format companyname.projectname.dataname

Dict Python dictionary of data to store

◆ ShowPart() [1/2]

```
void ShowPart ( AssembledPart  Part )
```

Shows a part

Parameters

Part **Part** to show

◆ ShowPart() [2/2]

```
void ShowPart ( string  Name )
```

Shows a part

Parameters

Name Name of part to show

◆ ShowSubAssembly()

void ShowSubAssembly (string Name)

Shows a sub-assembly

Parameters

Name Name of sub-assembly to show

◆ SuppressPart() [1/2]

void SuppressPart (**AssembledPart** Part)

Suppresses a part

Parameters

Part Part to suppress

◆ SuppressPart() [2/2]

void SuppressPart (string Name)

Suppresses a part

Parameters

Name Name of part to suppress

◆ SuppressSubAssembly()

void SuppressSubAssembly (string Name)

Suppresses a sub-assembly

Parameters

Name Name of sub-assembly to suppress

◆ UnanchorPart() [1/2]

void UnanchorPart (**AssembledPart** Part)

Un-anchors a part

Parameters

Part Part to un-anchor

◆ UnanchorPart() [2/2]

void UnanchorPart (string Name)

Un-anchors a part

Parameters

Name Name of part to un-anchor

◆ UnanchorSubAssembly()

void UnanchorSubAssembly (string Name)

Un-anchors a sub-assembly

Parameters

Name Name of sub-assembly to un-anchor

◆ UnsuppressPart() [1/2]

void UnsuppressPart (**AssembledPart** Part)

Un-suppresses a part

Parameters

Part Part to un-suppress

◆ UnsuppressPart() [2/2]

void UnsuppressPart (string Name)

Un-suppresses a part

Parameters

Name Name of part to un-suppress

◆ UnsuppressSubAssembly()

void UnsuppressSubAssembly (string Name)

Un-suppresses a sub-assembly

Parameters

Name Name of sub-assembly to un-suppress

Axis Class Reference

An axis More...

Inherits IConstrainable, IInstance, ISelectableGeometry, and IAxis.

Public Member Functions

Part **GetPart** ()

Gets the part that the axis is defined on More...

Assembly **GetSelectionAssembly** ()

The assembly that the edge was selected on Only valid when a selection has been made More...

void **Hide** ()

Hides the axis

void **Show** ()

Shows the axis

Properties

string **Name** `[get]`

The name of the axis

Detailed Description

An axis

Member Function Documentation

◆ GetPart()

Part GetPart ( )

Gets the part that the axis is defined on

Returns

Part that defines the axis

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

Bspline Class Reference

Defines a **Bspline** that can be added to 2D sketches More...

Inherits ISketchFigure.

Public Member Functions

	Bspline (int **Order**, List [] **ControlPoints**, List [] **KnotVectors**, List [] **Weights**, bool **IsReference**) Creates a bspline More...
List []	**GetNormalAt** (double u) Gets the normal vector at a point on the spline More...
List []	**GetPointAt** (double u) Gets a point on the spline More...
double	**GetX** (double u) Gets the X value of the spline at a location along the spline More...
double	**GetY** (double u) Gets the Y value of the spline at a location along the spline More...

List [] **Subdivide** (int Segments)

Divides the **Bspline** up into segments More...

Properties

List [] **ControlPoints** `[get, set]`

The control points [x1, y1, ..., xn, yn]

bool **IsReference** `[get, set]`

True if the bspline is a reference bspline, false if it is a regular bspline

List [] **KnotVectors** `[get, set]`

The knot vectors [k1, k2, ..., kn]

double **Length** `[get]`

Gets the length of the **Bspline**

int **Order** `[get, set]`

The order of the bspline

List [] **Weights** `[get, set]`

The weights [w1, w2, ..., wn]

Detailed Description

Defines a **Bspline** that can be added to 2D sketches

Constructor & Destructor Documentation

◆ Bspline()

```
Bspline ( int    Order,
          List [] ControlPoints,
          List [] KnotVectors,
          List [] Weights,
          bool   IsReference
        )
```

Creates a bspline

Parameters

Order	Order of the bspline
ControlPoints	Value of control points [Point1X, Point1Y, ...]
KnotVectors	Knot vectors [KnotVector1, KnotVector2, ...]
Weights	**Point** weights [Weight1, Weight2, ...]
IsReference	True if a reference bspline, false if a regular bspline

Member Function Documentation

◆ GetNormalAt()

```
List [] GetNormalAt ( double u )
```

Gets the normal vector at a point on the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Vector for point on the spline at the specified location (A, B)

◆ GetPointAt()

List [] GetPointAt (double u)

Gets a point on the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Point on the spline at the specified location [X, Y]

◆ GetX()

double GetX (double u)

Gets the X value of the spline at a location along the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

X value of spline at the specified location

◆ GetY()

double GetY (double u)

Gets the Y value of the spline at a location along the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Y value of spline at the specified location

◆ Subdivide()

List [] Subdivide (int Segments)

Divides the **Bspline** up into segments

Parameters

Segments Number of segments to obtain

Returns

List of points between segments [X1, Y1, X2, Y2, ...]

Bspline3D Class Reference

Defines a **Bspline** that can be added to 3D sketches More...

Inherits ISketchFigure3D.

Public Member Functions

Bspline3D (int **Order**, List [] **ControlPoints**, List [] **KnotVectors**, List [] **Weights**, bool **IsReference**)

Creates a bspline More...

List [] **GetNormalAt** (double u)

Gets the normal vector at a point on the spline More...

List [] **GetPointAt** (double u)

Gets a point on the spline More...

double **GetX** (double u)

Gets the X value of the spline at a location along the spline More...

double **GetY** (double u)

Gets the Y value of the spline at a location along the spline More...

double **GetZ** (double u)

Gets the Z value of the spline at a location along the spline More...

List [] **Subdivide** (int Segments)

Divides the **Bspline** up into segments More...

List [] **SubdivideGetNormals** (int Segments)

Divides the **Bspline** up into segments and gets the normal for each point More...

Properties

List [] **ControlPoints** `[get, set]`

The control points [x1, y1, ..., xn, yn]

bool **IsReference** `[get, set]`

True if the bspline is a reference bspline, false if it is a regular bspline

List []	**KnotVectors** `[get, set]` The knot vectors [k1, k2, ..., kn]
double	**Length** `[get]` Gets the length of the **Bspline**
int	**Order** `[get, set]` The order of the bspline
List []	**Weights** `[get, set]` The weights [w1, w2, ..., wn]

Detailed Description

Defines a **Bspline** that can be added to 3D sketches

Constructor & Destructor Documentation

◆ Bspline3D()

```
Bspline3D ( int      Order,
            List []  ControlPoints,
            List []  KnotVectors,
            List []  Weights,
            bool     IsReference
          )
```

Creates a bspline

Parameters

Order	Order of the bspline
ControlPoints	Value of control points [Point1X, Point1Y, ...]
KnotVectors	Knot vectors [KnotVector1, KnotVector2, ...]
Weights	**Point** weights [Weight1, Weight2, ...]
IsReference	True if a reference bspline, false if a regular bspline

Member Function Documentation

◆ GetNormalAt()

List [] GetNormalAt (double u)

Gets the normal vector at a point on the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Vector for point on the spline at the specified location (A, B, C)

◆ GetPointAt()

List [] GetPointAt (double u)

Gets a point on the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Point on the spline at the specified location [X, Y, Z]

◆ GetX()

double GetX (double u)

Gets the X value of the spline at a location along the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

X value of spline at the specified location

◆ GetY()

double GetY (double u)

Gets the Y value of the spline at a location along the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Y value of spline at the specified location

◆ GetZ()

double GetZ (double u)

Gets the Z value of the spline at a location along the spline

Parameters

u Location along the spline. 0.0 = start, 1.0 = end

Returns

Y value of spline at the specified location

◆ Subdivide()

List [] Subdivide (int Segments)

Divides the **Bspline** up into segments

Parameters

Segments Number of segments to obtain

Returns

List of points between segments [X1, Y1, Z1, X2, Y2, Z2, ...]

◆ SubdivideGetNormals()

List [] SubdivideGetNormals (int Segments)

Divides the **Bspline** up into segments and gets the normal for each point

Parameters

Segments Number of segments to obtain

Returns

List of points between segments and normals [X1, Y1, Z1, A1, B1, C1, X2, Y2, Z2, A2, B2, C2, ...]

Circle Class Reference

Describes a 2D circle, which can be added to 2D sketches More...

Inherits ISketchFigure.

Public Member Functions

Circle (List [] **Center**, double **Radius**, bool **IsReference**)

Creates a 2D circle which can be added to sketches More...

Properties

List [] **Center** `[get, set]`

The center of the circle [x, y]

SketchPoint **CenterPoint** `[get]`

The center of the circle as a sketch point

bool **IsReference** `[get, set]`

True if the circle is a reference circle, false if it is a regular circle

double **Length** `[get]`

The length of the circle circumference in script units

double **Radius** `[get, set]`

Radius of the circle

Detailed Description

Describes a 2D circle, which can be added to 2D sketches

Constructor & Destructor Documentation

◆ Circle()

```
Circle ( List []  Center,
         double   Radius,
         bool     IsReference
       )
```

Creates a 2D circle which can be added to sketches

Parameters

Center	Center of the circle as a python list [x, y]
Radius	Radius of circle
IsReference	True to create a reference circle

CircularArc Class Reference

Describes a 2D circular arc, which can be added to 2D sketches More...

Inherits ISketchFigure.

Public Types

enum **ArcType** { **CenterStartEnd** , **CenterStartAngle** }

Types of circular arcs More...

Public Member Functions

CircularArc (List [] **Center**, List [] **Start**, double **Angle**, bool **IsReference**)

Creates an arc using the center, start point and an angle More...

CircularArc (List [] **Center**, List [] **Start**, List [] **End**, bool **IsReference**)

Creates an arc using the center, start point and end point More...

Properties

double **Angle** `[get, set]`

Angle of arc

List [] **Center** `[get, set]`

The center of the arc [x, y]

SketchPoint **CenterPoint** `[get]`

The center point as a sketchpoint object

SketchPoint **End** `[get]`

	The end point as a sketchpoint object
List []	**EndPoint** `[get, set]` The end point of the arc [x, y]
bool	**IsReference** `[get, set]` True if the arc is a reference arc, false if it is a regular arc
double	**Radius** `[get, set]` Radius of arc
SketchPoint	**Start** `[get]` The start point as a sketchpoint object
List []	**StartPoint** `[get, set]` The start point of the arc [x, y]
ArcType	**Type** `[get]` Type of arc

Detailed Description

Describes a 2D circular arc, which can be added to 2D sketches

Member Enumeration Documentation

◆ ArcType

enum **ArcType**

Types of circular arcs

Enumerator	
CenterStartEnd	Arc defined by center, start and end
CenterStartAngle	Arc defines by center, start and angle

Constructor & Destructor Documentation

◆ CircularArc() `[1/2]`

```
CircularArc ( List []  Center,
              List []  Start,
              List []  End,
              bool     IsReference
            )
```

Creates an arc using the center, start point and end point

Parameters

Center	Center of the arc
Start	Start point of the arc
End	End point of the arc
IsReference	True to create a reference arc, false to create a regular arc

◆ CircularArc() [2/2]

```
CircularArc ( List []  Center,
              List []  Start,
              double   Angle,
              bool     IsReference
            )
```

Creates an arc using the center, start point and an angle

Parameters

Center	Location of center of arc
Start	Location of start of arc
Angle	Angle of arc
IsReference	True if a reference arc, false if a regular arc

CircularArc3D Class Reference

Describes a 3D circular arc, which can be added to 3D sketches More...

Inherits ISketchFigure3D.

Public Types

enum **ArcType** { **CenterStartEnd** , **CenterStartAngle** }

Types of circular arcs More...

Public Member Functions

CircularArc3D (List [] **Center**, List [] Start, double **Angle**, bool **IsReference**)

Creates an arc using the center, start point and an angle More...

CircularArc3D (List [] **Center**, List [] Start, List [] End, bool **IsReference**)

Creates an arc using the center, start point and end point More...

Properties

double **Angle** `[get, set]`

Angle of arc

List [] **Center** `[get, set]`

The center of the arc [x, y, z]

List [] **EndPoint** `[get, set]`

The end point of the arc [x, y, z]

bool **IsReference** `[get, set]`

True if the arc is a reference arc, false if it is a regular arc

double **Radius** `[get, set]`

Radius of arc

List [] **StartPoint** `[get, set]`

The start point of the arc [x, y, z]

ArcType **Type** `[get]`

Type of arc

Detailed Description

Describes a 3D circular arc, which can be added to 3D sketches

Member Enumeration Documentation

◆ ArcType

enum **ArcType**

Types of circular arcs

Enumerator	
CenterStartEnd	Arc defined by center, start and end
CenterStartAngle	Arc defines by center, start and angle

Constructor & Destructor Documentation

◆ CircularArc3D() [1/2]

```
CircularArc3D ( List []  Center,
                List []  Start,
                List []  End,
                bool     IsReference
              )
```

Creates an arc using the center, start point and end point

Parameters

Center	Center of the arc
Start	Start point of the arc
End	End point of the arc
IsReference	True to create a reference arc, false to create a regular arc

◆ CircularArc3D() [2/2]

```
CircularArc3D ( List []  Center,
                List []  Start,
                double   Angle,
                bool     IsReference
              )
```

Creates an arc using the center, start point and an angle

Parameters

Center	Location of center of arc
Start	Location of start of arc
Angle	Angle of arc
IsReference	True if a reference arc, false if a regular arc

Configuration Class Reference

Describes a configuration More...

Public Member Functions

void **Activate** ()

Makes the configuration active

void **LockAll** ()

Applies all locks to the configuration

void **SetLocks** (**LockTypes** Locks)

Sets the locks on the configuration More...

void **UnlockAll** ()

Removes all locks from the configuration

Properties

bool **IsActive** `[get]`

True if the configuration is currently active

string **Name** `[get]`

The name of the configuration

Detailed Description

Describes a configuration

Member Function Documentation

◆ SetLocks()

void SetLocks (**LockTypes** Locks)

Sets the locks on the configuration

Parameters

Locks Locks to set

CSharp Class Reference

Provides access to the full Alibre Design **API** by running C# code See the Advanced **API** manual for details More...

Public Member Functions

Script< object[]>	**Compile** (string Code) Compiles C# code More...
IronPython.Runtime.PythonDictionary	**CompileAndRun** (string Code) Compiles and runs C# code More...
IronPython.Runtime.PythonDictionary	**CompileAndRun** (string Code, IronPython.Runtime.PythonDictionary Variables) Compiles and runs C# code More...
IronPython.Runtime.PythonDictionary	**Run** (Script< object[]> Script) Runs compiled C# code More...
IronPython.Runtime.PythonDictionary	**Run** (Script< object[]> Script, IronPython.Runtime.PythonDictionary Variables) Runs compiled C# code More...

Detailed Description

Provides access to the full Alibre Design **API** by running C# code See the Advanced **API** manual for details

Member Function Documentation

◆ Compile()

Script< object[]> Compile (string Code)

Compiles C# code

Parameters

Code Code to compile

Returns

Compiled code object

◆ CompileAndRun() [1/2]

IronPython.Runtime.PythonDictionary CompileAndRun (string Code)

Compiles and runs C# code

Parameters

Code Code to compile and run

Returns

Updated dictionary of variables

◆ CompileAndRun() [2/2]

IronPython.Runtime.PythonDictionary CompileAndRun (string Code,
IronPython.Runtime.PythonDictionary Variables
)

Compiles and runs C# code

Parameters

Code Code to compile and run

Variables Dictionary of variables

Returns

Updated dictionary of variables

◆ Run() [1/2]

IronPython.Runtime.PythonDictionary Run (Script< object[]> Script)

Runs compiled C# code

Parameters

Script Compiled code object to run

Returns

Updated dictionary of variables

◆ Run() [2/2]

IronPython.Runtime.PythonDictionary Run (Script< object[]> Script,
IronPython.Runtime.PythonDictionary Variables
)

Runs compiled C# code

Parameters

Script Compiled code object to run

Variables Dictionary of variables or None for no variables

Returns

Updated dictionary of variables

Edge Class Reference

Describes an edge (can be filleted, chamfered, swept) More...

Inherits IFilletable, IChamferable, ISweepPath, IConstrainable, IInstance, ISelectableGeometry, and IAxis.

Public Member Functions

Part	**GetPart** ()
	Gets the part that the edge is defined on More...
Assembly	**GetSelectionAssembly** ()
	The assembly that the edge was selected on Only valid when a selection has been made More...
List []	**GetVertices** ()
	Gets a python list of the current vertices in the edge More...

Properties

double	**Diameter** `[get]`	The diameter of the edge, if it is a circle
double	**Length** `[get]`	The length of the edge
string	**Name** `[get]`	Name of the edge

Detailed Description

Describes an edge (can be filleted, chamfered, swept)

Member Function Documentation

◆ GetPart()

Part GetPart ( )

Gets the part that the edge is defined on

Returns

Part that contains edge

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ GetVertices()

List [] GetVertices ( )

Gets a python list of the current vertices in the edge

Returns

Python list of vertices

Ellipse Class Reference

Describes an ellipse used in 2D sketches More...

Inherits ISketchFigure.

Public Member Functions

Ellipse (List [] **Center**, double MajorRadius, double **MajorAxisAngle**, double **MinorMajorRatio**, bool **IsReference**)

Creates an ellipse More...

Properties

List [] **Center** `[get, set]`

The center of the ellipse [x, y]

SketchPoint **CenterPoint** `[get]`

The center point as a sketchpoint object

bool **IsReference** `[get, set]`

True if the ellipse is a reference ellipse, false if it is a regular ellipse

double **MajorAxisAngle** `[get, set]`

Angle of major axis

double **MinorMajorRatio** `[get, set]`

Ratio of minor radius to major radius

double **Radius** `[get, set]`

Radius on major axis

Detailed Description

Describes an ellipse used in 2D sketches

Constructor & Destructor Documentation

◆ Ellipse()

```
Ellipse ( List []   Center,
          double    MajorRadius,
          double    MajorAxisAngle,
          double    MinorMajorRatio,
          bool      IsReference
        )
```

Creates an ellipse

Parameters

Center	Center of the ellipse
MajorRadius	Radius on the major axis
MajorAxisAngle	Angle of the major axis in degrees
MinorMajorRatio	Radius on the minor axis as a ratio of the major radius
IsReference	True to create a reference arc, false to create a regular arc

EllipticalArc Class Reference

Describes an elliptical arc used in 2D sketches More...

Inherits ISketchFigure.

Public Member Functions

EllipticalArc (List [] **Center**, List [] **Start**, List [] **End**, double MajorRadius, double **MajorAxisAngle**, double **MinorMajorRatio**, bool **IsReference**)

Creates an elliptical arc More...

Properties

List [] **Center** `[get, set]`

The center of the elliptical arc [x, y]

SketchPoint **CenterPoint** `[get]`

The center point as a sketchpoint object

SketchPoint **End** `[get]`

The end point as a sketchpoint object

List [] **EndPoint** `[get, set]`

The end point of the arc [x, y]

bool **IsReference** `[get, set]`

True if the elliptical arc is a reference elliptical arc, false if it is a regular elliptical arc

double	**MajorAxisAngle** `[get, set]` Angle of major axis
double	**MinorMajorRatio** `[get, set]` Ratio of minor radius to major radius
double	**Radius** `[get, set]` Radius on major axis
SketchPoint	**Start** `[get]` The start point as a sketchpoint object
List []	**StartPoint** `[get, set]` The start point of the arc [x, y]

Detailed Description

Describes an elliptical arc used in 2D sketches

Constructor & Destructor Documentation

◆ EllipticalArc()

```
EllipticalArc ( List []  Center,
                List []  Start,
                List []  End,
                double   MajorRadius,
                double   MajorAxisAngle,
                double   MinorMajorRatio,
                bool     IsReference
              )
```

Creates an elliptical arc

Parameters

Center	Center of the elliptical arc
Start	The start point for the arc
End	The end point for the arc
MajorRadius	Radius on the major axis
MajorAxisAngle	Angle of the major axis in degrees
MinorMajorRatio	Radius on the minor axis as a ratio of the major radius
IsReference	True to create a reference arc, false to create a regular arc

Face Class Reference

Describes a face (can be filleted, chamfered, used for sketches, used for loft cross sections) More...

Inherits ISketchSurface, IFilletable, IChamferable, ICrossSection, IConstrainable, IInstance, ISelectableGeometry, and IPlane.

Public Member Functions

double	**DistanceTo** (**Face** OtherFace) Gets the distance from this face to another face More...
List []	**GetAdjoiningFaces** () Gets a list of the adjoining faces More...
double	**GetArea** () Gets the area of the face More...
List []	**GetEdges** () Gets a list of the current edges in the face More...
Part	**GetPart** () Gets the part that the face is defined on More...

Assembly	**GetSelectionAssembly** () The assembly that the edge was selected on Only valid when a selection has been made More...
List []	**GetVertices** () Gets a list of the current vertices in the face More...
bool	**IsParallel** (**Face** OtherFace) Checks if another face is parallel to this one More...
bool	**IsRectangle** () Determines if the face is a rectangle More...

Properties

string	**Name** `[get]` The name of the face

Detailed Description

Describes a face (can be filleted, chamfered, used for sketches, used for loft cross sections)

Member Function Documentation

◆ DistanceTo()

double DistanceTo (**Face** OtherFace)

Gets the distance from this face to another face

Parameters

OtherFace The other face to measure to

Returns

The distance between faces

◆ GetAdjoiningFaces()

List [] GetAdjoiningFaces ( )

Gets a list of the adjoining faces

Returns

List of faces

◆ GetArea()

double GetArea ( )

Gets the area of the face

From: https://stackoverflow.com/questions/20672183/calculating-the-area-of-a-closed-polygon-on-a-plane

Returns

Area of face

◆ GetEdges()

List [] GetEdges ( )

Gets a list of the current edges in the face

Returns

List of edges

◆ GetPart()

Part GetPart ( )

Gets the part that the face is defined on

Returns

Part that contains face

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ GetVertices()

List [] GetVertices ( )

Gets a list of the current vertices in the face

Returns

List of vertices

◆ IsParallel()

bool IsParallel (**Face** OtherFace)

Checks if another face is parallel to this one

Parameters

OtherFace The other face to check

Returns

true if the faces are parallel

◆ IsRectangle()

bool IsRectangle ( )

Determines if the face is a rectangle

Returns

true if face is a rectangle

Feature Class Reference

Describes a feature of an object, e.g. boss, cut More...

Public Member Functions

void **SetColor** (byte Red, byte Green, byte Blue)

Sets the color of the part More...

Properties

string **Name** `[get]`

Name of the feature

Detailed Description

Describes a feature of an object, e.g. boss, cut

Member Function Documentation

◆ SetColor()

```
void SetColor ( byte Red,
                byte Green,
                byte Blue
              )
```

Sets the color of the part

Parameters

Red	Red component 0 - 255
Green	Green component 0 - 255
Blue	Blue component 0 - 255

GearSketch Class Reference

A 2D sketch containing an involute gear profile. Can be treated as a regular sketch More...

Inherits **Sketch**.

Public Attributes

double	**CenterX** X coordinate of gear center
double	**CenterY** Y coordinate of gear center
double	**DiametralPitch** Diametral pitch of gear in teeth per inch
int	**NumberofTeeth** Number of teeth in gear
double	**PitchDiameter** Pitch diameter of gear in script units
double	**PressureAngle**

Pressure angle of gear

Additional Inherited Members

Public Types inherited from Sketch

Supported sketch constraints More...

Public Member Functions inherited from Sketch

Properties inherited from Sketch

Detailed Description

A 2D sketch containing an involute gear profile. Can be treated as a regular sketch

GlobalParameters Class Reference

A set of global parameters More...

Public Member Functions

GlobalParameters (string Folder, string **Name**)

Opens an existing global parameters set More...

GlobalParameters (string **Name**)

Creates a new global parameters set More...

GlobalParameters (string **Name**, bool CreateNew)

Creates a new global parameters set or accesses an already opened global parameters set More...

Configuration **AddConfiguration** (string **Name**)

Adds a configuration to the global parameters set More...

Configuration **AddConfiguration** (string **Name**, string BaseConfigurationName)

Adds a configuration to the global parameters set using another configuration as a base More...

Parameter **AddParameter** (string **Name**, **ParameterTypes** Type, double Value)

Adds a parameter to the global parameters set More...

Parameter **AddParameter** (string **Name**, **ParameterTypes** Type, string Equation)

Adds a parameter to the global parameters set More...

void **Close** ()

Closes the global parameters set If it is unsaved then changes will be lost

Configuration **GetActiveConfiguration** ()

Gets the currently active configuration More...

Configuration **GetConfiguration** (string **Name**)

Gets a configuration with a specific name More...

Parameter	**GetParameter** (string **Name**) Gets a parameter with a specific name More...
void	**Save** () Saves the global parameters set using the current path and file name
void	**Save** (string Folder) Saves the global parameters set to a specific folder More...
void	**SaveAs** (string Folder, string NewName) Saves the global parameters set to a specific folder with a new name More...

Properties

List []	**Configurations** `[get]` A list of configurations
string	**Name** `[get]` Name of the global parameters
List []	**Parameters** `[get]` A list of parameters

Detailed Description

A set of global parameters

Constructor & Destructor Documentation

◆ GlobalParameters() [1/3]

GlobalParameters (string Folder,
string Name
)

Opens an existing global parameters set

Parameters

Folder Folder containing global parameters

Name Name of global parameters to open

◆ GlobalParameters() [2/3]

GlobalParameters (string Name)

Creates a new global parameters set

Parameters

Name Name of new global parameters set

◆ GlobalParameters() [3/3]

GlobalParameters (string Name,
bool CreateNew
)

Creates a new global parameters set or accesses an already opened global parameters set

Parameters

Name Name of global parameters set to create or access

CreateNew True to create a new global parameters set, false to access an opened global parameters

Member Function Documentation

◆ AddConfiguration() [1/2]

Configuration AddConfiguration (string Name)

Adds a configuration to the global parameters set

Parameters

Name Name of configuration

Returns

New configuration

◆ AddConfiguration() [2/2]

```
Configuration AddConfiguration ( string  Name,
                                 string  BaseConfigurationName
                               )
```

Adds a configuration to the global parameters set using another configuration as a base

Parameters

Name	Name of configuration
BaseConfigurationName	Name of base configuration to use

Returns

New configuration

◆ AddParameter() [1/2]

```
Parameter AddParameter ( string          Name,
                         ParameterTypes  Type,
                         double          Value
                       )
```

Adds a parameter to the global parameters set

Parameters

Name	Name of parameter
Type	Type of parameter
Value	Value for parameter

Returns

New parameter

◆ AddParameter() [2/2]

```
Parameter AddParameter ( string          Name,
                         ParameterTypes  Type,
                         string          Equation
                       )
```

Adds a parameter to the global parameters set

Parameters

Name Name of parameter

Type Type of parameter

Equation Equation for parameter

Returns

New parameter

◆ GetActiveConfiguration()

```
Configuration GetActiveConfiguration ( )
```

Gets the currently active configuration

Returns

Configuration object

◆ GetConfiguration()

```
Configuration GetConfiguration ( string  Name )
```

Gets a configuration with a specific name

Parameters

Name Name of confguration

Returns

Configuration object

◆ GetParameter()

Parameter GetParameter (string Name)

Gets a parameter with a specific name

Parameters

Name Name of parameter

Returns

Parameter object

◆ Save()

void Save (string Folder)

Saves the global parameters set to a specific folder

Parameters

Folder Folder to save to

◆ SaveAs()

void SaveAs (string Folder,
string NewName
)

Saves the global parameters set to a specific folder with a new name

Parameters

Folder Folder to save to

NewName New name for global parameters set

Line Class Reference

Describes a 2D line, which can be added to 2D sketches More...

Inherits ISketchFigure.

Public Member Functions

Line (List [] **StartPoint**, List [] **EndPoint**, bool **IsReference**)

Creates a new 2D line More...

Properties

SketchPoint	**End** `[get]` The end point as a sketchpoint object
List []	**EndPoint** `[get, set]` The end point of the line [x, y]
bool	**IsReference** `[get, set]` True if the line is a reference line, false if it is a regular line
double	**Length** `[get]` The length of the line in script units
SketchPoint	**Start** `[get]` The start point as a sketchpoint object
List []	**StartPoint** `[get, set]` The start point of the line [x, y]

Detailed Description

Describes a 2D line, which can be added to 2D sketches

Constructor & Destructor Documentation

◆ Line()

```
Line ( List []  StartPoint,
       List []  EndPoint,
       bool     IsReference
     )
```

Creates a new 2D line

Parameters

StartPoint	Location of the start point [x, y]
EndPoint	Location of the end point [x, y]
IsReference	True if a reference line

Line3D Class Reference

Describes a 3D line, which can be added to 3D sketches More...

Inherits ISketchFigure3D.

Public Member Functions

Line3D (List [] **StartPoint**, List [] **EndPoint**, bool **IsReference**)

Creates a new 3D line More...

Properties

SketchPoint3D	**End** `[get]` The end point as a sketchpoint object
List []	**EndPoint** `[get, set]` The end point of the line [x, y, z]
bool	**IsReference** `[get, set]` True if the line is a reference line, false if it is a regular line
double	**Length** `[get]` The length of the line in script units
SketchPoint3D	**Start** `[get]` The start point as a sketchpoint object
List []	**StartPoint** `[get, set]` The start point of the line [x, y, z]

Detailed Description

Describes a 3D line, which can be added to 3D sketches

Constructor & Destructor Documentation

◆ Line3D()

```
Line3D ( List []  StartPoint,
         List []  EndPoint,
         bool     IsReference
       )
```

Creates a new 3D line

Parameters

StartPoint	Location of the start point [x, y, z]
EndPoint	Location of the end point [x, y, z]
IsReference	True if a reference line

Material Class Reference

Material densities in kg/cm3 More...

Static Public Attributes

static double **ABS**

Density for ABS plastic in kg/cm3

static double **PLA**

Density for PLA plastic in kg/cm3

Detailed Description

Material densities in kg/cm3

Parameter Class Reference

Describes a parameter More...

Public Member Functions

void **AttachToExcel** (string Document, string Sheet, string Cell, **UnitTypes Units**)

Attaches the parameter to a cell in an Ezcel spreadsheet More...

Properties

string **Comment** `[get, set]`

	Comment for the parameter
string	**Equation** `[get, set]` Equation of the parameter
string	**ExcelCell** `[get]` Excel cell associated with the parameter, e.g. 'B3'
string	**ExcelSheet** `[get]` Excel sheet associated with the parameter, e.g. 'Sheet1'
string	**ExcelWorkbook** `[get]` Excel workbook associated with the parameter e.g. 'Foo.xlsx'
string	**Name** `[get, set]` Name of the parameter
double	**RawValue** `[get, set]` Raw value of the parameter
ParameterTypes	**Type** `[get]` Type of the parameter
ParameterUnits	**Units** `[get, set]` Current units of the parameter
double	**Value** `[get, set]` Current value of the parameter in script units (for mm, cm, in), or degrees for angles, or raw value for other units

Detailed Description

Describes a parameter

Member Function Documentation

◆ AttachToExcel()

```
void AttachToExcel ( string      Document,
                     string      Sheet,
                     string      Cell,
                     UnitTypes   Units
                   )
```

Attaches the parameter to a cell in an Ezcel spreadsheet

Parameters

Document	Path and name of Excel spreadsheet
Sheet	Name of sheet to use
Cell	Cell to use
Units	Units used in the cell

Part Class Reference

Object that represents a part More...

Inherited by **AssembledPart**.

Public Types

enum **DirectionType** { **Axis** , **Edge** , **Normal** }

Extrusion directions - extrude along... More...

enum **EndCondition** {
ToDepth , **MidPlane** , **ToNext** , **ToGeometry** ,
EntirePath , **ThroughAll**
}

Extrusion end conditions - extrude until... More...

enum **FileTypes** {
AlibreDesignPart , **STEP** , **IGES** , **ThreeDM** ,
SAT , **STL_in** , **STL_cm** , **STL_mm** ,
GeomagicDesignPart
}

Supported file types More...

Public Member Functions

Part (string **FileName**, **FileTypes** Type)

Opens or imports an existing file for editing More...

	Part (string **FileName**, **FileTypes** Type, bool HideEditor) Opens or imports an existing file for editing, optionally hiding the editor More...
	Part (string Folder, string **Name**) Opens an existing part More...
	Part (string Folder, string **Name**, bool HideEditor) Opens an existing part, optionally hiding the editor More...
	Part (string **Name**) Creates a new part More...
	Part (string **Name**, bool CreateNew) Creates a new part or accesses an already opened part More...
	Part (string **Name**, bool CreateNew, bool HideEditor) Creates a new part or accesses an already opened part, optionally hiding the editor More...
Sketch3D	**Add3DSketch** (string **Name**) Creates a new 3D sketch More...
Axis	**AddAxis** (string **Name**, **Face** CylindricalFace) Creates an axis for a cylindrical face More...
Axis	**AddAxis** (string **Name**, List [] Point1, List [] Point2) Creates an axis based on two points More...
Axis	**AddAxis** (string **Name**, ISketchSurface Plane1, ISketchSurface Plane2) Creates an axis based on the intersection of two planes/faces More...
Axis	**AddAxis** (string **Name**, **Point** PointA, **Point** PointB) Creates an axis based on two points More...
Feature	**AddChamfer** (string **Name**, IChamferable Item, double Distance, bool TangentPropagate) Adds a chamfer to a face or edge More...
Feature	**AddChamfer** (string **Name**, IChamferable Item, double Distance1, double Distance2, bool TangentPropagate) Adds a chamfer to a face or edge More...
Feature	**AddChamfer** (string **Name**, List [] Items, double Distance, bool TangentPropagate) Adds a chamfer to a set of faces and edges More...
Feature	**AddChamfer** (string **Name**, List [] Items, double Distance1, double Distance2, bool TangentPropagate) Adds a chamfer to a set of faces and edges More...
Feature	**AddChamferAngle** (string **Name**, IChamferable Item, double Distance, double Angle, bool TangentPropagate) Adds a chamfer to a face or edge More...
Feature	**AddChamferAngle** (string **Name**, List [] Items, double Distance, double Angle, bool TangentPropagate) Adds a chamfer to a set of faces and edges More...
Configuration	**AddConfiguration** (string **Name**)

	Adds a configuration to the part More...
Configuration	**AddConfiguration** (string **Name**, string BaseConfigurationName) Adds a configuration to the part using another configuration as a base More...
Feature	**AddExtrudeBoss** (string **Name**, **Sketch Sketch**, double Depth, bool IsReversed) Adds a simple extrude boss to a specific depth More...
Feature	**AddExtrudeBoss** (string **Name**, **Sketch Sketch**, double Depth, bool IsReversed, **EndCondition EndCondition**, ISketchSurface EndPlane, double EndOffset, **DirectionType** Direction, ISweepPath SweepPath, double DraftAngle, bool OutwardDraft) Adds an extrude feature More...
Feature	**AddExtrudeCut** (string **Name**, **Sketch Sketch**, double Depth, bool IsReversed) Adds a simple extrude cut to a specific depth More...
Feature	**AddExtrudeCut** (string **Name**, **Sketch Sketch**, double Depth, bool IsReversed, **EndCondition EndCondition**, ISketchSurface EndPlane, double EndOffset, **DirectionType** Direction, ISweepPath SweepPath, double DraftAngle, bool OutwardDraft) Adds an extrude cut feature More...
Feature	**AddFillet** (string **Name**, IFilletable Item, double Radius, bool TangentPropagate) Adds a constant radius fillet to a face or edge More...
Feature	**AddFillet** (string **Name**, List [] Items, double Radius, bool TangentPropagate) Adds a constant radius fillet to a set of faces and edges More...
Feature	**AddFillet** (string **Name**, List [] Items, List [] StartRadii, List [] EndRadii, bool TangentPropagate) Adds a variable radius fillet to a set of faces and edges More...
GearSketch	**AddGear** (string **Name**, double DiametralPitch, int NumberofTeeth, double PitchDiameter, double PressureAngle, bool SingleTooth, double CenterX, double CenterY, int InvolutePoints, ISketchSurface **Plane**) Adds a gear sketch to the part More...
GearSketch	**AddGearDN** (string **Name**, double DiametralPitch, int NumberofTeeth, double PressureAngle, double CenterX, double CenterY, bool SingleTooth, ISketchSurface **Plane**) Adds a gear sketch to the part using diametral pitch and number of teeth More...
GearSketch	**AddGearDN** (string **Name**, double DiametralPitch, int NumberofTeeth, double PressureAngle, double CenterX, double CenterY, ISketchSurface **Plane**) Adds a gear sketch to the part using diametral pitch and number of teeth More...
GearSketch	**AddGearDP** (string **Name**, double DiametralPitch, double PitchDiameter, double PressureAngle, double CenterX, double CenterY, bool SingleTooth, ISketchSurface **Plane**) Adds a gear sketch to the part using diametral pitch and pitch diameter More...
GearSketch	**AddGearDP** (string **Name**, double DiametralPitch, double PitchDiameter, double PressureAngle, double CenterX, double CenterY, ISketchSurface **Plane**) Adds a gear sketch to the part using diametral pitch and pitch diameter More...
GearSketch	**AddGearNP** (string **Name**, int NumberofTeeth, double PitchDiameter, double PressureAngle, double CenterX, double CenterY, bool SingleTooth, ISketchSurface **Plane**) Adds a gear sketch to the part using number of teeth and pitch diameter More...

GearSketch	**AddGearNP** (string **Name**, int NumberofTeeth, double PitchDiameter, double PressureAngle, double CenterX, double CenterY, ISketchSurface **Plane**) Adds a gear sketch to the part using number of teeth and pitch diameter More...
Feature	**AddLoftBoss** (string **Name**, List [] CrossSections, bool MinimizeTwist, bool MinimizeCurvature, bool SimplifySurface, bool ConnectEnds) Adds a loft extrusion More...
Feature	**AddLoftBoss** (string **Name**, List [] CrossSections, List [] GuideCurves, **GuideCurveTypes** GuideType, bool MinimizeTwist, bool MinimizeCurvature, bool SimplifySurface, bool ConnectEnds) Adds a loft extrusion using guide curves More...
Feature	**AddLoftCut** (string **Name**, List [] CrossSections, bool MinimizeTwist, bool MinimizeCurvature, bool SimplifySurface, bool ConnectEnds) Adds a loft cut More...
Feature	**AddLoftCut** (string **Name**, List [] CrossSections, List [] GuideCurves, **GuideCurveTypes** GuideType, bool MinimizeTwist, bool MinimizeCurvature, bool SimplifySurface, bool ConnectEnds) Adds a loft cut using guide curves More...
Parameter	**AddParameter** (string **Name**, **ParameterTypes** Type, double Value) Adds a cm/mm/in/deg parameter to the part More...
Parameter	**AddParameter** (string **Name**, **ParameterTypes** Type, **ParameterUnits** UnitstoUse, double Value) Adds a parameter to the part with specific units More...
Parameter	**AddParameter** (string **Name**, **ParameterTypes** Type, string Equation) Adds a parameter to the part More...
Plane	**AddPlane** (string **Name**, **Axis Axis**, **Point Point**) Creates a new plane contaning an axis and a point More...
Plane	**AddPlane** (string **Name**, List [] NormalVector, List [] PointonPlane) Adds a plane using a normal vector and a point on the plane More...
Plane	**AddPlane** (string **Name**, List [] Point1, List [] Point2, List [] Point3) Creates a plane using three points. Each point is defined as list of [x, y, z] More...
Plane	**AddPlane** (string **Name**, ISketchSurface SourcePlane, **Axis** RotationAxis, double Angle) Creates a new plane at an angle to an existing plane More...
Plane	**AddPlane** (string **Name**, ISketchSurface SourcePlane, double Offset) Creates a plane based on the offset from an existing plane More...
Point	**AddPoint** (string **Name**, double X, double Y, double Z) Adds a point to the part More...
Point	**AddPoint** (string **Name**, **Edge** TargetEdge, double Ratio) Add a point on an edge More...
Point	**AddPoint** (string **Name**, IAxis AxisOrEdge, IPlane PlaneOrFace) Add a point at the the intersection of a axis or edge and a plane or face More...
Point	**AddPoint** (string **Name**, IAxis AxisOrEdge1, IAxis AxisOrEdge2) Add a point at the intersection or two axes or edges More...

Point	**AddPoint** (string **Name**, IPlane PlaneOrFace1, IPlane PlaneOrFace2, IPlane PlaneOrFace3) Add a point at the intersection of three planes or faces More...
Point	**AddPoint** (string **Name**, IPoint PointOrVertex, double XOffset, double YOffset, double ZOffset) Add a point at an offset to a point or a vertex More...
Point	**AddPoint** (string **Name**, IPoint PointOrVertex1, IPoint PointOrVertex2, double Ratio) Add a point between two points/vertices More...
Point	**AddPoint** (string **Name**, IPoint SourcePointOrVertex, IPlane TargetPlaneOrFace, double XOffset, double YOffset) Add a point by projecting a point or vertex onto a plane or face More...
Point	**AddPoint** (string **Name**, List [] **Point**) Adds a point to the part More...
void	**AddPoint** (string **Name**, **Point Point**) Adds a point to the part More...
Point	**AddPointFromCircularEdge** (string **Name**, **Edge** TargetEdge) Adds a point at the center of a circular edge More...
Point	**AddPointFromToroidalFace** (string **Name**, **Face** TargetFace) Adds a point at the center of a toroidal face More...
void	**AddPoints** (string Prefix, List [] Points) Adds a set of points to the part More...
Feature	**AddRevolveBoss** (string **Name**, **Sketch Sketch**, **Axis Axis**, double Angle) Creates a revolve boss feature More...
Feature	**AddRevolveCut** (string **Name**, **Sketch Sketch**, **Axis Axis**, double Angle) Creates a revolve cut feature More...
Sketch	**AddSketch** (string **Name**, ISketchSurface **Plane**) Creates a new sketch using a plane/face More...
Feature	**AddSweepBoss** (string **Name**, **Sketch** ProfileSketch, ISweepPath PathSketch, bool IsRigid, **EndCondition EndCondition**, ISketchSurface EndPlane, double EndOffset, double DraftAngle, bool OutwardDraft) Adds a sweep extrude feature More...
Feature	**AddSweepCut** (string **Name**, **Sketch** ProfileSketch, ISweepPath PathSketch, bool IsRigid, **EndCondition EndCondition**, ISketchSurface EndPlane, double EndOffset, double DraftAngle, bool OutwardDraft) Adds a sweep extrude cut feature More...
Feature	**AddVertexChamfer** (string **Name**, List [] Items, double Distance1, double Distance2, double Distance3) Adds a chamfer to a set of vertices More...
Feature	**AddVertexChamfer** (string **Name**, **Vertex** Item, double Distance1, double Distance2, double Distance3) Adds a chamfer to a vertex More...

void	**Close** () Closes the part If it is unsaved then changes will be lost
UnitTypes	**DisplayUnits** () Gets the display units for the part More...
void	**ExportBIP** (string **FileName**) Exports a keyshot file More...
void	**ExportIGES** (string **FileName**) Exports the part as a IGES file More...
void	**ExportRotatedSTL** (string **FileName**, **Face** BottomFace, bool ForcetoMillimeters, bool UseCustomSettings, double MaxCellSize, double NormalDeviation, double SurfaceDeviation) Exports the part as an STL rotated so that a specific face is on the bottom More...
void	**ExportSAT** (string **FileName**, int Version, bool SaveColors) Exports the part as a SAT file More...
void	**ExportSTEP203** (string **FileName**) Exports the part as a STEP 203 file More...
void	**ExportSTEP214** (string **FileName**) Exports the part as a STEP 214 file More...
void	**ExportSTL** (string **FileName**) Exports the part as an STL file More...
Sketch3D	**Get3DSketch** (string **Name**) Gets a sketch using the name of the sketch More...
Configuration	**GetActiveConfiguration** () Gets the currently active configuration More...
Axis	**GetAxis** (string **Name**) Gets an axis from an axis name More...
List []	**GetBoundingBox** () Gets the bounding box for the part as eight points More...
Configuration	**GetConfiguration** (string **Name**) Gets a configuration with a specific name More...
string	**GetCustomProperty** (string **Name**) Gets the value of a custonm property More...
Edge	**GetEdge** (string **Name**) Gets an edge using it's name "Edge<n>" More...
List []	**GetEdges** () Gets a python list of the current edges in the part More...
Face	**GetFace** (string **Name**) Gets a face using it's name "Face<n>" More...
List []	**GetFaces** ()

	Gets a python list of the current faces in the part More...
Feature	**GetFeature** (string **Name**) Gets a feature on the part More...
Parameter	**GetParameter** (string **Name**) Gets a parameter with a specific name More...
Plane	**GetPlane** (string **Name**) Gets a plane using the name of the plane More...
Point	**GetPoint** (string **Name**) Gets a point on the part using the point name. The point must have been created in a script More...
Assembly	**GetSelectionAssembly** () The assembly that the part was selected on Only valid when a selection has been made More...
Sketch	**GetSketch** (string **Name**) Gets a sketch using the name of the sketch More...
IronPython.Runtime.PythonDictionary	**GetUserData** (string **Name**) Gets user data More...
Vertex	**GetVertex** (string **Name**) Gets a vertex using it's name "Vertex<n>" More...
List []	**GetVertices** () Gets a python list of the current vertices in the part More...
void	**HideFeature** (**Feature Feature**) Hides a feature on the part More...
void	**HideFeature** (string **Name**) Hides a feature on the part More...
bool	**IsOpen** () Checks if the part is opened More...
Feature	**NonUniformScale** (string **Name**, bool ScaleAboutCenter, double ScaleFactorX, double ScaleFactorY, double ScaleFactorZ) Non-uniform scaling of the part More...
void	**PauseUpdating** () Pauses updating the part user interface
void	**Regenerate** () Regenerates the part
void	**RemoveFeature** (**Feature Feature**) Removes a feature from the part More...
void	**RemoveFeature** (string **Name**) Removes a feature from the part More...
void	**RemovePlane** (**Plane Plane**) Removes a plane from the part More...

void	**RemovePoint** (**Point Point**) Removes a point from the part More...
void	**RemoveSketch** (**Sketch Sketch**) Removes a sketch from the part More...
void	**RemoveSketch** (string **Name**) Removes a sketch from the part More...
void	**ResumeUpdating** () Resumes updating the part user interface
void	**Save** () Saves the part using the current path and file name
void	**Save** (string Folder) Saves the part to a specific folder More...
void	**SaveAs** (string Folder, string NewName) Saves the part to a specific folder with a new name More...
void	**SaveSnapshot** (string **FileName**, int Width, int Height, bool UseAspectRatio, bool UseWidthandHeight) Saves the current view as a bitmap image More...
void	**SaveThumbnail** (string **FileName**, int Width, int Height) Saves a thumbnail image of the part More...
Feature	**Scale** (string **Name**, bool ScaleAboutCenter, double ScaleFactor) Uniform scaling of the part More...
void	**Select** (List [] FacesEdgesList) Selects a group of faces, edges, vertices, points, axes, planes and sketches More...
void	**Select** (ISelectableGeometry FaceorEdge) Selects a face, edge, vertex, point, axis, plane, sketch More...
void	**SetColor** (byte Red, byte Green, byte Blue) Sets the color of the part More...
void	**SetCustomProperty** (string **Name**, string Value) Sets the value of a custom property The custom property must already be defined on the part or defined on the user's PC More...
void	**SetUserData** (string **Name**, IronPython.Runtime.PythonDictionary Dict) Sets user data More...
void	**ShowFeature** (**Feature Feature**) Shows a feature on the part More...
void	**ShowFeature** (string **Name**) Shows a feature on the part More...
void	**SuppressFeature** (**Feature Feature**) Suppresses a feature on the part More...

void **SuppressFeature** (string **Name**)

Suppresses a feature on the part More...

void **UnsuppressFeature** (**Feature Feature**)

Unsuppresses a feature on the part More...

void **UnsuppressFeature** (string **Name**)

Unsuppresses a feature on the part More...

Properties

string **Comment** `[get, set]`

Comment property

List [] **Configurations** `[get]`

List of configurations defined on the part

string **CostCenter** `[get, set]`

Cost center property

string **CreatedBy** `[get, set]`

Created By property

string **CreatedDate** `[get, set]`

Created Date property

string **CreatingApplication** `[get, set]`

Creating Application property

double **Density** `[get, set]`

Density of the part

string **Description** `[get, set]`

Description of the part

string **DocumentNumber** `[get, set]`

Document Number property

string **EngineeringApprovalDate** `[get, set]`

Engineering Approval Date property

string **EngineeringApprovedBy** `[get, set]`

Engineering Approved By property

string **EstimatedCost** `[get, set]`

Estimated Cost property

string **ExtendedMaterialInformation** `[get, set]`

Material (extended information) property

string **FileName** `[get]`

Path and filename of the part

string **Keywords** `[get, set]`

Keywords property

string **LastAuthor** `[get, set]`

Last Author property

string **LastUpdateDate** `[get, set]`

Last Update Date property

string **ManufacturingApprovedBy** `[get, set]`

Manufacturing Approved By property

string **ManufacturingApprovedDate** `[get, set]`

Product property

double **Mass** `[get]`

Mass of the part

string **Material** `[get, set]`

Material of the part

string **ModifiedInformation** `[get, set]`

Modified Information property

string **Name** `[get]`

Name of the part

string **Number** `[get, set]`

User-defined number for the part

Point **Origin** `[get]`

Gets the origin (language independent)

List [] **Parameters** `[get]`

List of parameters defined on the part

string **Product** `[get, set]`

Product property

string **ReceivedFrom** `[get, set]`

Received From property

string **Revision** `[get, set]`

Revision property

List [] **Selections** `[get]`

Gets the currently selected items as [ItemA, ItemB, ...] Supports faces, edges, vertices, planes, axes and points

string **StockSize** `[get, set]`

Stock Size property

string **Supplier** `[get, set]`

Supplier property

string **Title** `[get, set]`

Title property

string **Vendor** `[get, set]`

Vendor property

string **WebLink** `[get, set]`

Web Link property

Axis **XAxis** `[get]`

Gets the X-axis (language independent)

Plane **XYPlane** `[get]`

Gets the XY-plane (language independent)

Axis **YAxis** `[get]`

Gets the Y-axis (language independent)

Plane **YZPlane** `[get]`

Gets the YZ-plane (language independent)

Axis **ZAxis** `[get]`

Gets the Z-axis (language independent)

Plane **ZXPlane** `[get]`

Gets the ZX-plane (language independent)

Detailed Description

Object that represents a part

Member Enumeration Documentation

◆ DirectionType

enum **DirectionType**

Extrusion directions - extrude along...

Enumerator	
Axis	Extrude along axis
Edge	Extrude along edge
Normal	Extrude along normal

◆ EndCondition

enum **EndCondition**

Extrusion end conditions - extrude until...

Enumerator	
ToDepth	Extrude to depth
MidPlane	Midplane extrusion
ToNext	Extrude to next
ToGeometry	Extrude to geometry
EntirePath	Extrude entire path
ThroughAll	Extrude through all

◆ FileTypes

enum **FileTypes**

Supported file types

Enumerator	
AlibreDesignPart	Alibre Design **Part**
STEP	STEP
IGES	IGES
ThreeDM	3DM
SAT	SAT
STL_in	STL in inches
STL_cm	STL in centimeters
STL_mm	STL in millimeters
GeomagicDesignPart	Deprecated - do not use

Constructor & Destructor Documentation

◆ Part() [1/7]

```
Part ( string  Folder,
       string  Name
     )
```

Opens an existing part

Parameters

Folder Folder containing part

Name Name of part to open

◆ Part() [2/7]

```
Part ( string  Folder,
       string  Name,
       bool    HideEditor
     )
```

Opens an existing part, optionally hiding the editor

Parameters

Folder Folder containing part

Name Name of part to open

HideEditor True to hide the editor (only valid if part is not already open)

◆ Part() [3/7]

```
Part ( string  Name )
```

Creates a new part

Parameters

Name Name of new part

◆ Part() [4/7]

```
Part ( string  Name,
       bool    CreateNew
     )
```

Creates a new part or accesses an already opened part

Parameters

Name	Name of part to create or access
CreateNew	True to create a new part, false to access an opened part

◆ Part() [5/7]

```
Part ( string  Name,
       bool    CreateNew,
       bool    HideEditor
     )
```

Creates a new part or accesses an already opened part, optionally hiding the editor

Parameters

Name	Name of part to create or access
CreateNew	True to create a new part, false to access an opened part
HideEditor	True to hide the editor (only valid if CreateNew is true)

◆ Part() [6/7]

```
Part ( string      FileName,
       FileTypes   Type
     )
```

Opens or imports an existing file for editing

Parameters

FileName	Name of file to open
Type	Type of file (GeomagicDesignPart, STEP, IGES, ThreeDM, SAT, STL_in, STL_cm, STL_mm)

◆ Part() [7/7]

```
Part ( string      FileName,
       FileTypes   Type,
       bool        HideEditor
     )
```

Opens or imports an existing file for editing, optionally hiding the editor

Parameters

FileName	Name of file to open
Type	Type of file (GeomagicDesignPart, STEP, IGES, ThreeDM, SAT, STL_in, STL_cm, STL_mm)
HideEditor	True to hide the editor

Member Function Documentation

◆ Add3DSketch()

```
Sketch3D Add3DSketch ( string  Name )
```

Creates a new 3D sketch

Parameters

Name	Name of sketch

Returns

Created sketch

◆ AddAxis() [1/4]

```
Axis AddAxis ( string  Name,
               Face    CylindricalFace
             )
```

Creates an axis for a cylindrical face

Parameters

Name	Name of axis
CylindricalFace	Cylindrical face

Returns

New axis

◆ AddAxis() [2/4]

```
Axis AddAxis ( string  Name,
               List []  Point1,
               List []  Point2
             )
```

Creates an axis based on two points

Parameters

Name Name of axis

Point1 First point [X, Y, Z]

Point2 Second point [X, Y, Z]

Returns

New axis

◆ AddAxis() [3/4]

```
Axis AddAxis ( string            Name,
               ISketchSurface  Plane1,
               ISketchSurface  Plane2
             )
```

Creates an axis based on the intersection of two planes/faces

Parameters

Name Name of axis

Plane1 First plane/face

Plane2 Second plane/face

Returns

New **Axis**

◆ AddAxis() [4/4]

```
Axis AddAxis ( string  Name,
               Point   PointA,
               Point   PointB
             )
```

Creates an axis based on two points

Parameters

Name Name of axis

PointA First point object

PointB Second point object

Returns

New axis

◆ AddChamfer() [1/4]

```
Feature AddChamfer ( string          Name,
                     IChamferable    Item,
                     double          Distance,
                     bool            TangentPropagate
                   )
```

Adds a chamfer to a face or edge

Parameters

Name Name of chamfer

Item **Face** or edge to chamfer

Distance Chamfer distance

TangentPropagate True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddChamfer() [2/4]

```
Feature AddChamfer ( string          Name,
                     IChamferable    Item,
                     double          Distance1,
                     double          Distance2,
                     bool            TangentPropagate
                   )
```

Adds a chamfer to a face or edge

Parameters

Name	Name of chamfer
Item	**Face** or edge to chamfer
Distance1	First chamfer distance
Distance2	Second chamfer distance
TangentPropagate	True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddChamfer() [3/4]

```
Feature AddChamfer ( string    Name,
                     List []   Items,
                     double    Distance,
                     bool      TangentPropagate
                   )
```

Adds a chamfer to a set of faces and edges

Parameters

Name	Name of chamfer
Items	Faces and edges to chamfer
Distance	Chamfer distance
TangentPropagate	True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddChamfer() [4/4]

```
Feature AddChamfer ( string   Name,
                     List []  Items,
                     double   Distance1,
                     double   Distance2,
                     bool     TangentPropagate
                   )
```

Adds a chamfer to a set of faces and edges

Parameters

Name	Name of chamfer
Items	Faces and edges to chamfer
Distance1	First chamfer distance
Distance2	Second chamfer distance
TangentPropagate	True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddChamferAngle() [1/2]

```
Feature AddChamferAngle ( string          Name,
                          IChamferable    Item,
                          double          Distance,
                          double          Angle,
                          bool            TangentPropagate
                        )
```

Adds a chamfer to a face or edge

Parameters

Name	Name of chamfer
Item	**Face** or edge to chamfer
Distance	Chamfer distance
Angle	Chamfer angle
TangentPropagate	True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddChamferAngle() [2/2]

```
Feature AddChamferAngle ( string   Name,
                          List []  Items,
                          double   Distance,
                          double   Angle,
                          bool     TangentPropagate
                        )
```

Adds a chamfer to a set of faces and edges

Parameters

Name	Name of chamfer
Items	Faces and edges to chamfer
Distance	Chamfer distance
Angle	Chamfer angle
TangentPropagate	True to propagate the chamfer along connected edges

Returns

Chamfer feature

◆ AddConfiguration() [1/2]

```
Configuration AddConfiguration ( string  Name )
```

Adds a configuration to the part

Parameters

Name	Name of configuration

Returns

New configuration

◆ AddConfiguration() [2/2]

```
Configuration AddConfiguration ( string  Name,
                                 string  BaseConfigurationName
                               )
```

Adds a configuration to the part using another configuration as a base

Parameters

Name	Name of configuration
BaseConfigurationName	Name of base configuration to use

Returns

New configuration

◆ AddExtrudeBoss() [1/2]

```
Feature AddExtrudeBoss ( string  Name,
                         Sketch  Sketch,
                         double  Depth,
                         bool    IsReversed
                       )
```

Adds a simple extrude boss to a specific depth

Parameters

Name	Name of extrusion
Sketch	**Sketch** to extrude
Depth	Extrusion distance
IsReversed	True if extrusion direction is reversed

Returns

Extruded feature

◆ AddExtrudeBoss() [2/2]

```
Feature AddExtrudeBoss ( string          Name,
                         Sketch          Sketch,
                         double          Depth,
                         bool            IsReversed,
                         EndCondition    EndCondition,
                         ISketchSurface  EndPlane,
                         double          EndOffset,
                         DirectionType   Direction,
                         ISweepPath      SweepPath,
                         double          DraftAngle,
                         bool            OutwardDraft
                       )
```

Adds an extrude feature

Parameters

Name	Name of extrusion
Sketch	**Sketch** to extrude
Depth	Depth of extrusion
IsReversed	true if direction is reversed
EndCondition	End condition for extrusion
EndPlane	**Face** or plane to terminate extrusion
EndOffset	Offset from face or plane to terminate extrusion
Direction	Direction of extrusion
SweepPath	**Sketch** or edge to follow when extruding
DraftAngle	Angle of draft
OutwardDraft	true if outward draft

Returns

Extruded feature

◆ AddExtrudeCut() [1/2]

```
Feature AddExtrudeCut ( string   Name,
                        Sketch   Sketch,
                        double   Depth,
                        bool     IsReversed
                      )
```

Adds a simple extrude cut to a specific depth

Parameters

Name	Name of extrusion
Sketch	**Sketch** to extrude
Depth	Extrusion distance
IsReversed	True if extrusion direction is reversed

Returns

Extruded feature

◆ AddExtrudeCut() [2/2]

```
Feature AddExtrudeCut ( string          Name,
                        Sketch          Sketch,
                        double          Depth,
                        bool            IsReversed,
                        EndCondition    EndCondition,
                        ISketchSurface  EndPlane,
                        double          EndOffset,
                        DirectionType   Direction,
                        ISweepPath      SweepPath,
                        double          DraftAngle,
                        bool            OutwardDraft
                      )
```

Adds an extrude cut feature

Parameters

Name	Name of extrusion
Sketch	**Sketch** to extrude
Depth	Depth of extrusion
IsReversed	true if direction is reversed
EndCondition	End condition for extrusion
EndPlane	**Face** or plane to terminate extrusion
EndOffset	Offset from face or plane to terminate extrusion
Direction	Direction of extrusion
SweepPath	**Sketch** or edge to follow when extruding
DraftAngle	Angle of draft
OutwardDraft	true if outward draft

Returns

Extruded feature

◆ AddFillet() [1/3]

```
Feature AddFillet ( string      Name,
                    IFilletable Item,
                    double      Radius,
                    bool        TangentPropagate
                  )
```

Adds a constant radius fillet to a face or edge

Parameters

Name	Name of fillet
Item	**Face** or edge to fillet
Radius	Radius of fillet
TangentPropagate	True to propagate the fillet along connected edges

Returns

Fillet feature

◆ AddFillet() [2/3]

```
Feature AddFillet ( string   Name,
                    List []  Items,
                    double   Radius,
                    bool     TangentPropagate
                  )
```

Adds a constant radius fillet to a set of faces and edges

Parameters

Name	Name of fillet
Items	Faces and edges to fillet
Radius	Radius of fillet
TangentPropagate	True to propagate the fillet along connected edges

Returns

Fillet feature

◆ AddFillet() [3/3]

```
Feature AddFillet ( string  Name,
                    List []  Items,
                    List []  StartRadii,
                    List []  EndRadii,
                    bool    TangentPropagate
                  )
```

Adds a variable radius fillet to a set of faces and edges

Parameters

Name	Name of fillet
Items	Faces and edges to fillet
StartRadii	Start radii of fillets
EndRadii	End radii of fillets
TangentPropagate	True to propagate the fillet along connected edges

Returns

Fillet feature

◆ AddGear()

```
GearSketch AddGear ( string           Name,
                     double           DiametralPitch,
                     int              NumberofTeeth,
                     double           PitchDiameter,
                     double           PressureAngle,
                     bool             SingleTooth,
                     double           CenterX,
                     double           CenterY,
                     int              InvolutePoints,
                     ISketchSurface   Plane
                   )
```

Adds a gear sketch to the part

Parameters

Name	Name of gear sketch
NumberofTeeth	Number of teeth
PitchDiameter	Diameter of pitch circle in current units
PressureAngle	Pressure angle (14.5 is typical)
DiametralPitch	Diametral angle (tooth size) (25.4/module) in teeth per inch
SingleTooth	true to create only a single tooth profile
CenterX	X-coordinate of gear center
CenterY	Y-coordinate of gear center
InvolutePoints	Number of points for involute curve. Decreasing this makes Cubify/Geomagic faster. Increasing makes tooth profiles more accurate and allows gears with more teeth to be generated.
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearDN() [1/2]

```
GearSketch AddGearDN ( string          Name,
                       double          DiametralPitch,
                       int             NumberofTeeth,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       bool            SingleTooth,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using diametral pitch and number of teeth

Parameters

Name	Name of gear sketch
NumberofTeeth	Number of teeth
PressureAngle	Pressure angle (14.5 is typical)
DiametralPitch	Diametral angle (tooth size) (1/module)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
SingleTooth	True to generate a single tooth
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearDN() [2/2]

```
GearSketch AddGearDN ( string          Name,
                       double          DiametralPitch,
                       int             NumberofTeeth,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using diametral pitch and number of teeth

Parameters

Name	Name of gear sketch
NumberofTeeth	Number of teeth
PressureAngle	Pressure angle (14.5 is typical)
DiametralPitch	Diametral angle (tooth size) (1/module)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearDP() [1/2]

```
GearSketch AddGearDP ( string          Name,
                       double          DiametralPitch,
                       double          PitchDiameter,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       bool            SingleTooth,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using diametral pitch and pitch diameter

Parameters

Name	Name of gear sketch
PitchDiameter	Diameter of pitch circle
PressureAngle	Pressure angle (14.5 is typical)
DiametralPitch	Diametral angle (tooth size) (1/module)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
SingleTooth	True to generate a single tooth
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearDP() [2/2]

```
GearSketch AddGearDP ( string          Name,
                       double          DiametralPitch,
                       double          PitchDiameter,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using diametral pitch and pitch diameter

Parameters

Name	Name of gear sketch
PitchDiameter	Diameter of pitch circle
PressureAngle	Pressure angle (14.5 is typical)
DiametralPitch	Diametral angle (tooth size) (1/module)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearNP() [1/2]

```
GearSketch AddGearNP ( string          Name,
                       int             NumberofTeeth,
                       double          PitchDiameter,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       bool            SingleTooth,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using number of teeth and pitch diameter

Parameters

Name	Name of gear sketch
NumberofTeeth	Number of teeth
PitchDiameter	Diameter of pitch circle
PressureAngle	Pressure angle (14.5 is typical)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
SingleTooth	True to generate a single tooth
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddGearNP() [2/2]

```
GearSketch AddGearNP ( string          Name,
                       int             NumberofTeeth,
                       double          PitchDiameter,
                       double          PressureAngle,
                       double          CenterX,
                       double          CenterY,
                       ISketchSurface  Plane
                     )
```

Adds a gear sketch to the part using number of teeth and pitch diameter

Parameters

Name	Name of gear sketch
NumberofTeeth	Number of teeth
PitchDiameter	Diameter of pitch circle
PressureAngle	Pressure angle (14.5 is typical)
CenterX	X-coordinate of center of gear
CenterY	Y-coordinate of center of gear
Plane	**Plane** or face to create gear sketch on

Returns

Gear sketch

◆ AddLoftBoss() [1/2]

```
Feature AddLoftBoss ( string  Name,
                      List []  CrossSections,
                      bool    MinimizeTwist,
                      bool    MinimizeCurvature,
                      bool    SimplifySurface,
                      bool    ConnectEnds
                    )
```

Adds a loft extrusion

Parameters

TangentAngles List of tangent angles in degrees

Parameters

Name	Name of loft
CrossSections	Python list of cross sections (faces, 2D sketches, design points)
MinimizeTwist	True to minimize twist
MinimizeCurvature	True to minimize curvature
SimplifySurface	True to simplify the loft surface
ConnectEnds	True to connect the start of the loft with the end

Returns

Extruded feature

◆ AddLoftBoss() [2/2]

```
Feature AddLoftBoss ( string              Name,
                      List []             CrossSections,
                      List []             GuideCurves,
                      GuideCurveTypes     GuideType,
                      bool                MinimizeTwist,
                      bool                MinimizeCurvature,
                      bool                SimplifySurface,
                      bool                ConnectEnds
                    )
```

Adds a loft extrusion using guide curves

Parameters

Name	Name of loft
CrossSections	Python list of cross sections (faces, 2D sketches, design points)
GuideCurves	Python list of guide curves (3D sketches)
GuideType	Type of guide curve
MinimizeTwist	True to minimize twist
MinimizeCurvature	True to minimize curvature
SimplifySurface	True to simplify the loft surface
ConnectEnds	True to connect the start of the loft with the end

Returns

Extruded feature

◆ AddLoftCut() [1/2]

```
Feature AddLoftCut ( string  Name,
                     List []  CrossSections,
                     bool    MinimizeTwist,
                     bool    MinimizeCurvature,
                     bool    SimplifySurface,
                     bool    ConnectEnds
                   )
```

Adds a loft cut

Parameters

TangentAngles List of tangent angles in degrees

Parameters

Name	Name of loft
CrossSections	Python list of cross sections (faces, 2D sketches, design points)
MinimizeTwist	True to minimize twist
MinimizeCurvature	True to minimize curvature
SimplifySurface	True to simplify the loft surface
ConnectEnds	True to connect the start of the loft with the end

Returns

Cut feature

◆ AddLoftCut() [2/2]

```
Feature AddLoftCut ( string          Name,
                     List []         CrossSections,
                     List []         GuideCurves,
                     GuideCurveTypes GuideType,
                     bool            MinimizeTwist,
                     bool            MinimizeCurvature,
                     bool            SimplifySurface,
                     bool            ConnectEnds
                   )
```

Adds a loft cut using guide curves

Parameters

Name	Name of loft
CrossSections	Python list of cross sections (faces, 2D sketches, design points)
GuideCurves	Python list of guide curves (3D sketches)
GuideType	Type of guide curve
MinimizeTwist	True to minimize twist
MinimizeCurvature	True to minimize curvature
SimplifySurface	True to simplify the loft surface
ConnectEnds	True to connect the start of the loft with the end

Returns

Extruded feature

◆ AddParameter() [1/3]

```
Parameter AddParameter ( string         Name,
                         ParameterTypes Type,
                         double         Value
                       )
```

Adds a cm/mm/in/deg parameter to the part

Parameters

Name Name of parameter

Type Type of parameter

Value Value for parameter

Returns

New parameter

◆ AddParameter() [2/3]

```
Parameter AddParameter ( string         Name,
                         ParameterTypes Type,
                         ParameterUnits UnitstoUse,
                         double         Value
                       )
```

Adds a parameter to the part with specific units

Parameters

Name Name of parameter

Type Type of parameter

UnitstoUse Units to use

Value Value for parameter

Returns

New parameter

◆ AddParameter() [3/3]

```
Parameter AddParameter ( string          Name,
                         ParameterTypes  Type,
                         string          Equation
                       )
```

Adds a parameter to the part

NOTE: DOESN'T SEEM TO WORK IN GD V16 - THROWS EXCEPTION ABOUT TRANSACTION ALREADY BEING OPEN

Parameters

Name Name of parameter

Type Type of parameter

Equation Equation for parameter

Returns

New parameter

◆ AddPlane() [1/5]

```
Plane AddPlane ( string  Name,
                 Axis    Axis,
                 Point   Point
               )
```

Creates a new plane contaning an axis and a point

Parameters

Name Name of new plane

Axis **Axis** that lies on plane

Point **Point** that lies on plane

Returns

New plane

◆ AddPlane() [2/5]

```
Plane AddPlane ( string Name,
                 List [] NormalVector,
                 List [] PointonPlane
               )
```

Adds a plane using a normal vector and a point on the plane

Parameters

Name	Name of plane to add
NormalVector	Normal vector as a list [nx, ny, nz]. Does not need to be a unit vector
PointonPlane	A point on the plane as a list [px, py, pz]

Returns

Created plane

◆ AddPlane() [3/5]

```
Plane AddPlane ( string Name,
                 List [] Point1,
                 List [] Point2,
                 List [] Point3
               )
```

Creates a plane using three points. Each point is defined as list of [x, y, z]

Parameters

Name	Name of plane
Point1	**Point** on plane
Point2	**Point** on plane
Point3	**Point** on plane

Returns

Created plane

◆ AddPlane() [4/5]

```
Plane AddPlane ( string          Name,
                 ISketchSurface  SourcePlane,
                 Axis            RotationAxis,
                 double          Angle
               )
```

Creates a new plane at an angle to an existing plane

Parameters

Name	Name of new plane
SourcePlane	Plane/face to use as basis for new plane
RotationAxis	**Axis** of rotation for new plane
Angle	Angle of new plane in degrees

Returns

New plane

◆ AddPlane() [5/5]

```
Plane AddPlane ( string          Name,
                 ISketchSurface  SourcePlane,
                 double          Offset
               )
```

Creates a plane based on the offset from an existing plane

Parameters

Name	Name of plane
SourcePlane	Plane/face to use as basis
Offset	Offset from basis plane in currently chosen units

Returns

Created plane

◆ AddPoint() [1/10]

```
Point AddPoint ( string  Name,
                 double  X,
                 double  Y,
                 double  Z
               )
```

Adds a point to the part

Parameters

Name Name of new point

X X coordinate

Y Y coordinate

Z Z coordinate

Returns

The new point

◆ AddPoint() [2/10]

```
Point AddPoint ( string  Name,
                 Edge    TargetEdge,
                 double  Ratio
               )
```

Add a point on an edge

Parameters

Name Name of point

TargetEdge The edge to create the point on

Ratio Ratio along the edge from 0.0 -> 1.0

Returns

The created point

◆ AddPoint() [3/10]

```
Point AddPoint ( string  Name,
                 IAxis   AxisOrEdge,
                 IPlane  PlaneOrFace
               )
```

Add a point at the the intersection of a axis or edge and a plane or face

Parameters

Name Name of point

AxisOrEdge **Axis** or edge

PlaneOrFace **Plane** or face

Returns

The created point

◆ AddPoint() [4/10]

```
Point AddPoint ( string  Name,
                 IAxis   AxisOrEdge1,
                 IAxis   AxisOrEdge2
               )
```

Add a point at the intersection or two axes or edges

Parameters

Name Name of point

AxisOrEdge1 First axis or edge

AxisOrEdge2 Second axis or edge

Returns

The created point

◆ AddPoint() [5/10]

```
Point AddPoint ( string  Name,
                 IPlane  PlaneOrFace1,
                 IPlane  PlaneOrFace2,
                 IPlane  PlaneOrFace3
               )
```

Add a point at the intersection of three planes or faces

Parameters

Name	Name of point
PlaneOrFace1	First plane or face
PlaneOrFace2	Second plane or face
PlaneOrFace3	Third plane or face

Returns

The created point

◆ AddPoint() [6/10]

```
Point AddPoint ( string  Name,
                 IPoint  PointOrVertex,
                 double  XOffset,
                 double  YOffset,
                 double  ZOffset
               )
```

Add a point at an offset to a point or a vertex

Parameters

Name	Name of point
PointOrVertex	**Point** or vertex
XOffset	X offse
YOffset	Y offset
ZOffset	Z offset

Returns

The created point

◆ AddPoint() [7/10]

```
Point AddPoint ( string  Name,
                 IPoint  PointOrVertex1,
                 IPoint  PointOrVertex2,
                 double  Ratio
               )
```

Add a point between two points/vertices

Parameters

Name	Name of point
PointOrVertex1	First point or vertex
PointOrVertex2	Second point or vertex
Ratio	Ratio of distance between points/vertices

Returns

The created point

◆ AddPoint() [8/10]

```
Point AddPoint ( string  Name,
                 IPoint  SourcePointOrVertex,
                 IPlane  TargetPlaneOrFace,
                 double  XOffset,
                 double  YOffset
               )
```

Add a point by projecting a point or vertex onto a plane or face

Parameters

Name	Name of point
SourcePointOrVertex	**Point** or vertex to project
TargetPlaneOrFace	**Plane** or face to project onto
XOffset	X offset to apply to point once projected
YOffset	Y offset to apply to point once projected

Returns

The created point

◆ AddPoint() [9/10]

```
Point AddPoint ( string  Name,
                 List []  Point
               )
```

Adds a point to the part

Parameters

Name Name of the new point

Point **Point** location [x, y, z]

Returns

The new point

◆ AddPoint() [10/10]

```
void AddPoint ( string  Name,
                Point  Point
              )
```

Adds a point to the part

Parameters

Name Name of the point

Point **Point** to add

◆ AddPointFromCircularEdge()

```
Point AddPointFromCircularEdge ( string  Name,
                                 Edge  TargetEdge
                               )
```

Adds a point at the center of a circular edge

Parameters

Name Name of point

TargetEdge The edge to use for creating the point

Returns

The created point

◆ AddPointFromToroidalFace()

```
Point AddPointFromToroidalFace ( string  Name,
                                 Face    TargetFace
                               )
```

Adds a point at the center of a toroidal face

Parameters

Name Name of point

TargetFace Toroidal face to use in creating the point

Returns

The created point

◆ AddPoints()

```
void AddPoints ( string  Prefix,
                 List []  Points
               )
```

Adds a set of points to the part

Parameters

Prefix Prefix for the point names

Points List of points [x1,y1,z1, ..., xn,yn,zn]

◆ AddRevolveBoss()

```
Feature AddRevolveBoss ( string   Name,
                         Sketch   Sketch,
                         Axis     Axis,
                         double   Angle
                       )
```

Creates a revolve boss feature

Parameters

Name	Name of feature
Sketch	**Sketch** to revolve
Axis	**Axis** to rotate around
Angle	Rotation angle in degrees

Returns

Created feature

◆ AddRevolveCut()

```
Feature AddRevolveCut ( string   Name,
                        Sketch   Sketch,
                        Axis     Axis,
                        double   Angle
                      )
```

Creates a revolve cut feature

Parameters

Name	Name of feature
Sketch	**Sketch** to revolve
Axis	**Axis** to rotate around
Angle	Rotation angle in degrees

Returns

Created feature

◆ AddSketch()

```
Sketch AddSketch ( string          Name,
                   ISketchSurface  Plane
                 )
```

Creates a new sketch using a plane/face

Parameters

Name Name of sketch

Plane Plane/face to use for sketch

Returns

Created sketch

◆ AddSweepBoss()

```
Feature AddSweepBoss ( string          Name,
                       Sketch          ProfileSketch,
                       ISweepPath      PathSketch,
                       bool            IsRigid,
                       EndCondition    EndCondition,
                       ISketchSurface  EndPlane,
                       double          EndOffset,
                       double          DraftAngle,
                       bool            OutwardDraft
                     )
```

Adds a sweep extrude feature

Parameters

Name	Name of extrusion
ProfileSketch	**Sketch** to extrude
PathSketch	**Sketch** or edge to sweep along
IsRigid	true if path is parallel to profile
EndCondition	End condition for extrusion
EndPlane	**Face** or plane to terminate extrusion
EndOffset	Offset from face or plane to terminate extrusion
DraftAngle	Angle of draft
OutwardDraft	true if outward draft

Returns

Extruded feature

◆ AddSweepCut()

```
Feature AddSweepCut ( string          Name,
                      Sketch          ProfileSketch,
                      ISweepPath      PathSketch,
                      bool            IsRigid,
                      EndCondition    EndCondition,
                      ISketchSurface  EndPlane,
                      double          EndOffset,
                      double          DraftAngle,
                      bool            OutwardDraft
                    )
```

Adds a sweep extrude cut feature

Parameters

Name	Name of extrusion
ProfileSketch	**Sketch** to extrude
PathSketch	**Sketch** or edge to sweep along
IsRigid	true if path is parallel to profile
EndCondition	End condition for extrusion
EndPlane	**Face** or plane to terminate extrusion
EndOffset	Offset from face or plane to terminate extrusion
DraftAngle	Angle of draft
OutwardDraft	true if outward draft

Returns

Extruded feature

◆ AddVertexChamfer() [1/2]

```
Feature AddVertexChamfer ( string  Name,
                           List []  Items,
                           double  Distance1,
                           double  Distance2,
                           double  Distance3
                         )
```

Adds a chamfer to a set of vertices

Parameters

Name	Name of chamfer
Items	Vertices to chamfer
Distance1	First chamfer distance
Distance2	Second chamfer distance
Distance3	Third chamfer distance

Returns

Chamfer feature

◆ AddVertexChamfer() [2/2]

```
Feature AddVertexChamfer ( string  Name,
                           Vertex  Item,
                           double  Distance1,
                           double  Distance2,
                           double  Distance3
                         )
```

Adds a chamfer to a vertex

Parameters

Name	Name of chamfer
Item	**Vertex** to chamfer
Distance1	First chamfer distance
Distance2	Second chamfer distance
Distance3	Third chamfer distance

Returns

Chamfer feature

◆ DisplayUnits()

UnitTypes DisplayUnits ( )

Gets the display units for the part

Returns

The display units

◆ ExportBIP()

void ExportBIP (string FileName)

Exports a keyshot file

Parameters

FileName Path and name of keyshot file

◆ ExportIGES()

void ExportIGES (string FileName)

Exports the part as a IGES file

Parameters

FileName Path and name of IGES file

◆ ExportRotatedSTL()

```
void ExportRotatedSTL ( string  FileName,
                        Face    BottomFace,
                        bool    ForcetoMillimeters,
                        bool    UseCustomSettings,
                        double  MaxCellSize,
                        double  NormalDeviation,
                        double  SurfaceDeviation
                      )
```

Exports the part as an STL rotated so that a specific face is on the bottom

Parameters

FileName	Path and name of STL file
BottomFace	**Face** to use as bottom of part
ForcetoMillimeters	true to output STL in millimeters regardless of part units
UseCustomSettings	true to use custom STL settings, false to use settings in system properties
MaxCellSize	Custom max cell size
NormalDeviation	Custom normal deviation
SurfaceDeviation	Custom surface deviation

◆ ExportSAT()

```
void ExportSAT ( string  FileName,
                 int     Version,
                 bool    SaveColors
               )
```

Exports the part as a SAT file

Parameters

FileName	Path and name of SAT file
Version	Exported SAT file version
SaveColors	true to preseve colors

◆ ExportSTEP203()

void ExportSTEP203 (string FileName)

Exports the part as a STEP 203 file

Parameters

FileName Path and name of STEP 203 file

◆ ExportSTEP214()

void ExportSTEP214 (string FileName)

Exports the part as a STEP 214 file

Parameters

FileName Path and name of STEP 214 file

◆ ExportSTL()

void ExportSTL (string FileName)

Exports the part as an STL file

Parameters

FileName Path and name of STL file

◆ Get3DSketch()

Sketch3D Get3DSketch (string Name)

Gets a sketch using the name of the sketch

Parameters

Name Name of sketch

Returns

Sketch object

◆ GetActiveConfiguration()

Configuration GetActiveConfiguration ( )

Gets the currently active configuration

Returns

Configuration object

◆ GetAxis()

Axis GetAxis (string Name)

Gets an axis from an axis name

Parameters

Name Name of axis to find

Returns

Found axis

◆ GetBoundingBox()

List [] GetBoundingBox ( )

Gets the bounding box for the part as eight points

Returns

Python list of eight points as [P1, P2, ... P8]. Each point is [X, Y, Z]

◆ GetConfiguration()

Configuration GetConfiguration (string Name)

Gets a configuration with a specific name

Parameters

Name Name of confguration

Returns

Configuration object

◆ GetCustomProperty()

string GetCustomProperty (string Name)

Gets the value of a custonm property

Parameters

Name Name of the custom property

Returns

The value of the property as a string

◆ GetEdge()

Edge GetEdge (string Name)

Gets an edge using it's name "Edge<n>"

Parameters

Name Name of edge

Returns

Edge if found

◆ GetEdges()

List [] GetEdges ( )

Gets a python list of the current edges in the part

Returns

Python list of edges

◆ GetFace()

Face GetFace (string Name)

Gets a face using it's name "Face<n>"

Parameters

Name Name of face

Returns

Face if found

◆ GetFaces()

List [] GetFaces ( )

Gets a python list of the current faces in the part

Returns

Python list of faces

◆ GetFeature()

Feature GetFeature (string Name)

Gets a feature on the part

Parameters

Name Name of the feature to get

Returns

The feature or null if not found

◆ GetParameter()

Parameter GetParameter (string Name)

Gets a parameter with a specific name

Parameters

Name Name of parameter

Returns

Parameter object

◆ GetPlane()

Plane GetPlane (string Name)

Gets a plane using the name of the plane

Parameters

Name Name of plane to find

Returns

The plane

◆ GetPoint()

Point GetPoint (string Name)

Gets a point on the part using the point name. The point must have been created in a script

Parameters

Name Name of point to get

Returns

Point on the part

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the part was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ GetSketch()

Sketch GetSketch (string Name)

Gets a sketch using the name of the sketch

Parameters

Name Name of sketch

Returns

Sketch object

◆ GetUserData()

IronPython.Runtime.PythonDictionary GetUserData (string Name)

Gets user data

Parameters

Name Name of data to get

Returns

Data as a python dictionary or None if not found

◆ GetVertex()

Vertex GetVertex (string Name)

Gets a vertex using it's name "Vertex<n>"

Parameters

Name Name of vertex

Returns

Vertex if found

◆ GetVertices()

List [] GetVertices ( )

Gets a python list of the current vertices in the part

Returns

Python list of vertices

◆ HideFeature() [1/2]

void HideFeature (**Feature** Feature)

Hides a feature on the part

Parameters

Feature **Feature** to hide

◆ HideFeature() [2/2]

void HideFeature (string Name)

Hides a feature on the part

Parameters

Name Name of the feature to hide

◆ IsOpen()

bool IsOpen ( )

Checks if the part is opened

Returns

◆ NonUniformScale()

```
Feature NonUniformScale ( string Name,
                          bool   ScaleAboutCenter,
                          double ScaleFactorX,
                          double ScaleFactorY,
                          double ScaleFactorZ
                        )
```

Non-uniform scaling of the part

Parameters

Name	Name of the scaling
ScaleAboutCenter	true to scale around the center of the part
ScaleFactorX	X scale factor
ScaleFactorY	Y scale factor
ScaleFactorZ	Z scale factor

Returns

Scale feature

◆ RemoveFeature() [1/2]

void RemoveFeature (**Feature** Feature)

Removes a feature from the part

Parameters

Feature **Feature** to remove

◆ RemoveFeature() [2/2]

void RemoveFeature (string Name)

Removes a feature from the part

Parameters

Name Name of the feature to remove

◆ RemovePlane()

void RemovePlane (**Plane** Plane)

Removes a plane from the part

Parameters

Plane **Plane** to remove

◆ RemovePoint()

void RemovePoint (**Point** Point)

Removes a point from the part

Parameters

Point **Point** to remove

◆ RemoveSketch() [1/2]

void RemoveSketch (**Sketch** Sketch)

Removes a sketch from the part

Parameters

Sketch **Sketch** to remove

◆ RemoveSketch() [2/2]

void RemoveSketch (string Name)

Removes a sketch from the part

Parameters

Name Name of sketch to remove

◆ Save()

void Save (string Folder)

Saves the part to a specific folder

Parameters

Folder Folder to save to

◆ SaveAs()

void SaveAs (string Folder,
string NewName
)

Saves the part to a specific folder with a new name

Parameters

Folder Folder to save to

NewName New name for part

◆ SaveSnapshot()

```
void SaveSnapshot ( string  FileName,
                    int     Width,
                    int     Height,
                    bool    UseAspectRatio,
                    bool    UseWidthandHeight
                  )
```

Saves the current view as a bitmap image

Parameters

FileName	Path and name of file to save to
Width	Width in pixels
Height	Height in pixels
UseAspectRatio	if true uses greater of width/height along with current aspect ratio
UseWidthandHeight	if true uses current width/height of view

◆ SaveThumbnail()

```
void SaveThumbnail ( string  FileName,
                     int     Width,
                     int     Height
                   )
```

Saves a thumbnail image of the part

Parameters

FileName	Path and name of file to save to
Width	Width of thumbnail in pixels
Height	Height of thumbnail in pixels

◆ Scale()

```
Feature Scale ( string  Name,
                bool    ScaleAboutCenter,
                double  ScaleFactor
              )
```

Uniform scaling of the part

Parameters

Name	Name of the scaling
ScaleAboutCenter	true to scale around the center of the part
ScaleFactor	Scale factor

Returns

Scale feature

◆ Select() [1/2]

```
void Select ( List []  FacesEdgesList )
```

Selects a group of faces, edges, vertices, points, axes, planes and sketches

Parameters

FacesEdgesList	List of Faces, edges, vertices, points, axes, planes and sketches to select [FaceA, FaceB, EdgeA, EdgeB, ...]

◆ Select() [2/2]

```
void Select ( ISelectableGeometry  FaceorEdge )
```

Selects a face, edge, vertex, point, axis, plane, sketch

Parameters

FaceorEdge	**Face**, edge, vertex, point, axis plane or sketch to select

◆ SetColor()

```
void SetColor ( byte  Red,
                byte  Green,
                byte  Blue
              )
```

Sets the color of the part

Parameters

Red Red component 0 - 255

Green Green component 0 - 255

Blue Blue component 0 - 255

◆ SetCustomProperty()

```
void SetCustomProperty ( string  Name,
                         string  Value
                       )
```

Sets the value of a custom property The custom property must already be defined on the part or defined on the user's PC

Parameters

Name Name of the custom property

Value New value for the custom property

◆ SetUserData()

```
void SetUserData ( string                                Name,
                   IronPython.Runtime.PythonDictionary   Dict
                 )
```

Sets user data

Parameters

Name Data name of the format companyname.projectname.dataname

Dict Python dictionary of data to store

◆ ShowFeature() [1/2]

void ShowFeature (**Feature** Feature)

Shows a feature on the part

Parameters

Feature **Feature** to show

◆ ShowFeature() [2/2]

void ShowFeature (string Name)

Shows a feature on the part

Parameters

Name Name of the feature to show

◆ SuppressFeature() [1/2]

void SuppressFeature (**Feature** Feature)

Suppresses a feature on the part

Parameters

Feature **Feature** to suppress

◆ SuppressFeature() [2/2]

void SuppressFeature (string Name)

Suppresses a feature on the part

Parameters

Name Name of the feature to suppress

◆ UnsuppressFeature() [1/2]

void UnsuppressFeature (**Feature** Feature)

Unsuppresses a feature on the part

Parameters

Feature **Feature** to unsuppress

◆ UnsuppressFeature() [2/2]

void UnsuppressFeature (string Name)

Unsuppresses a feature on the part

Parameters

Name Name of the feature to unsuppress

Plane Class Reference

A design plane. Can be used for creating sketches More...

Inherits ISketchSurface, IConstrainable, IInstance, ISelectableGeometry, and IPlane.

Public Member Functions

Part	**GetPart** () Gets the part that defined this plane
Assembly	**GetSelectionAssembly** () The assembly that the edge was selected on Only valid when a selection has been made More...
void	**Hide** () Hides the plane
bool	**IsParallel** (**Plane** OtherPlane) Checks if another plane is parallel to this one More...
void	**Show** () Shows the plane

Properties

string	**Name** [get] The name of the plane

Detailed Description

A design plane. Can be used for creating sketches

Member Function Documentation

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ IsParallel()

bool IsParallel (**Plane** OtherPlane)

Checks if another plane is parallel to this one

Parameters

OtherPlane The other plane to check

Returns

true if the planes are parallel

Point Class Reference

A design point More...

Inherits ICrossSection, IConstrainable, IInstance, ISelectableGeometry, and IPoint.

Public Member Functions

List [] **GetCoordinates** ()

Gets the coordiates of the point as a list [X, Y, Z] More...

Part **GetPart** ()

Gets the part that the point is defined in More...

Assembly **GetSelectionAssembly** ()

The assembly that the edge was selected on Only valid when a selection has been made More...

void **Hide** ()

Hides the point

void **Show** ()

Shows the point

Properties

string **Name** `[get]`

Name of the point

double **X** `[get]`

Point X coordinate

double **Y** `[get]`

Point Y coordinate

double **Z** `[get]`

Point Z coordinate

Detailed Description

A design point

Member Function Documentation

◆ GetCoordinates()

List [] GetCoordinates ( )

Gets the coordiates of the point as a list [X, Y, Z]

Returns

List of coordinates [X, Y, Z]

◆ GetPart()

Part GetPart ( )

Gets the part that the point is defined in

Returns

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

Polyline Class Reference

A line constructed from a set of line segments More...

Public Member Functions

Polyline ()

Creates a new 2D polyline that can be later added to a 2D sketch

Polyline (List [] Points)

Creates a new 2D polyline that can be later added to a 2D sketch More...

void **AddArc** (**PolylinePoint** Center, **PolylinePoint** Start, **PolylinePoint** End, int MinimumSegments)

Adds an arc to the polyline. The arc is approcimated with straight line segments More...

void **AddCircle** (double CenterX, double CenterY, double Diameter, int sides)

Adds a circle to the line More...

void **AddPoint** (**PolylinePoint Point**)

Adds a new point to the polyline More...

void **AddPolyline** (**Polyline** AppendLine)

Appends a line to the current line More...

Polyline **Clone** ()

Creates an exact copy of the line More...

Polyline **Clone** (int StartIndex, int EndIndex)

Creates an exact copy of a section of the line More...

void **InsertPoint** (int Index, **PolylinePoint Point**)

Inserts a point at a specific location More...

Polyline **Join** (**Polyline** AppendLine)

Joins a line onto the end of the current line and returns the new line More...

void **Offset** (double OffsetX, double OffsetY)

Applies an offset to all points on the line More...

void **RemoveDuplicates** ()

Removes duplicate points that are next to each other

void	**RotateZ** (double CenterX, double CenterY, double Angle) Rotates the polyline around the Z axis More...
Polyline[]	**SplitAtPoint** (**PolylinePoint** SplitPoint, double Tolerence) Splits a polyline at a point, creating two polylines More...

Static Public Member Functions

static **PolylinePoint**	**FindIntersection** (**Polyline** L1, **Polyline** L2) Finds the first intersection point between two lines More...
static **PolylinePoint**	**FindIntersection** (**PolylinePoint** A1, **PolylinePoint** A2, **PolylinePoint** B1, **PolylinePoint** B2) Gets the intersection between the line segments A1A2 and B1B2 More...
static **PolylinePoint**	**FindIntersectionWithCircle** (**Polyline** L1, double CircleX, double CircleY, double Radius) Finds first intersection of line with a circle More...
static bool	**IsPointOnLine** (**PolylinePoint** A1, **PolylinePoint** A2, **PolylinePoint Point**, double Tolerance) Determines if a point is on a line segment More...

Detailed Description

A line constructed from a set of line segments

Constructor & Destructor Documentation

◆ Polyline()

Polyline (List [] Points)

Creates a new 2D polyline that can be later added to a 2D sketch

Parameters

Points List of points in the polyline [X1, Y1, X2, Y2, ...]

Member Function Documentation

◆ AddArc()

```
void AddArc ( PolylinePoint  Center,
              PolylinePoint  Start,
              PolylinePoint  End,
              int            MinimumSegments
            )
```

Adds an arc to the polyline. The arc is approcimated with straight line segments

Parameters

Center	**Point** defining center of arc
Start	**Point** defining start of arc
End	**Point** defining end of arc
MinimumSegments	Minimum number of line segments to use to form arc

◆ AddCircle()

```
void AddCircle ( double  CenterX,
                 double  CenterY,
                 double  Diameter,
                 int     sides
               )
```

Adds a circle to the line

Parameters

CenterX	X coordinate of circle center
CenterY	Y coordinate of circle center
Diameter	Diameter of circle
sides	Number of sides to use to approximate circle

◆ AddPoint()

```
void AddPoint ( PolylinePoint  Point )
```

Adds a new point to the polyline

Parameters

Point	**Point** to add

◆ AddPolyline()

void AddPolyline (**Polyline** AppendLine)

Appends a line to the current line

Parameters

AppendLine **Line** to append

◆ Clone() [1/2]

Polyline Clone ( )

Creates an exact copy of the line

Returns

Copy of line

◆ Clone() [2/2]

Polyline Clone (int StartIndex,
int EndIndex
)

Creates an exact copy of a section of the line

Parameters

StartIndex 0-based index of first point to include in copy

EndIndex 0-based index of last point to include in copy

Returns

Copied line

◆ FindIntersection() [1/2]

static **PolylinePoint** FindIntersection (**Polyline** L1,
Polyline L2
) static

Finds the first intersection point between two lines

Parameters

L1 First line

L2 Second line

Returns

First intersection point or null if none found

◆ FindIntersection() [2/2]

static **PolylinePoint** FindIntersection (**PolylinePoint** A1,
PolylinePoint A2,
PolylinePoint B1,
PolylinePoint B2
) static

Gets the intersection between the line segments A1A2 and B1B2

Uses the method at: http://stackoverflow.com/a/565282/444611

Parameters

A1 First segment start point

A2 First segment end point

B1 Second segment start point

B2 Second segment end point

Returns

Intersection point or null if not found

◆ FindIntersectionWithCircle()

static **PolylinePoint** FindIntersectionWithCircle (**Polyline** L1,
double CircleX,
double CircleY,
double Radius
) static

Finds first intersection of line with a circle

Adapted from: http://stackoverflow.com/questions/1073336/circle-line-collision-detection

Parameters

L1 **Line** to check

CircleX X-coordinate of circle center

CircleY Y-coordinate of circle center

Radius Radius of circle

Returns

Intersection point or null if not found

◆ InsertPoint()

void InsertPoint (int Index,
PolylinePoint Point
)

Inserts a point at a specific location

Parameters

Index 0-based index of location to insert

Point **Point** to insert

◆ IsPointOnLine()

```
static bool IsPointOnLine ( PolylinePoint A1,
                            PolylinePoint A2,
                            PolylinePoint Point,
                            double        Tolerance
                          )
```

static

Determines if a point is on a line segment

Parameters

A1	First point of line segment
A2	Last point of line segment
Point	**Point** to check
Tolerance	Fudge factor

Returns

True if point is on line

◆ Join()

```
Polyline Join ( Polyline AppendLine )
```

Joins a line onto the end of the current line and returns the new line

Parameters

AppendLine	The line to join to the current line

Returns

The new line created from this line plus the appended line

◆ Offset()

```
void Offset ( double OffsetX,
              double OffsetY
            )
```

Applies an offset to all points on the line

Parameters

OffsetX	X offset to apply
OffsetY	Y offset to apply

◆ RotateZ()

```
void RotateZ ( double  CenterX,
               double  CenterY,
               double  Angle
             )
```

Rotates the polyline around the Z axis

Parameters

CenterX X coordinate of center of rotation

CenterY Y coordinate of center of rotation

Angle Number of degrees to rotate

◆ SplitAtPoint()

```
Polyline[] SplitAtPoint ( PolylinePoint  SplitPoint,
                          double         Tolerence
                        )
```

Splits a polyline at a point, creating two polylines

Parameters

SplitPoint **Point** to split at

Tolerence Tolerance to determine if point is on/near line

Returns

List of polylines [A, B]

Polyline3D Class Reference

A 3D line constructed from a set of line segments More...

Public Member Functions

Polyline3D ()

Creates a new 3D polyline that can be later added to a 3D sketch

Polyline3D (List [] Points)

Creates a new 3D polyline that can be later added to a 3D sketch More...

void	**AddPoint** (**PolylinePoint3D Point**) Adds a new point to the polyline More...
void	**AddPolyline** (**Polyline3D** AppendLine) Appends a line to the current line More...
Polyline3D	**Clone** () Creates an exact copy of the line More...
Polyline3D	**Clone** (int StartIndex, int EndIndex) Creates an exact copy of a section of the line More...
void	**InsertPoint** (int Index, **PolylinePoint3D Point**) Inserts a point at a specific location More...
Polyline3D	**Join** (**Polyline3D** AppendLine) Joins a line onto the end of the current line and returns the new line More...
void	**Offset** (double OffsetX, double OffsetY, double OffsetZ) Applies an offset to all points on the line More...
void	**RemoveDuplicates** () Removes duplicate points that are next to each other
Polyline3D[]	**SplitAtPoint** (**PolylinePoint3D** SplitPoint, double Tolerence) Splits a polyline at a point, creating two polylines More...

Static Public Member Functions

static bool	**IsPointOnLine** (**PolylinePoint3D** A, **PolylinePoint3D** B, **PolylinePoint3D** P, double Tolerance) Determines if a point is on a line segment More...

Detailed Description

A 3D line constructed from a set of line segments

Constructor & Destructor Documentation

◆ Polyline3D()

Polyline3D (List [] Points)

Creates a new 3D polyline that can be later added to a 3D sketch

Parameters

Points List of points in the polyline [X1, Y1, Z1, X2, Y2, Z2, ...]

Member Function Documentation

◆ AddPoint()

void AddPoint (**PolylinePoint3D** Point)

Adds a new point to the polyline

Parameters

Point **Point** to add

◆ AddPolyline()

void AddPolyline (**Polyline3D** AppendLine)

Appends a line to the current line

Parameters

AppendLine **Line** to append

◆ Clone() [1/2]

Polyline3D Clone ( )

Creates an exact copy of the line

Returns

Copy of line

◆ Clone() [2/2]

```
Polyline3D Clone ( int  StartIndex,
                   int  EndIndex
                 )
```

Creates an exact copy of a section of the line

Parameters

StartIndex 0-based index of first point to include in copy

EndIndex 0-based index of last point to include in copy

Returns

Copied line

◆ InsertPoint()

```
void InsertPoint ( int              Index,
                   PolylinePoint3D  Point
                 )
```

Inserts a point at a specific location

Parameters

Index 0-based index of location to insert

Point **Point** to insert

◆ IsPointOnLine()

```
static bool IsPointOnLine ( PolylinePoint3D A,
                            PolylinePoint3D B,
                            PolylinePoint3D P,
                            double          Tolerance
                          )                                                    static
```

Determines if a point is on a line segment

Parameters

A	First point of line segment
B	Last point of line segment
P	**Point** to check
Tolerance	Fudge factor

Returns

True if point is on line

◆ Join()

```
Polyline3D Join ( Polyline3D AppendLine )
```

Joins a line onto the end of the current line and returns the new line

Parameters

AppendLine	The line to join to the current line

Returns

The new line created from this line plus the appended line

◆ Offset()

```
void Offset ( double  OffsetX,
              double  OffsetY,
              double  OffsetZ
            )
```

Applies an offset to all points on the line

Parameters

OffsetX X offset to apply

OffsetY Y offset to apply

OffsetZ Z offset to apply

◆ SplitAtPoint()

```
Polyline3D[] SplitAtPoint ( PolylinePoint3D  SplitPoint,
                            double           Tolerence
                          )
```

Splits a polyline at a point, creating two polylines

Parameters

SplitPoint **Point** to split at

Tolerence Tolerance to determine if point is on/near line

Returns

List of polylines [A, B]

PolylinePoint Class Reference

A single point in a polyline More...

Public Member Functions

	PolylinePoint () Creates a new polyline point
	PolylinePoint (double **X**, double **Y**) Creates a new polyline point More...
PolylinePoint	**Offset** (double **X**, double **Y**) Applies an offset to the point and creates a new point More...

void	**RotateZ** (double CenterX, double CenterY, double Angle) Rotates the point around the Z axis More...
PolylinePoint	**Scale** (double ScaleOriginX, double ScaleOriginY, double ScaleFactor) Scales the point location based on an origin for the scaling More...

Public Attributes

double	**X** X coordinate
double	**Y** Y coordinate

Detailed Description

A single point in a polyline

Constructor & Destructor Documentation

◆ PolylinePoint()

PolylinePoint (double X,
double Y
)

Creates a new polyline point

Parameters

X X coordinate

Y Y coordinate

Member Function Documentation

◆ Offset()

```
PolylinePoint Offset ( double  X,
                       double  Y
                     )
```

Applies an offset to the point and creates a new point

Parameters

X X offset to apply

Y Y offset to apply

Returns

New point with offset applied

◆ RotateZ()

```
void RotateZ ( double  CenterX,
               double  CenterY,
               double  Angle
             )
```

Rotates the point around the Z axis

Parameters

CenterX X coordinate of center of rotation

CenterY Y coordinate of center of rotation

Angle Number of degrees to rotate

◆ Scale()

```
PolylinePoint Scale ( double  ScaleOriginX,
                      double  ScaleOriginY,
                      double  ScaleFactor
                    )
```

Scales the point location based on an origin for the scaling

Parameters

ScaleOriginX X-coordinate for scaling origin

ScaleOriginY Y-coordinate for scaling origin

ScaleFactor Factor for scaling as a percentage

Returns

New point with scaling applied

PolylinePoint3D Class Reference

A single point in a polyline for 3D sketches More...

Public Member Functions

PolylinePoint3D ()

Creates a new polyline point

PolylinePoint3D (double **X**, double **Y**, double **Z**)

Creates a new 3D polyline point More...

PolylinePoint3D **Offset** (double **X**, double **Y**, double **Z**)

Applies an offset to the point and creates a new point More...

PolylinePoint3D **Scale** (double ScaleOriginX, double ScaleOriginY, double ScaleOriginZ, double ScaleFactor)

Scales the point location based on an origin for the scaling More...

Public Attributes

double **X**

X coordinate

double **Y**

Y coordinate

double **Z**

Z coordinate

Detailed Description

A single point in a polyline for 3D sketches

Constructor & Destructor Documentation

◆ PolylinePoint3D()

```
PolylinePoint3D ( double X,
                  double Y,
                  double Z
                )
```

Creates a new 3D polyline point

Parameters

X X coordinate

Y Y coordinate

Z Z coordinate

Member Function Documentation

◆ Offset()

```
PolylinePoint3D Offset ( double X,
                         double Y,
                         double Z
                       )
```

Applies an offset to the point and creates a new point

Parameters

X X offset to apply

Y Y offset to apply

Z Z offset to apply

Returns

New point with offset applied

◆ Scale()

```
PolylinePoint3D Scale ( double ScaleOriginX,
                        double ScaleOriginY,
                        double ScaleOriginZ,
                        double ScaleFactor
                      )
```

Scales the point location based on an origin for the scaling

Parameters

ScaleOriginX X-coordinate for scaling origin

ScaleOriginY Y-coordinate for scaling origin

ScaleOriginZ Z-coordinate for scaling origin

ScaleFactor Factor for scaling as a percentage

Returns

New point with scaling applied

Sketch Class Reference

A 2D sketch More...

Inherits ISweepPath, ICrossSection, IInstance, and ISelectableGeometry.

Inherited by **GearSketch**.

Public Types

enum **Constraints** {
Horizontal , **Vertical** , **Collinear** , **Coradial** ,
Coincident , **Perpendicular** , **Parallel** , **Tangent** ,
Equal , **Midpoint** , **Intersection** , **Symmetric** ,
Fix , **Normal**
}

Supported sketch constraints More...

Public Member Functions

CircularArc **AddArc** (**CircularArc** NewArc)

Adds a circular arc to the sketch More...

CircularArc **AddArcCenterStartAngle** (double CenterX, double CenterY, double StartX, double StartY, double Angle, bool IsReference)

Adds a circular arc using center, start and angle Arc goes anti-clockwise from start More...

CircularArc	**AddArcCenterStartEnd** (double CenterX, double CenterY, double StartX, double StartY, double EndX, double EndY, bool IsReference) Adds a circular arc using three points - center, start and end Arc goes anti-clockwise from start to end More...
Bspline	**AddBspline** (**Bspline** NewBspline) Adds a new bspline to the sketch More...
Bspline	**AddBspline** (int Order, List [] ControlPoints, List [] KnotVectors, List [] Weights, bool IsReference) Adds a **Bspline** to the sketch More...
Bspline	**AddBspline** (List [] Points, bool IsReference) Adds a **Bspline** to the sketch through a set of points More...
Circle	**AddCircle** (**Circle** NewCircle) Adds a circle to the sketch More...
Circle	**AddCircle** (double CenterX, double CenterY, double Diameter, bool IsReference) Adds a circle to the sketch More...
bool	**AddConstraint** (List [] **Figures**, **Constraints** Constraint) Adds a constraint to the sketch More...
bool	**AddConstraint** (ISketchFigure Figure, **Constraints** Constraint) Adds a constraint to the sketch More...
void	**AddDimension** (**Circle Circle**) Adds a dimension to the radius of a circle More...
void	**AddDimension** (**CircularArc** Arc) Adds a dimension to the radius of an arc More...
void	**AddDimension** (**SketchPoint** P1, **SketchPoint** P2) Adds a dimension to the sketch between two points More...
Ellipse	**AddEllipse** (double CenterX, double CenterY, double MajorAxisDiameter, double MinorMajorRatio, double MajorAxisAngle, bool IsReference) Adds an ellipse to the sketch More...
Ellipse	**AddEllipse** (double CenterX, double CenterY, double MajorX, double MajorY, double MinorX, double MinorY, bool IsReference) Adds an ellipse to the sketch using three points More...
Ellipse	**AddEllipse** (**Ellipse** NewEllipse) Adds an ellipse to the sketch More...
EllipticalArc	**AddEllipticalArc** (double CenterX, double CenterY, double StartX, double StartY, double EndX, double EndY, double MajorAxisDiameter, double MinorMajorRatio, double MajorAxisAngle, bool IsReference) Adds an elliptical arc to the sketch More...
EllipticalArc	**AddEllipticalArc** (**EllipticalArc** NewEllipticalArc) Adds an elliptical arc to the sketch More...
object	**AddFigure** (ISketchFigure NewFigure) Adds a figure to the sketch More...

Line	**AddLine** (double X1, double Y1, double X2, double Y2, bool IsReference) Adds a line to the sketch More...
Line	**AddLine** (List [] StartPoint, List [] EndPoint, bool IsReference) Adds a line to the sketch More...
Line	**AddLine** (**Line** NewLine) Adds a line to the sketch More...
void	**AddLines** (List [] Points, bool IsReference) Adds a polyline to the sketch More...
SketchPoint	**AddPoint** (double X, double Y) Adds a point to the sketch More...
SketchPoint	**AddPoint** (double X, double Y, bool IsReference) Adds a point to the sketch [DEPRECATED - DO NOT USE] More...
SketchPoint	**AddPoint** (**SketchPoint** NewPoint) Adds a point to the sketch More...
void	**AddPolygon** (double CenterX, double CenterY, double Diameter, int Sides, bool IsReference) Adds a regular polygon to the sketch More...
void	**AddPolyhole** (double CenterX, double CenterY, double Diameter, bool IsReference) Adds a polyhole to the sketch Create a "circle" whose size should be accurate regardless of the 3D printing method See: http://hydraraptor.blogspot.co.uk/2011/02/polyholes.html More...
void	**AddPolyline** (**Polyline Line**, bool IsReference) Adds a polyline to the sketch More...
void	**AddRectangle** (double BottomLeftX, double BottomLeftY, double TopRightX, double TopRightY, bool IsReference) Adds a rectangle to the sketch More...
void	**CopyFrom** (**Sketch** Source) Copies a sketch into this sketch More...
void	**CopyFrom** (**Sketch** Source, double Angle, double RotationCenterX, double RotationCenterY, double TranslateX, double TranslateY, double ScaleOriginX, double ScaleOriginY, double ScaleFactor) Copies a sketch into this sketch More...
void	**ExportSVG** (string FileName) Exports the sketch to an SVG More...
void	**ExportSVG** (string FileName, bool IncludeReferences) Exports the sketch to an SVG More...
void	**ExportSVG** (string FileName, bool IncludeReferences, double StrokeWidth, string StrokeColor, string StrokeLineCap, bool StrokeDashed, double StrokeDashLength, double ReferenceStrokeWidth, string ReferenceStrokeColor, string ReferenceStrokeLineCap, bool ReferenceStrokeDashed, double ReferenceStrokeDashLength) Exports the sketch to an SVG with specific styling More...
void	**FromXml** (string Xml) Adds elements to the sketch from XML More...
Part	**GetPart** ()

	Part that the sketch is defined on More...
Assembly	**GetSelectionAssembly** () The assembly that the sketch was selected on Only valid when a selection has been made More...
ISketchSurface	**GetSurface** () Gets the surface that the sketch was created on, e.g. a design plane or a face More...
List []	**GlobaltoPoint** (double x, double y, double z) Projects a 3D point in the part coordinate system into a point on the sketch More...
void	**ImportSVG** (string FileName) Imports an SVG into the sketch More...
void	**ImportSVG** (string FileName, double TranslateX, double TranslateY, double RotationAngle, bool TranslateThenRotate, bool NativeFigures) Imports an SVG into the sketch More...
void	**LoadXml** (string FileName) Loads the sketch from an XML file More...
List []	**PointtoGlobal** (double x, double y) Converts a point on the sketch into a 3D point in the part coordinate system More...
void	**SavetoXml** (string FileName) Saves the sketch to an XML file Does not support ellipses and elliptical arcs More...
void	**StartFaceMapping** (List [] EdgeEndPoint1, List [] EdgeEndPoint2) Starts mapping the face so that the specified edge is at [0, 0] Affects only the operation of the AddXXX functions and the GlobaltoPoint and PointtoGlobal functions, which will now use mapped X and Y values More...
void	**StartFaceMapping** (**Vertex** EdgeVertex1, **Vertex** EdgeVertex2) Starts mapping the face so that the specified edge is at [0, 0] More...
void	**StartMapping** (List [] Point1, List [] Point2, List [] PointAboveAxis) Starts mapping the sketch so that the specified line is at [0, 0] Affects only the operation of the AddXXX functions and the GlobaltoPoint and PointtoGlobal functions, which will now use mapped X and Y values More...
void	**StopFaceMapping** () Stops mapping the face
void	**StopMapping** () Stops mapping the sketch
string	**ToXml** () Saves the sketch to an XML string Does not support ellipses and elliptical arcs More...

Properties

List []	**Figures** `[get]` A list of figures (line, circle, circulararc, bspline, ellipse, elliptical arc) defined on the sketch
string	**Name** `[get]` Name of the sketch

SketchPoint **Origin** `[get]`

The point that defines the origin

Detailed Description

A 2D sketch

Member Enumeration Documentation

◆ Constraints

enum **Constraints**

Supported sketch constraints

Enumerator	
Horizontal	Horizontal constraint
Vertical	Vertical constraint
Collinear	Collinear constraint
Coradial	Coradial constraint
Coincident	Coincident constraint
Perpendicular	Perpendicular constraint
Parallel	Parallel constraint
Tangent	Tangent constraint
Equal	Equal size constraint
Midpoint	Midpoint constraint
Intersection	Intersection constraint
Symmetric	Symmetric constraint
Fix	Fixed constraint
Normal	Normal constraint

Member Function Documentation

◆ AddArc()

```
CircularArc AddArc ( CircularArc  NewArc )
```

Adds a circular arc to the sketch

Parameters

NewArc Arc to add

Returns

The added circular arc

◆ AddArcCenterStartAngle()

```
CircularArc AddArcCenterStartAngle ( double  CenterX,
                                     double  CenterY,
                                     double  StartX,
                                     double  StartY,
                                     double  Angle,
                                     bool    IsReference
                                   )
```

Adds a circular arc using center, start and angle Arc goes anti-clockwise from start

Parameters

CenterX	X coordinate for center
CenterY	Y coordinate for center
StartX	X coordinate for start
StartY	Y coordinate for start
Angle	Arc angle in degrees
IsReference	True if arc is a reference figure

Returns

The added circular arc

◆ AddArcCenterStartEnd()

```
CircularArc AddArcCenterStartEnd ( double  CenterX,
                                   double  CenterY,
                                   double  StartX,
                                   double  StartY,
                                   double  EndX,
                                   double  EndY,
                                   bool    IsReference
                                 )
```

Adds a circular arc using three points - center, start and end Arc goes anti-clockwise from start to end

Parameters

CenterX	X coordinate for center
CenterY	Y coordinate for center
StartX	X coordinate for start
StartY	Y coordinate for start
EndX	X coordinate for end
EndY	Y cordinate for end
IsReference	True if arc is a reference figure

Returns

The added circular arc

◆ AddBspline() [1/3]

```
Bspline AddBspline ( Bspline  NewBspline )
```

Adds a new bspline to the sketch

Parameters

NewBspline	**Bspline** to add to the sketch

Returns

The added **Bspline**

◆ AddBspline() [2/3]

```
Bspline AddBspline ( int      Order,
                     List []  ControlPoints,
                     List []  KnotVectors,
                     List []  Weights,
                     bool     IsReference
                   )
```

Adds a **Bspline** to the sketch

Parameters

Order	Order of the **Bspline** (Degree - 1)
ControlPoints	List of control points
KnotVectors	List of knot vectors
Weights	List of control point weights
IsReference	True to create a reference bspline

Returns

The created **Bspline**

◆ AddBspline() [3/3]

```
Bspline AddBspline ( List []  Points,
                     bool     IsReference
                   )
```

Adds a **Bspline** to the sketch through a set of points

Parameters

Points	List of points
IsReference	True to create a reference bspline

Returns

The created **Bspline**

◆ AddCircle() [1/2]

```
Circle AddCircle ( Circle  NewCircle )
```

Adds a circle to the sketch

Parameters

NewCircle **Circle** to add to sketch

Returns

The added circle

◆ AddCircle() [2/2]

```
Circle AddCircle ( double  CenterX,
                   double  CenterY,
                   double  Diameter,
                   bool    IsReference
                 )
```

Adds a circle to the sketch

Parameters

CenterX X coordinate of circle center

CenterY Y coordinate of circle center

Diameter **Circle** diameter

IsReference True to create a reference circle

Returns

A circle object

◆ AddConstraint() [1/2]

```
bool AddConstraint ( List []        Figures,
                     Constraints    Constraint
                   )
```

Adds a constraint to the sketch

Parameters

Figures	List of **Sketch** figures to constrain [Figure1, Figure2, ...] (**Circle**, **Line**, **CircularArc**, etc.)
Constraint	Constraint to apply

Returns

Returns True if constraint was added

◆ AddConstraint() [2/2]

```
bool AddConstraint ( ISketchFigure  Figure,
                     Constraints    Constraint
                   )
```

Adds a constraint to the sketch

Parameters

Figure	Figure to constrain (e.g. **Line**)
Constraint	Constraint to apply

Returns

True if constraint was added

◆ AddDimension() [1/3]

```
void AddDimension ( Circle  Circle )
```

Adds a dimension to the radius of a circle

Parameters

Circle	**Circle** to dimension

◆ AddDimension() [2/3]

void AddDimension (**CircularArc** Arc)

Adds a dimension to the radius of an arc

Parameters

Arc Arc to dimension

◆ AddDimension() [3/3]

void AddDimension (**SketchPoint** P1,
SketchPoint P2
)

Adds a dimension to the sketch between two points

Parameters

P1 First point

P2 Second point

◆ AddEllipse() [1/3]

```
Ellipse AddEllipse ( double  CenterX,
                     double  CenterY,
                     double  MajorAxisDiameter,
                     double  MinorMajorRatio,
                     double  MajorAxisAngle,
                     bool    IsReference
                   )
```

Adds an ellipse to the sketch

Parameters

CenterX	X coordinate of ellipse center
CenterY	Y coordinate of ellipse center
MajorAxisDiameter	Diameter of ellipse on major axis
MinorMajorRatio	Ratio of minor diameter to major diameter
MajorAxisAngle	Angle of major axis
IsReference	True to create a reference ellipse

Returns

An ellipse object

◆ AddEllipse() [2/3]

```
Ellipse AddEllipse ( double  CenterX,
                     double  CenterY,
                     double  MajorX,
                     double  MajorY,
                     double  MinorX,
                     double  MinorY,
                     bool    IsReference
                   )
```

Adds an ellipse to the sketch using three points

Parameters

CenterX	X coordinate of ellipse center
CenterY	Y coordinate of ellipse center
MajorX	X coordinate of ellipse on major axis
MajorY	Y coordinate of ellipse on major axis
MinorX	X coordinate of ellipse on minor axis
MinorY	Y coordinate of ellipse on minor axis
IsReference	True to create a reference ellipse

Returns

An ellipse object

◆ AddEllipse() [3/3]

```
Ellipse AddEllipse ( Ellipse  NewEllipse )
```

Adds an ellipse to the sketch

Parameters

NewEllipse	**Ellipse** to add

Returns

Added ellipse

◆ AddEllipticalArc() [1/2]

```
EllipticalArc AddEllipticalArc ( double  CenterX,
                                 double  CenterY,
                                 double  StartX,
                                 double  StartY,
                                 double  EndX,
                                 double  EndY,
                                 double  MajorAxisDiameter,
                                 double  MinorMajorRatio,
                                 double  MajorAxisAngle,
                                 bool    IsReference
                               )
```

Adds an elliptical arc to the sketch

Parameters

CenterX	X coordinate of arc center
CenterY	Y coordinate of arc center
StartX	X coorindate of arc start
StartY	Y coordinate of arc start
EndX	X coordinate of arc end
EndY	Y coordinate of arc end
MajorAxisDiameter	Diameter of ellipse on major axis
MinorMajorRatio	Ratio of minor diameter to major diameter
MajorAxisAngle	Angle of major axis
IsReference	True to create a reference elliptical arc

Returns

An elliptical arc object

◆ AddEllipticalArc() [2/2]

```
EllipticalArc AddEllipticalArc ( EllipticalArc  NewEllipticalArc )
```

Adds an elliptical arc to the sketch

Parameters

NewEllipticalArc	Elliptical arc to add

Returns

Added elliptical arc

◆ AddFigure()

```
object AddFigure ( ISketchFigure  NewFigure )
```

Adds a figure to the sketch

Parameters

NewFigure Figure to add

Returns

The added figure

◆ AddLine() [1/3]

```
Line AddLine ( double  X1,
               double  Y1,
               double  X2,
               double  Y2,
               bool    IsReference
             )
```

Adds a line to the sketch

Parameters

X1 Start point X

Y1 Start point Y

X2 End point X

Y2 End point Y

IsReference true to create a reference line

Returns

The added line

◆ AddLine() [2/3]

```
Line AddLine ( List []  StartPoint,
               List []  EndPoint,
               bool     IsReference
             )
```

Adds a line to the sketch

Parameters

StartPoint Start of line [X, Y]

EndPoint End of line [X, Y]

IsReference true if line is a reference line

Returns

The added line

◆ AddLine() [3/3]

```
Line AddLine ( Line  NewLine )
```

Adds a line to the sketch

Parameters

NewLine 2D line to add

Returns

The added line

◆ AddLines()

```
void AddLines ( List []  Points,
                bool     IsReference
              )
```

Adds a polyline to the sketch

Parameters

Points Set of points [Point1X, Point1Y, Point2X, Point2Y, ...]

IsReference true if line is a reference line

◆ AddPoint() [1/3]

```
SketchPoint AddPoint ( double  X,
                       double  Y
                     )
```

Adds a point to the sketch

Parameters

X **Point** X coordinate

Y **Point** Y coordinate

Returns

The created sketch point

◆ AddPoint() [2/3]

```
SketchPoint AddPoint ( double  X,
                       double  Y,
                       bool    IsReference
                     )
```

Adds a point to the sketch [DEPRECATED - DO NOT USE]

Parameters

X **Point** X coordinate

Y **Point** Y coordinate

IsReference Set to false

Returns

The added point

◆ AddPoint() [3/3]

```
SketchPoint AddPoint ( SketchPoint  NewPoint )
```

Adds a point to the sketch

Parameters

NewPoint **Point** to add

Returns

The added point

◆ AddPolygon()

```
void AddPolygon ( double  CenterX,
                  double  CenterY,
                  double  Diameter,
                  int     Sides,
                  bool    IsReference
                )
```

Adds a regular polygon to the sketch

Parameters

CenterX	X coordinate for polygon center
CenterY	Y coordinate for polygon center
Diameter	Diameter of polygon
Sides	Number of sides
IsReference	True to create a reference polygon

◆ AddPolyhole()

```
void AddPolyhole ( double  CenterX,
                   double  CenterY,
                   double  Diameter,
                   bool    IsReference
                 )
```

Adds a polyhole to the sketch Create a "circle" whose size should be accurate regardless of the 3D printing method See: http://hydraraptor.blogspot.co.uk/2011/02/polyholes.html

Parameters

CenterX	X coordinate for hole center
CenterY	Y coordinate for hole center
Diameter	Diameter of hole
IsReference	true if line is a reference line

◆ AddPolyline()

```
void AddPolyline ( Polyline  Line,
                   bool      IsReference
                 )
```

Adds a polyline to the sketch

Parameters

Line	Polyine to add
IsReference	true if line is a reference line

◆ AddRectangle()

```
void AddRectangle ( double  BottomLeftX,
                    double  BottomLeftY,
                    double  TopRightX,
                    double  TopRightY,
                    bool    IsReference
                  )
```

Adds a rectangle to the sketch

Parameters

BottomLeftX	X coordinate of bottom left corner
BottomLeftY	Y coordinate of bottom left corner
TopRightX	X coordinate of top right
TopRightY	Y coordinate of top right
IsReference	True to create a reference rectangle

◆ CopyFrom() [1/2]

```
void CopyFrom ( Sketch  Source )
```

Copies a sketch into this sketch

Parameters

Source	**Sketch** to copy from

◆ CopyFrom() [2/2]

```
void CopyFrom ( Sketch  Source,
                double  Angle,
                double  RotationCenterX,
                double  RotationCenterY,
                double  TranslateX,
                double  TranslateY,
                double  ScaleOriginX,
                double  ScaleOriginY,
                double  ScaleFactor
              )
```

Copies a sketch into this sketch

Parameters

Source	**Sketch** to copy from
Angle	Rotation angle
RotationCenterX	X-coodinate for center of rotation
RotationCenterY	Y-coordinate for center of rotation
TranslateX	Amount to move sketch in X direction
TranslateY	Amount to move sketch in Y direction
ScaleOriginX	X-coordinate for scaling origin
ScaleOriginY	Y-coordinate for scaling origin
ScaleFactor	Factor for scaling as a percentage

◆ ExportSVG() [1/3]

```
void ExportSVG ( string  FileName )
```

Exports the sketch to an SVG

Parameters

FileName	Path and name of SVG file to export to

◆ ExportSVG() [2/3]

```
void ExportSVG ( string  FileName,
                 bool    IncludeReferences
               )
```

Exports the sketch to an SVG

Parameters

FileName	Path and name of SVG file to export to
IncludeReferences	true to include reference figures in export

◆ ExportSVG() [3/3]

```
void ExportSVG ( string  FileName,
                 bool    IncludeReferences,
                 double  StrokeWidth,
                 string  StrokeColor,
                 string  StrokeLineCap,
                 bool    StrokeDashed,
                 double  StrokeDashLength,
                 double  ReferenceStrokeWidth,
                 string  ReferenceStrokeColor,
                 string  ReferenceStrokeLineCap,
                 bool    ReferenceStrokeDashed,
                 double  ReferenceStrokeDashLength
               )
```

Exports the sketch to an SVG with specific styling

Parameters

FileName	Path and name of SVG file to export to
IncludeReferences	true to include reference figures in export
StrokeWidth	Stroke width
StrokeColor	String containing name of stroke color
StrokeLineCap	String containing name of stroke line cap type
StrokeDashed	true if stroke dashed, false if solid
StrokeDashLength	Length of stroke dashes if dashed
ReferenceStrokeWidth	Reference stroke width
ReferenceStrokeColor	String containing name of reference stroke color
ReferenceStrokeLineCap	String containing name of reference stroke line cap type, can be: butt, round, square
ReferenceStrokeDashed	true if reference stroke dashed, false if solid
ReferenceStrokeDashLength	Length of reference stroke dashes if dashed

◆ FromXml()

```
void FromXml ( string  Xml )
```

Adds elements to the sketch from XML

Parameters

Xml	XML to parse

◆ GetPart()

Part GetPart ( )

Part that the sketch is defined on

Returns

Part that defines the sketch

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the sketch was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ GetSurface()

ISketchSurface GetSurface ( )

Gets the surface that the sketch was created on, e.g. a design plane or a face

Returns

Plane or **Face** object

◆ GlobaltoPoint()

```
List [] GlobaltoPoint ( double  x,
                        double  y,
                        double  z
                      )
```

Projects a 3D point in the part coordinate system into a point on the sketch

Parameters

x X coordinate of 3D point

y Y coordinate of 3D point

z Z coordinate of 3D point

Returns

Python list [x, y]

◆ ImportSVG() [1/2]

```
void ImportSVG ( string  FileName )
```

Imports an SVG into the sketch

Parameters

FileName Path and name of SVG file

◆ ImportSVG() [2/2]

```
void ImportSVG ( string  FileName,
                 double  TranslateX,
                 double  TranslateY,
                 double  RotationAngle,
                 bool    TranslateThenRotate,
                 bool    NativeFigures
               )
```

Imports an SVG into the sketch

Parameters

FileName	Path and name of SVG file
TranslateX	Amount to translate in the X direction
TranslateY	Amount to translate in the Y direction
RotationAngle	Amount to rotate in degrees
TranslateThenRotate	true to perform translation passed to this function before rotation passed to this function, false to reverse order
NativeFigures	true to create native circles and arcs when possible, false to always use Bezier curves

◆ LoadXml()

```
void LoadXml ( string  FileName )
```

Loads the sketch from an XML file

Parameters

FileName	Path and name of file to load from

◆ PointtoGlobal()

```
List [] PointtoGlobal ( double  x,
                        double  y
                      )
```

Converts a point on the sketch into a 3D point in the part coordinate system

Parameters

x X coordinate of point on sketch

y Y coordinate of point on sketch

Returns

Python list [x, y, z]

◆ SavetoXml()

```
void SavetoXml ( string  FileName )
```

Saves the sketch to an XML file Does not support ellipses and elliptical arcs

Parameters

FileName Path and name of file to save to

◆ StartFaceMapping() [1/2]

```
void StartFaceMapping ( List []  EdgeEndPoint1,
                        List []  EdgeEndPoint2
                      )
```

Starts mapping the face so that the specified edge is at [0, 0] Affects only the operation of the AddXXX functions and the GlobaltoPoint and PointtoGlobal functions, which will now use mapped X and Y values

Parameters

EdgeEndPoint1 First end point of edge [X, Y, Z]

EdgeEndPoint2 Second end point of edge [X, Y, Z]

◆ StartFaceMapping() [2/2]

```
void StartFaceMapping ( Vertex  EdgeVertex1,
                        Vertex  EdgeVertex2
                      )
```

Starts mapping the face so that the specified edge is at [0, 0]

Parameters

EdgeVertex1 Firrt vertex of edge

EdgeVertex2 Second vertex of edge

◆ StartMapping()

```
void StartMapping ( List []  Point1,
                    List []  Point2,
                    List []  PointAboveAxis
                  )
```

Starts mapping the sketch so that the specified line is at [0, 0] Affects only the operation of the AddXXX functions and the GlobaltoPoint and PointtoGlobal functions, which will now use mapped X and Y values

Parameters

Point1 First line end point [X, Y, Z]

Point2 Second line end point [X, Y, Z]

PointAboveAxis **Point** to be located above the X-axis

◆ ToXml()

```
string ToXml ( )
```

Saves the sketch to an XML string Does not support ellipses and elliptical arcs

Returns

XML string representing sketch

Sketch3D Class Reference

3D sketch More...

Inherits ISweepPath, IInstance, and ISelectableGeometry.

Public Member Functions

void	**AddArc** (**CircularArc3D** NewArc) Adds a circular arc to the sketch More...
void	**AddArcCenterStartEnd** (double CenterX, double CenterY, double CenterZ, double StartX, double StartY, double StartZ, double EndX, double EndY, double EndZ) Adds a circular arc using three points - center, start and end Arc goes anti-clockwise from start to end More...
void	**AddBspline** (**Bspline3D Bspline**) Adds a **Bspline** to the sketch More...
Bspline3D	**AddBspline** (List [] Points) Adds a **Bspline** to the sketch More...
void	**AddLine** (double X1, double Y1, double Z1, double X2, double Y2, double Z2) Adds a line to the sketch More...
void	**AddLine** (List [] StartPoint, List [] EndPoint) Adds a line to the sketch More...
void	**AddLine** (**Line3D** NewLine) Adds a line to the sketch More...
void	**AddLines** (List [] Points) Adds a polyline to the sketch More...
void	**AddPoint** (double X, double Y, double Z) Adds a point to the sketch More...
void	**AddPoint** (**SketchPoint3D** NewPoint) Adds a point to the sketch More...
void	**AddPolyline** (**Polyline3D Line**) Adds a polyline to the sketch More...
void	**FromXml** (string Xml) Adds elements to the sketch from XML More...
Part	**GetPart** () **Part** that the sketch is defined on More...
Assembly	**GetSelectionAssembly** () The assembly that the edge was selected on Only valid when a selection has been made More...
void	**LoadXml** (string FileName) Loads the sketch from an XML file More...
void	**SavetoXml** (string FileName) Saves the sketch to an XML file More...
string	**ToXml** () Saves the sketch to an XML string More...

Properties

List [] **Figures** `[get]`

A list of figures defines on the sketch, e.g. bspline

string **Name** `[get]`

Name of the sketch

Detailed Description

3D sketch

Member Function Documentation

◆ AddArc()

void AddArc (**CircularArc3D** NewArc)

Adds a circular arc to the sketch

Parameters

NewArc Arc to add

◆ AddArcCenterStartEnd()

```
void AddArcCenterStartEnd ( double  CenterX,
                            double  CenterY,
                            double  CenterZ,
                            double  StartX,
                            double  StartY,
                            double  StartZ,
                            double  EndX,
                            double  EndY,
                            double  EndZ
                          )
```

Adds a circular arc using three points - center, start and end Arc goes anti-clockwise from start to end

Parameters

CenterX	X coordinate for center
CenterY	Y coordinate for center
CenterZ	Z coordinate for center
StartX	X coordinate for start
StartY	Y coordinate for start
StartZ	Z coordinate for start
EndX	X coordinate for end
EndY	Y cordinate for end
EndZ	Z coordnate for end

◆ AddBspline() [1/2]

```
void AddBspline ( Bspline3D  Bspline )
```

Adds a **Bspline** to the sketch

Parameters

Bspline **Bspline**	to add

◆ AddBspline() [2/2]

Bspline3D AddBspline (List [] Points)

Adds a **Bspline** to the sketch

Parameters

Points List of control points [X1, Y1, Z1, X2, Y2, Z2, ...]

Returns

The **Bspline** object that was created

◆ AddLine() [1/3]

```
void AddLine ( double X1,
               double Y1,
               double Z1,
               double X2,
               double Y2,
               double Z2
             )
```

Adds a line to the sketch

Parameters

X1 Start point X

Y1 Start point Y

Z1 Start point Z

X2 End point X

Y2 End point Y

Z2 End point Z

◆ AddLine() [2/3]

```
void AddLine ( List []  StartPoint,
               List []  EndPoint
             )
```

Adds a line to the sketch

Parameters

StartPoint Start of line [X, Y, Z]

EndPoint End of line [X, Y, Z]

◆ AddLine() [3/3]

```
void AddLine ( Line3D  NewLine )
```

Adds a line to the sketch

Parameters

NewLine 3D line to add

◆ AddLines()

```
void AddLines ( List []  Points )
```

Adds a polyline to the sketch

Parameters

Points Set of points [Point1X, Point1Y, Point1Z, Point2X, Point2Y, Point2Z, ...]

◆ AddPoint() [1/2]

```
void AddPoint ( double  X,
                double  Y,
                double  Z
              )
```

Adds a point to the sketch

Parameters

X **Point** X coordinate

Y **Point** Y coordinate

Z **Point** Z coordinate

◆ AddPoint() [2/2]

```
void AddPoint ( SketchPoint3D  NewPoint )
```

Adds a point to the sketch

Parameters

NewPoint **Point** to add

◆ AddPolyline()

```
void AddPolyline ( Polyline3D  Line )
```

Adds a polyline to the sketch

Parameters

Line Polyine to add

◆ FromXml()

```
void FromXml ( string  Xml )
```

Adds elements to the sketch from XML

Parameters

Xml XML to parse

◆ GetPart()

Part GetPart ( )

Part that the sketch is defined on

Returns

Part

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

◆ LoadXml()

void LoadXml (string FileName)

Loads the sketch from an XML file

Parameters

FileName Path and name of file to load from

◆ SavetoXml()

void SavetoXml (string FileName)

Saves the sketch to an XML file

Parameters

FileName Path and name of file to save to

◆ ToXml()

string ToXml ( )

Saves the sketch to an XML string

Returns

XML string representing sketch

SketchPoint Class Reference

A 2D sketch point More...

Inherits ISketchFigure.

Public Member Functions

SketchPoint (double **X**, double **Y**, bool **IsReference**)

Creates a new sketch point which can be added to sketches More...

Properties

bool **IsReference** `[get, set]`

True if the point is a reference point, false if it is a regular point

double **X** `[get, set]`

X-coordinate of point

double **Y** `[get, set]`

Y-coordinate of point

Detailed Description

A 2D sketch point

Constructor & Destructor Documentation

◆ SketchPoint()

```
SketchPoint ( double  X,
              double  Y,
              bool    IsReference
            )
```

Creates a new sketch point which can be added to sketches

Parameters

X	X coordinate of sketch point
Y	Y coordinate of sketch point
IsReference	true to create a reference point, false to create a regular point

SketchPoint3D Class Reference

A 3D sketch point More...

Inherits ISketchFigure3D.

Public Member Functions

SketchPoint3D (double **X**, double **Y**, double **Z**, bool **IsReference**)

Creates a new 3D sketch point which can be added to sketches More...

Properties

bool **IsReference** `[get, set]`

True if the point is a reference point, false if it is a regular point

double **X** `[get, set]`

X-coordinate of point

double **Y** `[get, set]`

Y-coordinate of point

double **Z** `[get, set]`

Z-coordinate of point

Detailed Description

A 3D sketch point

Constructor & Destructor Documentation

◆ SketchPoint3D()

```
SketchPoint3D ( double X,
                double Y,
                double Z,
                bool   IsReference
              )
```

Creates a new 3D sketch point which can be added to sketches

Parameters

X	X coordinate of point
Y	Y coordinate of point
Z	Z coordinate of point
IsReference	true to create a reference point, false to create a regular point

ThreeD Class Reference

3D mathematical operations More...

Public Types

enum **RotationDirections** { **X** , **Y** , **Z** }

Public Member Functions

List [] **GetPerpendicularVector** (List [] Vector)

Gets a vector that is perpendicular to a vector More...

List [] **TransformPointUsingVectors** (List [] SourceVector, List [] DestinationVector, List [] **Point**)

Transforms a point based on two vectors More...

Detailed Description

3D mathematical operations

Member Enumeration Documentation

◆ RotationDirections

enum **RotationDirections**

Enumerator	
X	X rotation direction
Y	Y rotation direction
Z	Z rotation direction

Member Function Documentation

◆ GetPerpendicularVector()

List [] GetPerpendicularVector (List [] Vector)

Gets a vector that is perpendicular to a vector

Parameters

Vector Vector [X, Y, Z]

Returns

Vector that is perpendicular [X, Y, Z]

◆ TransformPointUsingVectors()

List [] TransformPointUsingVectors (List [] SourceVector,
List [] DestinationVector,
List [] Point
)

Transforms a point based on two vectors

Parameters

SourceVector Source vector [X, Y, Z]

DestinationVector Destination vector [X, Y, Z]

Point **Point** to transform [X, Y, Z]

Returns

Transformed point [X, Y, Z]

TwoD Class Reference

2D mathematical operations More...

Public Member Functions

List [] **GetPerpendicularVector** (List [] Vector)

Gets a vector that is perpendicular to a vector More...

List [] **NormalizeVector** (List [] Vector)

Normalizes a vector More...

List [] **RotatePoint** (List [] **Point**, double Angle)

Rotates a point More...

Detailed Description

2D mathematical operations

Member Function Documentation

◆ GetPerpendicularVector()

List [] GetPerpendicularVector (List [] Vector)

Gets a vector that is perpendicular to a vector

Parameters

Vector Vector [X, Y]

Returns

Vector that is perpendicular [X, Y]

◆ NormalizeVector()

List [] NormalizeVector (List [] Vector)

Normalizes a vector

Parameters

Vector Vector [X, Y]

Returns

Normalized vector [X, Y]

◆ RotatePoint()

List [] RotatePoint (List [] Point,
double Angle
)

Rotates a point

Parameters

Point **Point** to rotate as [X, Y]

Angle Angle to rotate in degrees

Returns

Rotated point as [RX, RY]

Vertex Class Reference

Describes a vertex More...

Inherits IChamferable, IConstrainable, IInstance, ISelectableGeometry, and IPoint.

Public Member Functions

Part **GetPart** ()

Part that the vertex is defined on More...

Assembly **GetSelectionAssembly** ()

The assembly that the edge was selected on Only valid when a selection has been made More...

Properties

string **Name** `[get]`

Name of the vertex

double **X** `[get]`

X-coordinate of vertex

double **Y** `[get]`

Y-coordinate of vertex

double **Z** `[get]`

Z-coordinate of vertex

Detailed Description

Describes a vertex

Member Function Documentation

◆ GetPart()

Part GetPart ( )

Part that the vertex is defined on

Returns

Part

◆ GetSelectionAssembly()

Assembly GetSelectionAssembly ( )

The assembly that the edge was selected on Only valid when a selection has been made

Returns

Assembly or null for no assembly

Windows Class Reference

Graphical user interface creation and interaction More...

Public Member Functions

Windows ()

Creates a new **Windows** object allowing user interfaces to be constructed

void	**CloseForm** (string SessionIdentifier) Close all currently open forms for a specific session More...
void	**DisableInput** (int Index) Disables an input More...
void	**EnableInput** (int Index) Enables an input More...
void	**ErrorDialog** (string Message, string Title) Shows an error window More...
object	**GetInputValue** (int Index) Gets the current value of an input More...
void	**InfoDialog** (string Message, string Title) Shows an information window More...
string	**OpenFileDialog** (string Title, string Filter, string DefaultExtension) Prompts user to select a file More...
List []	**OptionsDialog** (string Title, List [] Inputs, int InputAreaWidth, object InputChangedCallback, object UpdateUserInterfaceCallback) Shows a dialog prompting the user to enter values More...
List []	**OptionsDialog** (string Title, List [] Inputs, int InputAreaWidth=200) Shows a dialog prompting the user to enter values More...
bool	**QuestionDialog** (string Question, string Title) Shows a question window More...
string	**SaveFileDialog** (string Title, string Filter, string DefaultExtension) Prompts user to save a file More...
string	**SelectFolderDialog** (string CurrentFolder, string Description) Prompts the user to select a folder More...
void	**SetInputValue** (int Index, object Value) Sets the current value for an input More...
void	**SetStringList** (int Index, object Strings) Updates the list of strings for a stringlist input More...
void	**UtilityDialog** (string Title, string ActionButtonText, object ActionButtonCallback, object InputChangedCallback, List [] Inputs, int InputAreaWidth, object UpdateUserInterfaceCallback) Shows a dialog prompting the user to enter values The dialog remains open until the user clicks on the close button A callback function is called to give the input values to the script More...
void	**UtilityDialog** (string Title, string ActionButtonText, object ActionButtonCallback, object InputChangedCallback, List [] Inputs, int InputAreaWidth=200) Shows a dialog prompting the user to enter values The dialog remains open until the user clicks on the close button A callback function is called to give the input values to the script More...

Static Public Member Functions

static Form **GetDisplayedForm** (string SessionIdentifier)

Gets the currently displayed form for a specific session More...

Detailed Description

Graphical user interface creation and interaction

Member Function Documentation

◆ CloseForm()

void CloseForm (string SessionIdentifier)

Close all currently open forms for a specific session

Parameters

SessionIdentifier Identifier for session

◆ DisableInput()

void DisableInput (int Index)

Disables an input

Parameters

Index Index of the input

◆ EnableInput()

void EnableInput (int Index)

Enables an input

Parameters

Index Index of the input

◆ ErrorDialog()

```
void ErrorDialog ( string  Message,
                   string  Title
                 )
```

Shows an error window

Parameters

Message Error message

Title Title of window

◆ GetDisplayedForm()

```
static Form GetDisplayedForm ( string  SessionIdentifier )
```

static

Gets the currently displayed form for a specific session

Parameters

SessionIdentifier Identifier of session

Returns

Displayed form or null for none

◆ GetInputValue()

```
object GetInputValue ( int  Index )
```

Gets the current value of an input

Parameters

Index Index of the input

Returns

Current value

◆ InfoDialog()

```
void InfoDialog ( string  Message,
                  string  Title
                )
```

Shows an information window

Parameters

Message Message to show

Title Title of window

◆ OpenFileDialog()

```
string OpenFileDialog ( string  Title,
                        string  Filter,
                        string  DefaultExtension
                      )
```

Prompts user to select a file

Parameters

Title Title of dialog window

Filter File filter, example filter: '**Part** Files|*.AD_PRT'

DefaultExtension Default file extension, e.g. '.AD_PRT'

Returns

Path and name of selected file or empty string if canceled

◆ OptionsDialog() [1/2]

```
List [] OptionsDialog ( string  Title,
                        List []  Inputs,
                        int      InputAreaWidth,
                        object   InputChangedCallback,
                        object   UpdateUserInterfaceCallback
                      )
```

Shows a dialog prompting the user to enter values

Parameters

Title	Title of dialog window
Inputs	List of input definitions [[Name, Type, DefaultValue, OptionalSettings], [...]] Example: ['Image', **WindowsInputTypes.Image**, 'Logo.png']
InputAreaWidth	Width of input area
InputChangedCallback	Function called when an input is changed
UpdateUserInterfaceCallback	Function called after dialog is created to update the state of the dialog

Returns

List of entered values

◆ OptionsDialog() [2/2]

```
List [] OptionsDialog ( string  Title,
                        List []  Inputs,
                        int      InputAreaWidth = 200
                      )
```

Shows a dialog prompting the user to enter values

Parameters

Title	Title of dialog window
Inputs	List of input definitions [[Name, Type, DefaultValue], [...]]
InputAreaWidth	Width of input area, optional

Returns

List of entered values

◆ QuestionDialog()

```
bool QuestionDialog ( string  Question,
                      string  Title
                    )
```

Shows a question window

Parameters

Question Question to show

Title Title of window

Returns

true if 'yes' was clicked, false if 'no' was clicked

◆ SaveFileDialog()

```
string SaveFileDialog ( string  Title,
                        string  Filter,
                        string  DefaultExtension
                      )
```

Prompts user to save a file

Parameters

Title Title of dialog window

Filter File filter, example filter: '**Part** Files|*.AD_PRT'

DefaultExtension Default file extension, e.g. '.AD_PRT'

Returns

Path and name of selected file or empty string if canceled

◆ SelectFolderDialog()

```
string SelectFolderDialog ( string  CurrentFolder,
                            string  Description
                          )
```

Prompts the user to select a folder

Parameters

CurrentFolder The current folder, if any

Description Description of what is being chosen, shown to user

Returns

Path of selected folder or empty if canceled

◆ SetInputValue()

```
void SetInputValue ( int     Index,
                     object  Value
                   )
```

Sets the current value for an input

Parameters

Index Index of the input

Value Value to show

◆ SetStringList()

```
void SetStringList ( int     Index,
                     object  Strings
                   )
```

Updates the list of strings for a stringlist input

Parameters

Index Index of the stringlist input

Strings New list of strings to show

◆ UtilityDialog() [1/2]

```
void UtilityDialog ( string   Title,
                     string   ActionButtonText,
                     object   ActionButtonCallback,
                     object   InputChangedCallback,
                     List []  Inputs,
                     int      InputAreaWidth,
                     object   UpdateUserInterfaceCallback
                   )
```

Shows a dialog prompting the user to enter values The dialog remains open until the user clicks on the close button A callback function is called to give the input values to the script

Parameters

Title	Title of dialog window
ActionButtonText	Text for action button
ActionButtonCallback	Function called when the action button is clicked
InputChangedCallback	Function called when an input is changed
Inputs	List of input definitions [[Name, Type, DefaultValue, OptionalSettings], [...]] Example: ['Image', **WindowsInputTypes.Image**, 'Logo.png']
InputAreaWidth	Width of dialog input area
UpdateUserInterfaceCallback	Function called after dialog is created to update the state of the dialog

◆ UtilityDialog() [2/2]

```
void UtilityDialog ( string   Title,
                     string   ActionButtonText,
                     object   ActionButtonCallback,
                     object   InputChangedCallback,
                     List []  Inputs,
                     int      InputAreaWidth = 200
                   )
```

Shows a dialog prompting the user to enter values The dialog remains open until the user clicks on the close button A callback function is called to give the input values to the script

Parameters

Title	Title of dialog window
ActionButtonText	Text for action button
ActionButtonCallback	Function called when the action button is clicked
InputChangedCallback	Function called when an input is changed
Inputs	List of input definitions [[Name, Type, DefaultValue, OptionalSettings], [...]]
InputAreaWidth	Width of dialog input area, optional

Class Index

A | B | C | E | F | G | L | M | P | S | T | V | W

A

- **AssembledPart** (**AlibreScript.API**)
- **AssembledSubAssembly** (**AlibreScript.API**)
- **Assembly** (**AlibreScript.API**)
- **Axis** (**AlibreScript.API**)

B

- **Bspline** (**AlibreScript.API**)
- **Bspline3D** (**AlibreScript.API**)

C

- **Circle** (**AlibreScript.API**)
- **CircularArc** (**AlibreScript.API**)
- **CircularArc3D** (**AlibreScript.API**)
- **Configuration** (**AlibreScript.API**)
- **CSharp** (**AlibreScript.API**)

E

- **Edge** (**AlibreScript.API**)

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

[detail level 1 2]

▼ C **Assembly**	An assembly
C **AssembledSubAssembly**	A subassembly that is in an assembly
C **Axis**	An axis
C **Bspline**	Defines a **Bspline** that can be added to 2D sketches
C **Bspline3D**	Defines a **Bspline** that can be added to 3D sketches
C **Circle**	Describes a 2D circle, which can be added to 2D sketches
C **CircularArc**	Describes a 2D circular arc, which can be added to 2D sketches
C **CircularArc3D**	Describes a 3D circular arc, which can be added to 3D sketches
C **Configuration**	Describes a configuration
C **CSharp**	Provides access to the full Alibre Design **API** by running C# code See the Advanced **API** manual for details
C **Edge**	Describes an edge (can be filleted, chamfered, swept)
C **Ellipse**	Describes an ellipse used in 2D sketches
C **EllipticalArc**	Describes an elliptical arc used in 2D sketches
C **Face**	Describes a face (can be filleted, chamfered, used for sketches, used for loft cross sections)
C **Feature**	Describes a feature of an object, e.g. boss, cut
C **GlobalParameters**	A set of global parameters
C **Line**	Describes a 2D line, which can be added to 2D sketches
C **Line3D**	Describes a 3D line, which can be added to 3D sketches
C **Material**	**Material** densities in kg/cm3
C **Parameter**	Describes a parameter
▼ C **Part**	Object that represents a part
C **AssembledPart**	A part that is in an assembly
C **Plane**	A design plane. Can be used for creating sketches
C **Point**	A design point
C **Polyline**	A line constructed from a set of line segments
C **Polyline3D**	A 3D line constructed from a set of line segments
C **PolylinePoint**	A single point in a polyline
C **PolylinePoint3D**	A single point in a polyline for 3D sketches
▼ C **Sketch**	A 2D sketch
C **GearSketch**	A 2D sketch containing an involute gear profile. Can be treated as a regular sketch
C **Sketch3D**	3D sketch
C **SketchPoint**	A 2D sketch point
C **SketchPoint3D**	A 3D sketch point
C **ThreeD**	3D mathematical operations
C **TwoD**	2D mathematical operations
C **Vertex**	Describes a vertex
C **Windows**	Graphical user interface creation and interaction

Here is a list of all documented class members with links to the class documentation for each member:

- a -

- ABS : **Material**
- Activate() : **Configuration**
- Add3DSketch() : **Part**
- AddAlignConstraint() : **Assembly**
- AddAlignConstraint2() : **Assembly**
- AddAngleConstraint() : **Assembly**
- AddAngleConstraint2() : **Assembly**
- AddArc() : **Polyline**, **Sketch**, **Sketch3D**
- AddArcCenterStartAngle() : **Sketch**
- AddArcCenterStartEnd() : **Sketch**, **Sketch3D**
- AddAxis() : **Assembly**, **Part**
- AddBspline() : **Sketch**, **Sketch3D**
- AddChamfer() : **Part**
- AddChamferAngle() : **Part**
- AddCircle() : **Polyline**, **Sketch**
- AddConfiguration() : **Assembly**, **GlobalParameters**, **Part**
- AddConstraint() : **Sketch**
- AddDimension() : **Sketch**
- AddEllipse() : **Sketch**
- AddEllipticalArc() : **Sketch**
- AddExtrudeBoss() : **Part**
- AddExtrudeCut() : **Part**
- AddFastenerConstraint() : **Assembly**
- AddFastenerConstraint2() : **Assembly**
- AddFigure() : **Sketch**
- AddFillet() : **Part**
- AddGear() : **Part**
- AddGearConstraint() : **Assembly**
- AddGearDN() : **Part**
- AddGearDP() : **Part**
- AddGearNP() : **Part**
- AddLine() : **Sketch**, **Sketch3D**
- AddLines() : **Sketch**, **Sketch3D**
- AddLoftBoss() : **Part**
- AddLoftCut() : **Part**
- AddMateConstraint() : **Assembly**
- AddMateConstraint2() : **Assembly**
- AddNewPart() : **Assembly**
- AddNewSubAssembly() : **Assembly**
- AddOrientConstraint() : **Assembly**
- AddParameter() : **Assembly**, **GlobalParameters**, **Part**

- AddPart() : **Assembly**
- AddPlane() : **Assembly**, **Part**
- AddPoint() : **AssembledPart**, **Assembly**, **Part**, **Polyline**, **Polyline3D**, **Sketch**, **Sketch3D**
- AddPointFromCircularEdge() : **AssembledPart**, **Assembly**, **Part**
- AddPointFromToroidalFace() : **AssembledPart**, **Assembly**, **Part**
- AddPoints() : **Assembly**, **Part**
- AddPolygon() : **Sketch**
- AddPolyhole() : **Sketch**
- AddPolyline() : **Polyline**, **Polyline3D**, **Sketch**, **Sketch3D**
- AddRackAndPinionConstraint() : **Assembly**
- AddRectangle() : **Sketch**
- AddRevolveBoss() : **Part**
- AddRevolveCut() : **Part**
- AddScrewConstraint() : **Assembly**
- AddSketch() : **Part**
- AddSubAssembly() : **Assembly**
- AddSweepBoss() : **Part**
- AddSweepCut() : **Part**
- AddTangentConstraint() : **Assembly**
- AddVertexChamfer() : **Part**
- AnchorPart() : **Assembly**
- AnchorSubAssembly() : **Assembly**
- Angle : **CircularArc**, **CircularArc3D**
- ArcType : **CircularArc**, **CircularArc3D**
- Assembly() : **Assembly**
- AssemblyPointtoPartPoint() : **AssembledPart**
- AttachToExcel() : **Parameter**

Here is a list of all documented class members with links to the class documentation for each member:

- b -

- Bspline() : **Bspline**
- Bspline3D() : **Bspline3D**

Here is a list of all documented class members with links to the class documentation for each member:

- c -

- Center : **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**
- CenterPoint : **Circle**, **CircularArc**, **Ellipse**, **EllipticalArc**
- CenterX : **GearSketch**
- CenterY : **GearSketch**
- Circle() : **Circle**
- CircularArc() : **CircularArc**

- CircularArc3D() : **CircularArc3D**
- Clone() : **Polyline**, **Polyline3D**
- Close() : **Assembly**, **GlobalParameters**, **Part**
- CloseForm() : **Windows**
- Comment : **Assembly**, **Parameter**, **Part**
- Compile() : **CSharp**
- CompileAndRun() : **CSharp**
- Configurations : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **GlobalParameters**, **Part**
- ConstraintBoundsType : **Assembly**
- Constraints : **Sketch**
- ControlPoints : **Bspline**, **Bspline3D**
- CopyFrom() : **Sketch**
- CostCenter : **Assembly**, **Part**
- CreatedBy : **Assembly**, **Part**
- CreatedDate : **Assembly**, **Part**
- CreateUniqueName() : **Assembly**
- CreatingApplication : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- d -

- Density : **Assembly**, **Part**
- Description : **Assembly**, **Part**
- Diameter : **Edge**
- DiametralPitch : **GearSketch**
- DirectionType : **Part**
- DisableInput() : **Windows**
- DisplayUnits() : **Assembly**, **Part**
- DistanceTo() : **Face**
- DocumentNumber : **Assembly**, **Part**
- DuplicatePart() : **Assembly**
- DuplicateSubAssembly() : **Assembly**

Here is a list of all documented class members with links to the class documentation for each member:

- e -

- Ellipse() : **Ellipse**
- EllipticalArc() : **EllipticalArc**
- EnableInput() : **Windows**
- End : **CircularArc**, **EllipticalArc**, **Line**, **Line3D**
- EndCondition : **Part**
- EndPoint : **CircularArc**, **CircularArc3D**, **EllipticalArc**, **Line**, **Line3D**
- EngineeringApprovalDate : **Assembly**, **Part**

- EngineeringApprovedBy : **Assembly**, **Part**
- Equation : **Parameter**
- ErrorDialog() : **Windows**
- EstimatedCost : **Assembly**, **Part**
- ExcelCell : **Parameter**
- ExcelSheet : **Parameter**
- ExcelWorkbook : **Parameter**
- ExportBIP() : **Assembly**, **Part**
- ExportIGES() : **Assembly**, **Part**
- ExportRotatedSTL() : **Part**
- ExportSAT() : **Assembly**, **Part**
- ExportSTEP203() : **Assembly**, **Part**
- ExportSTEP214() : **Assembly**, **Part**
- ExportSTL() : **Assembly**, **Part**
- ExportSVG() : **Sketch**
- ExtendedMaterialInformation : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- f -

- Figures : **Sketch**, **Sketch3D**
- FileName : **Assembly**, **Part**
- FileTypes : **Part**
- FindIntersection() : **Polyline**
- FindIntersectionWithCircle() : **Polyline**
- FromXml() : **Sketch**, **Sketch3D**

Here is a list of all documented class members with links to the class documentation for each member:

- g -

- Get3DSketch() : **Part**
- GetActiveConfiguration() : **Assembly**, **GlobalParameters**, **Part**
- GetAdjoiningFaces() : **Face**
- GetArea() : **Face**
- GetAssembly() : **AssembledPart**
- GetAssemblyBoundingBox() : **AssembledPart**
- GetAssemblyVertices() : **AssembledPart**
- GetAxis() : **Assembly**, **Part**
- GetBoundingBox() : **Part**
- GetConfiguration() : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **GlobalParameters**, **Part**
- GetCoordinates() : **Point**
- GetCustomProperty() : **Assembly**, **Part**
- GetDisplayedForm() : **Windows**

- GetEdge() : **AssembledPart**, **Part**
- GetEdges() : **AssembledPart**, **Face**, **Part**
- GetFace() : **AssembledPart**, **Part**
- GetFaces() : **AssembledPart**, **Part**
- GetFeature() : **Part**
- GetInputValue() : **Windows**
- GetMappedOccurrence() : **AssembledPart**, **AssembledSubAssembly**
- GetNormalAt() : **Bspline**, **Bspline3D**
- GetParameter() : **Assembly**, **GlobalParameters**, **Part**
- GetPart() : **Assembly**, **Axis**, **Edge**, **Face**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- GetPartOrientation() : **Assembly**
- GetPerpendicularVector() : **ThreeD**, **TwoD**
- GetPlane() : **Assembly**, **Part**
- GetPoint() : **Assembly**, **Part**
- GetPointAt() : **Bspline**, **Bspline3D**
- GetSelectionAssembly() : **AssembledSubAssembly**, **Axis**, **Edge**, **Face**, **Part**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- GetSketch() : **Part**
- GetSubAssembly() : **Assembly**
- GetSurface() : **Sketch**
- GetUserData() : **Assembly**, **Part**
- GetVertex() : **Part**
- GetVertices() : **Edge**, **Face**, **Part**
- GetX() : **Bspline**, **Bspline3D**
- GetY() : **Bspline**, **Bspline3D**
- GetZ() : **Bspline3D**
- GlobalParameters() : **GlobalParameters**
- GlobaltoPoint() : **Sketch**

Here is a list of all documented class members with links to the class documentation for each member:

- h -

- Hide() : **Axis**, **Plane**, **Point**
- HideFeature() : **Part**
- HidePart() : **Assembly**
- HideSubAssembly() : **Assembly**

Here is a list of all documented class members with links to the class documentation for each member:

- i -

- ImportSVG() : **Sketch**
- InfoDialog() : **Windows**
- InsertPoint() : **Polyline**, **Polyline3D**
- IsActive : **Configuration**

- IsOpen() : **Part**
- IsParallel() : **Face**, **Plane**
- IsPointOnLine() : **Polyline**, **Polyline3D**
- IsRectangle() : **Face**
- IsReference : **Bspline**, **Bspline3D**, **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**, **Line**, **Line3D**, **SketchPoint**, **SketchPoint3D**

Here is a list of all documented class members with links to the class documentation for each member:

- j -

- Join() : **Polyline**, **Polyline3D**

Here is a list of all documented class members with links to the class documentation for each member:

- k -

- Keywords : **Assembly**, **Part**
- KnotVectors : **Bspline**, **Bspline3D**

Here is a list of all documented class members with links to the class documentation for each member:

- l -

- LastAuthor : **Assembly**, **Part**
- LastUpdateDate : **Assembly**, **Part**
- Length : **Bspline**, **Bspline3D**, **Circle**, **Edge**, **Line**, **Line3D**
- Line() : **Line**
- Line3D() : **Line3D**
- LoadXml() : **Sketch**, **Sketch3D**
- LockAll() : **Configuration**

Here is a list of all documented class members with links to the class documentation for each member:

- m -

- MajorAxisAngle : **Ellipse**, **EllipticalArc**
- ManufacturingApprovedBy : **Assembly**, **Part**
- ManufacturingApprovedDate : **Assembly**, **Part**
- Mass : **Part**
- Material : **Assembly**, **Part**
- MinorMajorRatio : **Ellipse**, **EllipticalArc**
- ModifiedInformation : **Assembly**, **Part**
- MovePart() : **Assembly**
- MoveParts() : **Assembly**
- MoveSubAssemblies() : **Assembly**
- MoveSubAssembly() : **Assembly**

Here is a list of all documented class members with links to the class documentation for each member:

- n -

- Name : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **Axis**, **Configuration**, **Edge**, **Face**, **Feature**, **GlobalParameters**, **Parameter**, **Part**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- NonUniformScale() : **Part**
- NormalizeVector() : **TwoD**
- Number : **Assembly**, **Part**
- NumberofTeeth : **GearSketch**

Here is a list of all documented class members with links to the class documentation for each member:

- o -

- Offset() : **Polyline**, **Polyline3D**, **PolylinePoint**, **PolylinePoint3D**
- OpenFileDialog() : **Windows**
- OptionsDialog() : **Windows**
- Order : **Bspline**, **Bspline3D**
- Origin : **Assembly**, **Part**, **Sketch**

Here is a list of all documented class members with links to the class documentation for each member:

- p -

- Parameters : **Assembly**, **GlobalParameters**, **Part**
- Part() : **Part**
- PartPointtoAssemblyPoint() : **AssembledPart**
- Parts : **Assembly**
- PauseUpdating() : **Assembly**, **Part**
- PitchDiameter : **GearSketch**
- PLA : **Material**
- PointtoGlobal() : **Sketch**
- Polyline() : **Polyline**
- Polyline3D() : **Polyline3D**
- PolylinePoint() : **PolylinePoint**
- PolylinePoint3D() : **PolylinePoint3D**
- PressureAngle : **GearSketch**
- Product : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- q -

- QuestionDialog() : **Windows**

Here is a list of all documented class members with links to the class documentation for each member:

- r -

- Radius : **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**
- RawValue : **Parameter**
- ReceivedFrom : **Assembly**, **Part**
- Regenerate() : **Assembly**, **Part**
- RemoveDuplicates() : **Polyline**, **Polyline3D**
- RemoveFeature() : **Part**
- RemovePlane() : **Part**
- RemovePoint() : **Part**
- RemoveSketch() : **Part**
- ResumeUpdating() : **Assembly**, **Part**
- Revision : **Assembly**, **Part**
- RotatePart() : **Assembly**
- RotateParts() : **Assembly**
- RotatePoint() : **TwoD**
- RotateSubAssemblies() : **Assembly**
- RotateSubAssembly() : **Assembly**
- RotateZ() : **Polyline**, **PolylinePoint**
- RotationDirections : **ThreeD**
- Run() : **CSharp**

Here is a list of all documented class members with links to the class documentation for each member:

- s -

- Save() : **Assembly**, **GlobalParameters**, **Part**
- SaveAll() : **Assembly**
- SaveAs() : **Assembly**, **GlobalParameters**, **Part**
- SaveFileDialog() : **Windows**
- SaveSnapshot() : **Assembly**, **Part**
- SaveThumbnail() : **Assembly**, **Part**
- SavetoXml() : **Sketch**, **Sketch3D**
- Scale() : **Part**, **PolylinePoint**, **PolylinePoint3D**
- Select() : **Part**
- SelectFolderDialog() : **Windows**
- Selections : **Assembly**, **Part**
- SetColor() : **Feature**, **Part**
- SetCustomProperty() : **Assembly**, **Part**
- SetInputValue() : **Windows**
- SetLocks() : **Configuration**
- SetStringList() : **Windows**
- SetUserData() : **Assembly**, **Part**
- Show() : **Axis**, **Plane**, **Point**
- ShowFeature() : **Part**

- ShowPart() : **Assembly**
- ShowSubAssembly() : **Assembly**
- SketchPoint() : **SketchPoint**
- SketchPoint3D() : **SketchPoint3D**
- SplitAtPoint() : **Polyline**, **Polyline3D**
- Start : **CircularArc**, **EllipticalArc**, **Line**, **Line3D**
- StartFaceMapping() : **Sketch**
- StartMapping() : **Sketch**
- StartPoint : **CircularArc**, **CircularArc3D**, **EllipticalArc**, **Line**, **Line3D**
- StockSize : **Assembly**, **Part**
- StopFaceMapping() : **Sketch**
- StopMapping() : **Sketch**
- SubAssemblies : **Assembly**
- Subdivide() : **Bspline**, **Bspline3D**
- SubdivideGetNormals() : **Bspline3D**
- Supplier : **Assembly**, **Part**
- SuppressFeature() : **Part**
- SuppressPart() : **Assembly**
- SuppressSubAssembly() : **Assembly**

Here is a list of all documented class members with links to the class documentation for each member:

- t -

- Title : **Assembly**, **Part**
- ToXml() : **Sketch**, **Sketch3D**
- TransformPointUsingVectors() : **ThreeD**
- Type : **CircularArc**, **CircularArc3D**, **Parameter**

Here is a list of all documented class members with links to the class documentation for each member:

- u -

- UnanchorPart() : **Assembly**
- UnanchorSubAssembly() : **Assembly**
- Units : **Parameter**
- UnlockAll() : **Configuration**
- UnsuppressFeature() : **Part**
- UnsuppressPart() : **Assembly**
- UnsuppressSubAssembly() : **Assembly**
- UtilityDialog() : **Windows**

Here is a list of all documented class members with links to the class documentation for each member:

- v -

- Value : **Parameter**
- Vendor : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- w -

- WebLink : **Assembly**, **Part**
- Weights : **Bspline**, **Bspline3D**
- Windows() : **Windows**

Here is a list of all documented class members with links to the class documentation for each member:

- x -

- X : **Point**, **PolylinePoint**, **PolylinePoint3D**, **SketchPoint**, **SketchPoint3D**, **Vertex**
- XAxis : **Assembly**, **Part**
- XYPlane : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- y -

- Y : **Point**, **PolylinePoint**, **PolylinePoint3D**, **SketchPoint**, **SketchPoint3D**, **Vertex**
- YAxis : **Assembly**, **Part**
- YZPlane : **Assembly**, **Part**

Here is a list of all documented class members with links to the class documentation for each member:

- z -

- Z : **Point**, **PolylinePoint3D**, **SketchPoint3D**, **Vertex**
- ZAxis : **Assembly**, **Part**
- ZXPlane : **Assembly**, **Part**

- a -

- Activate() : **Configuration**
- Add3DSketch() : **Part**
- AddAlignConstraint() : **Assembly**
- AddAlignConstraint2() : **Assembly**
- AddAngleConstraint() : **Assembly**
- AddAngleConstraint2() : **Assembly**
- AddArc() : **Polyline**, **Sketch**, **Sketch3D**
- AddArcCenterStartAngle() : **Sketch**
- AddArcCenterStartEnd() : **Sketch**, **Sketch3D**

- AddAxis() : **Assembly**, **Part**
- AddBspline() : **Sketch**, **Sketch3D**
- AddChamfer() : **Part**
- AddChamferAngle() : **Part**
- AddCircle() : **Polyline**, **Sketch**
- AddConfiguration() : **Assembly**, **GlobalParameters**, **Part**
- AddConstraint() : **Sketch**
- AddDimension() : **Sketch**
- AddEllipse() : **Sketch**
- AddEllipticalArc() : **Sketch**
- AddExtrudeBoss() : **Part**
- AddExtrudeCut() : **Part**
- AddFastenerConstraint() : **Assembly**
- AddFastenerConstraint2() : **Assembly**
- AddFigure() : **Sketch**
- AddFillet() : **Part**
- AddGear() : **Part**
- AddGearConstraint() : **Assembly**
- AddGearDN() : **Part**
- AddGearDP() : **Part**
- AddGearNP() : **Part**
- AddLine() : **Sketch**, **Sketch3D**
- AddLines() : **Sketch**, **Sketch3D**
- AddLoftBoss() : **Part**
- AddLoftCut() : **Part**
- AddMateConstraint() : **Assembly**
- AddMateConstraint2() : **Assembly**
- AddNewPart() : **Assembly**
- AddNewSubAssembly() : **Assembly**
- AddOrientConstraint() : **Assembly**
- AddParameter() : **Assembly**, **GlobalParameters**, **Part**
- AddPart() : **Assembly**
- AddPlane() : **Assembly**, **Part**
- AddPoint() : **AssembledPart**, **Assembly**, **Part**, **Polyline**, **Polyline3D**, **Sketch**, **Sketch3D**
- AddPointFromCircularEdge() : **AssembledPart**, **Assembly**, **Part**
- AddPointFromToroidalFace() : **AssembledPart**, **Assembly**, **Part**
- AddPoints() : **Assembly**, **Part**
- AddPolygon() : **Sketch**
- AddPolyhole() : **Sketch**
- AddPolyline() : **Polyline**, **Polyline3D**, **Sketch**, **Sketch3D**
- AddRackAndPinionConstraint() : **Assembly**
- AddRectangle() : **Sketch**
- AddRevolveBoss() : **Part**
- AddRevolveCut() : **Part**

- AddScrewConstraint() : **Assembly**
- AddSketch() : **Part**
- AddSubAssembly() : **Assembly**
- AddSweepBoss() : **Part**
- AddSweepCut() : **Part**
- AddTangentConstraint() : **Assembly**
- AddVertexChamfer() : **Part**
- AnchorPart() : **Assembly**
- AnchorSubAssembly() : **Assembly**
- Assembly() : **Assembly**
- AssemblyPointtoPartPoint() : **AssembledPart**
- AttachToExcel() : **Parameter**

- b -

- Bspline() : **Bspline**
- Bspline3D() : **Bspline3D**

- c -

- Circle() : **Circle**
- CircularArc() : **CircularArc**
- CircularArc3D() : **CircularArc3D**
- Clone() : **Polyline**, **Polyline3D**
- Close() : **Assembly**, **GlobalParameters**, **Part**
- CloseForm() : **Windows**
- Compile() : **CSharp**
- CompileAndRun() : **CSharp**
- CopyFrom() : **Sketch**
- CreateUniqueName() : **Assembly**

- d -

- DisableInput() : **Windows**
- DisplayUnits() : **Assembly**, **Part**
- DistanceTo() : **Face**
- DuplicatePart() : **Assembly**
- DuplicateSubAssembly() : **Assembly**

- e -

- Ellipse() : **Ellipse**
- EllipticalArc() : **EllipticalArc**
- EnableInput() : **Windows**
- ErrorDialog() : **Windows**
- ExportBIP() : **Assembly**, **Part**
- ExportIGES() : **Assembly**, **Part**
- ExportRotatedSTL() : **Part**
- ExportSAT() : **Assembly**, **Part**
- ExportSTEP203() : **Assembly**, **Part**
- ExportSTEP214() : **Assembly**, **Part**
- ExportSTL() : **Assembly**, **Part**
- ExportSVG() : **Sketch**

- f -

- FindIntersection() : **Polyline**
- FindIntersectionWithCircle() : **Polyline**
- FromXml() : **Sketch**, **Sketch3D**

- g -

- Get3DSketch() : **Part**
- GetActiveConfiguration() : **Assembly**, **GlobalParameters**, **Part**
- GetAdjoiningFaces() : **Face**
- GetArea() : **Face**
- GetAssembly() : **AssembledPart**
- GetAssemblyBoundingBox() : **AssembledPart**
- GetAssemblyVertices() : **AssembledPart**
- GetAxis() : **Assembly**, **Part**
- GetBoundingBox() : **Part**
- GetConfiguration() : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **GlobalParameters**, **Part**
- GetCoordinates() : **Point**
- GetCustomProperty() : **Assembly**, **Part**
- GetDisplayedForm() : **Windows**
- GetEdge() : **AssembledPart**, **Part**
- GetEdges() : **AssembledPart**, **Face**, **Part**
- GetFace() : **AssembledPart**, **Part**
- GetFaces() : **AssembledPart**, **Part**
- GetFeature() : **Part**
- GetInputValue() : **Windows**

- GetMappedOccurrence() : **AssembledPart**, **AssembledSubAssembly**
- GetNormalAt() : **Bspline**, **Bspline3D**
- GetParameter() : **Assembly**, **GlobalParameters**, **Part**
- GetPart() : **Assembly**, **Axis**, **Edge**, **Face**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- GetPartOrientation() : **Assembly**
- GetPerpendicularVector() : **ThreeD**, **TwoD**
- GetPlane() : **Assembly**, **Part**
- GetPoint() : **Assembly**, **Part**
- GetPointAt() : **Bspline**, **Bspline3D**
- GetSelectionAssembly() : **AssembledSubAssembly**, **Axis**, **Edge**, **Face**, **Part**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- GetSketch() : **Part**
- GetSubAssembly() : **Assembly**
- GetSurface() : **Sketch**
- GetUserData() : **Assembly**, **Part**
- GetVertex() : **Part**
- GetVertices() : **Edge**, **Face**, **Part**
- GetX() : **Bspline**, **Bspline3D**
- GetY() : **Bspline**, **Bspline3D**
- GetZ() : **Bspline3D**
- GlobalParameters() : **GlobalParameters**
- GlobaltoPoint() : **Sketch**

- h -

- Hide() : **Axis**, **Plane**, **Point**
- HideFeature() : **Part**
- HidePart() : **Assembly**
- HideSubAssembly() : **Assembly**

- i -

- ImportSVG() : **Sketch**
- InfoDialog() : **Windows**
- InsertPoint() : **Polyline**, **Polyline3D**
- IsOpen() : **Part**
- IsParallel() : **Face**, **Plane**
- IsPointOnLine() : **Polyline**, **Polyline3D**
- IsRectangle() : **Face**

- j -

- Join() : **Polyline**, **Polyline3D**

- l -

- Line() : **Line**
- Line3D() : **Line3D**
- LoadXml() : **Sketch**, **Sketch3D**
- LockAll() : **Configuration**

- m -

- MovePart() : **Assembly**
- MoveParts() : **Assembly**
- MoveSubAssemblies() : **Assembly**
- MoveSubAssembly() : **Assembly**

- n -

- NonUniformScale() : **Part**
- NormalizeVector() : **TwoD**

- o -

- Offset() : **Polyline**, **Polyline3D**, **PolylinePoint**, **PolylinePoint3D**
- OpenFileDialog() : **Windows**
- OptionsDialog() : **Windows**

- p -

- Part() : **Part**
- PartPointtoAssemblyPoint() : **AssembledPart**
- PauseUpdating() : **Assembly**, **Part**
- PointtoGlobal() : **Sketch**
- Polyline() : **Polyline**
- Polyline3D() : **Polyline3D**
- PolylinePoint() : **PolylinePoint**

- PolylinePoint3D() : **PolylinePoint3D**

- q -

- QuestionDialog() : **Windows**

- r -

- Regenerate() : **Assembly**, **Part**
- RemoveDuplicates() : **Polyline**, **Polyline3D**
- RemoveFeature() : **Part**
- RemovePlane() : **Part**
- RemovePoint() : **Part**
- RemoveSketch() : **Part**
- ResumeUpdating() : **Assembly**, **Part**
- RotatePart() : **Assembly**
- RotateParts() : **Assembly**
- RotatePoint() : **TwoD**
- RotateSubAssemblies() : **Assembly**
- RotateSubAssembly() : **Assembly**
- RotateZ() : **Polyline**, **PolylinePoint**
- Run() : **CSharp**

- s -

- Save() : **Assembly**, **GlobalParameters**, **Part**
- SaveAll() : **Assembly**
- SaveAs() : **Assembly**, **GlobalParameters**, **Part**
- SaveFileDialog() : **Windows**
- SaveSnapshot() : **Assembly**, **Part**
- SaveThumbnail() : **Assembly**, **Part**
- SavetoXml() : **Sketch**, **Sketch3D**
- Scale() : **Part**, **PolylinePoint**, **PolylinePoint3D**
- Select() : **Part**
- SelectFolderDialog() : **Windows**
- SetColor() : **Feature**, **Part**
- SetCustomProperty() : **Assembly**, **Part**
- SetInputValue() : **Windows**
- SetLocks() : **Configuration**
- SetStringList() : **Windows**

- SetUserData() : **Assembly**, **Part**
- Show() : **Axis**, **Plane**, **Point**
- ShowFeature() : **Part**
- ShowPart() : **Assembly**
- ShowSubAssembly() : **Assembly**
- SketchPoint() : **SketchPoint**
- SketchPoint3D() : **SketchPoint3D**
- SplitAtPoint() : **Polyline**, **Polyline3D**
- StartFaceMapping() : **Sketch**
- StartMapping() : **Sketch**
- StopFaceMapping() : **Sketch**
- StopMapping() : **Sketch**
- Subdivide() : **Bspline**, **Bspline3D**
- SubdivideGetNormals() : **Bspline3D**
- SuppressFeature() : **Part**
- SuppressPart() : **Assembly**
- SuppressSubAssembly() : **Assembly**

- t -

- ToXml() : **Sketch**, **Sketch3D**
- TransformPointUsingVectors() : **ThreeD**

- u -

- UnanchorPart() : **Assembly**
- UnanchorSubAssembly() : **Assembly**
- UnlockAll() : **Configuration**
- UnsuppressFeature() : **Part**
- UnsuppressPart() : **Assembly**
- UnsuppressSubAssembly() : **Assembly**
- UtilityDialog() : **Windows**

- w -

- Windows() : **Windows**

- ABS : **Material**
- CenterX : **GearSketch**

- CenterY : **GearSketch**
- DiametralPitch : **GearSketch**
- NumberofTeeth : **GearSketch**
- PitchDiameter : **GearSketch**
- PLA : **Material**
- PressureAngle : **GearSketch**
- X : **PolylinePoint**, **PolylinePoint3D**
- Y : **PolylinePoint**, **PolylinePoint3D**
- Z : **PolylinePoint3D**

- ArcType : **CircularArc**, **CircularArc3D**
- ConstraintBoundsType : **Assembly**
- Constraints : **Sketch**
- DirectionType : **Part**
- EndCondition : **Part**
- FileTypes : **Part**
- RotationDirections : **ThreeD**

- a -

- Angle : **CircularArc**, **CircularArc3D**

- c -

- Center : **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**
- CenterPoint : **Circle**, **CircularArc**, **Ellipse**, **EllipticalArc**
- Comment : **Assembly**, **Parameter**, **Part**
- Configurations : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **GlobalParameters**, **Part**
- ControlPoints : **Bspline**, **Bspline3D**
- CostCenter : **Assembly**, **Part**
- CreatedBy : **Assembly**, **Part**
- CreatedDate : **Assembly**, **Part**
- CreatingApplication : **Assembly**, **Part**

- d -

- Density : **Assembly**, **Part**
- Description : **Assembly**, **Part**
- Diameter : **Edge**
- DocumentNumber : **Assembly**, **Part**

- e -

- End : **CircularArc**, **EllipticalArc**, **Line**, **Line3D**

- EndPoint : **CircularArc**, **CircularArc3D**, **EllipticalArc**, **Line**, **Line3D**
- EngineeringApprovalDate : **Assembly**, **Part**
- EngineeringApprovedBy : **Assembly**, **Part**
- Equation : **Parameter**
- EstimatedCost : **Assembly**, **Part**
- ExcelCell : **Parameter**
- ExcelSheet : **Parameter**
- ExcelWorkbook : **Parameter**
- ExtendedMaterialInformation : **Assembly**, **Part**

- f -

- Figures : **Sketch**, **Sketch3D**
- FileName : **Assembly**, **Part**

- i -

- IsActive : **Configuration**
- IsReference : **Bspline**, **Bspline3D**, **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**, **Line**, **Line3D**, **SketchPoint**, **SketchPoint3D**

- k -

- Keywords : **Assembly**, **Part**
- KnotVectors : **Bspline**, **Bspline3D**

- l -

- LastAuthor : **Assembly**, **Part**
- LastUpdateDate : **Assembly**, **Part**
- Length : **Bspline**, **Bspline3D**, **Circle**, **Edge**, **Line**, **Line3D**

- m -

- MajorAxisAngle : **Ellipse**, **EllipticalArc**
- ManufacturingApprovedBy : **Assembly**, **Part**
- ManufacturingApprovedDate : **Assembly**, **Part**
- Mass : **Part**
- Material : **Assembly**, **Part**
- MinorMajorRatio : **Ellipse**, **EllipticalArc**
- ModifiedInformation : **Assembly**, **Part**

- n -

- Name : **AssembledPart**, **AssembledSubAssembly**, **Assembly**, **Axis**, **Configuration**, **Edge**, **Face**, **Feature**, **GlobalParameters**, **Parameter**, **Part**, **Plane**, **Point**, **Sketch**, **Sketch3D**, **Vertex**
- Number : **Assembly**, **Part**

- o -

- Order : **Bspline**, **Bspline3D**

- Origin : **Assembly**, **Part**, **Sketch**

- p -

- Parameters : **Assembly**, **GlobalParameters**, **Part**
- Parts : **Assembly**
- Product : **Assembly**, **Part**

- r -

- Radius : **Circle**, **CircularArc**, **CircularArc3D**, **Ellipse**, **EllipticalArc**
- RawValue : **Parameter**
- ReceivedFrom : **Assembly**, **Part**
- Revision : **Assembly**, **Part**

- s -

- Selections : **Assembly**, **Part**
- Start : **CircularArc**, **EllipticalArc**, **Line**, **Line3D**
- StartPoint : **CircularArc**, **CircularArc3D**, **EllipticalArc**, **Line**, **Line3D**
- StockSize : **Assembly**, **Part**
- SubAssemblies : **Assembly**
- Supplier : **Assembly**, **Part**

- t -

- Title : **Assembly**, **Part**
- Type : **CircularArc**, **CircularArc3D**, **Parameter**

- u -

- Units : **Parameter**

- v -

- Value : **Parameter**
- Vendor : **Assembly**, **Part**

- w -

- WebLink : **Assembly**, **Part**
- Weights : **Bspline**, **Bspline3D**

- x -

- X : **Point**, **SketchPoint**, **SketchPoint3D**, **Vertex**
- XAxis : **Assembly**, **Part**
- XYPlane : **Assembly**, **Part**

- y -

- Y : **Point**, **SketchPoint**, **SketchPoint3D**, **Vertex**
- YAxis : **Assembly**, **Part**
- YZPlane : **Assembly**, **Part**

- z -

- Z : **Point**, **SketchPoint3D**, **Vertex**
- ZAxis : **Assembly**, **Part**
- ZXPlane : **Assembly**, **Part**